

Toolbox Introduction



Yaskawa has created several IEC-61131 projects for MotionWorks IEC which can be imported for use by another project as a User Library, or "Toolbox." These toolboxes were designed to save time by providing application code for a wide variety of situations.

- Cam Toolbox contains functions that increase the power of the PLCopen cam function in the firmware library by providing extras such as functions for calculating motion profiles, making adjustments based on latch inputs, and EStop recovery.
- Communications toolbox provides advanced communication protocol function blocks (DNS, SMTP, FTP).
- File Read / Write Toolbox builds upon the basic file manipulation functions available in the ProConOS firmware library to more quickly read and write application data files.
- Group Toolbox is the successor to Gantry Toolbox and provides enhancements to PLCopen Part 4 for interpolation, including G-Code support. Its GroupCommManager and Pendant_Driver function blocks make it easy to add manual mode and position teaching support for any group based application.
- Kinematics Toolbox contains forward and inverse kinematics for selected mechanisms.
- Math Toolbox provides compatibility with the built in function that include EN and ENO outputs, and also provides other tools such as ATAN2, and Floating Point Remainder (REM).
- PackML is both a Template and Toolbox for designing applications to take advantage of the PackML specification. It emphasizes machine state and transition logic and provides predefined PackML data structures.
- PLCopen Toolbox contains functions that build upon the PLCopen standard functions. It can serve as a starting point for every project.
- **NEW!** Winding Toolbox contains functions which perform calculations and other control for various winding applications.
- Yaskawa Toolbox contains functions that add basic functionality, such as PID Control, or a Moving Average Filter.

A toolbox or user library is just another project. What makes it a user library is the import method. When a project is imported as a user library, only the functions, function blocks and datatypes are available to the main project. None of the hardware specific information of the user library applies.

Please refer to the document [TN.MCD.08.130](#) on www.yaskawa.com for a comprehensive look at how user libraries can increase programming efficiency by reducing development time.

See our [Youtube channel](#) for video tutorials and examples for MotionWorks IEC and many of our toolboxes.



Data Type: AXIS_ARRAY

AXIS_ARRAY is a sub structure of the [AXES_GROUP_REF](#) structure which identifies the physical motors or virtual axes that participate in the group.

Data Type Declaration

*	Element	Data Type	Description	Usage
C	AXIS_ARRAY	ARRAY: [1..32] OF AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number). Note that this array is 1 based, not zero.	MyAXES_GROUP_REF.AxisRef[1].AxisNum

Example

Watch Window

Variable	Value	Type
[-] GantryDemo		AXES_GROUP_REF
[+] Name		STRING32
Handle	1	UINT
[+] Mechanism		STRING32
[-] AxisRef		AXIS_ARRAY
[-] [1]		AXIS_REF
AxisNum	5	UINT
[-] [2]		AXIS_REF
AxisNum	3	UINT
[-] [3]		AXIS_REF
AxisNum	4	UINT
[-] [4]		AXIS_REF
AxisNum	6	UINT



AXES_GROUP_REF

An AXES_GROUP_REF identifies a group of axes that work together as a single mechanical unit. AXES_GROUP_REF is used as a VAR_IN_OUT for PLCopen Part 4 - Interpolation Function Blocks.

Variables of this type are automatically declared in the Global Variables grid when the MotionWorks IEC project is configured with a Group in the Hardware Configuration. When a group is configured with Mechatrolink axes, the group structure is mapped to a %M memory area at specific location, and a portion of the contents of the structure is updated by the MPiec firmware at the task interval to which the group I/O is assigned.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAXES_GROUP_REF	AXES_GROUP_REF		
C	Name	STRING32	ARRAY[1..32] OF BYTE. An array of ASCII characters representing the Group name as entered in the Hardware Configuration. Tip: Use the BUF_TO_STRING function block to convert this data to a STRING type. See example below.	MyAXES_GROUP_REF.Name
C	Handle	UINT	This is a unique value used internally to identify the group with lower level function blocks.	MyAXES_GROUP_REF.Handle
C	Mechanism	STRING32	ARRAY[1..32] OF BYTE] Reserved for future use.	MyAXES_GROUP_REF.Mechanism
C	AxisRef	<u>AXIS_ARRAY</u>	Array of AXIS_REF identifying the physical motors or virtual axes. Note that this array is 1 based, not 0.	MyAXES_GROUP_REF.Axis[1].AxisNum
C	Padding1	UINT	reserved	
C	Status	<u>GROUP_STATUS</u>	Structure containing status and alarm information.	
C	Axis	<u>GROUP_COORD_SYS</u>	Structure containing position, velocity and acceleration data for the Axis Coordinate System.	MyAxesGroup.Axis.CmdPos[1]
C	Machine	<u>GROUP_COORD_SYS</u>	Structure containing position, velocity and acceleration data for the Machine Coordinate System.	MyAxesGroup.Machine.CmdPos[1]
C	World	<u>GROUP_COORD_SYS</u>	Structure containing position, velocity and acceleration data for the World Coordinate System.	MyAxesGroup.World.CmdPos[1]
C	Part	<u>GROUP_COORD_SYS</u>	Structure containing position, velocity and acceleration data for the Part Coordinate System.	MyAxesGroup.Part.CmdPos[1]
C	Tool	<u>GROUP_COORD_SYS</u>	Structure containing position, velocity and acceleration data for the Tool Coordinate System.	MyAxesGroup.Tool.CmdPos[1]

*	Element	Data Type	Description	Usage
	MyAXES_GROUP_REF	AXES_GROUP_REF		
C	Limits	GROUP_LIMITS	Position, Velocity, Acceleration and Jerk limits for the Axes, Machine and Tool Coordinate Systems	MyAxesGroup.Limits.MachineLimits.MaxVelocity [1]
C	Reserved	Reserved	Reserved for future expansion.	---
C	Host_ID	INT	0 = Group is operated locally on the MPiec via Mechatrolink. 1 = group is operated remotely via MLX200 gateway.	MyAXES_GROUP_REF.Host_ID
C	Mechanism_ID	INT	Indicates the type of mechanism: 0 = Gantry, 1 = Scara, 2 = MPP3, custom = 32767	MyAXES_GROUP_REF.Mechanism_ID
C	Interface_ID	INT	Required when Host_ID = 1. This identifies on which of the potentially 8 MLX200 interfaces in the system the AxesGroup exists.	MyAXES_GROUP_REF.Interface_ID
C	Device_ID	INT	Robot number within the MLX200 Gateway. This value must be 1.	MyAXES_GROUP_REF.Device_ID
C	MaxLinearVelocity	LREAL	Maximum linear velocity of the TCP	MyAXES_GROUP_REF.MaxLinearVelocity
C	MaxAngularVelocity	LREAL	Maximum angular velocity of the TCP	MyAXES_GROUP_REF.MaxAngularVelocity
C	MaxLinearAcceleration	LREAL	Maximum linear acceleration of the TCP	MyAXES_GROUP_REF.MaxLinearAcceleration
C	MaxAngularAcceleration	LREAL	Maximum angular acceleration of the TCP	MyAXES_GROUP_REF.MaxAngularAcceleration
C	MaxLinearJerk	LREAL	Maximum linear jerk of the TCP	MyAXES_GROUP_REF.MaxLinearJerk
C	MaxAngularJerk	LREAL	Maximum angular jerk of the TCP	MyAXES_GROUP_REF.MaxAngularJerk
C	ActiveCoordinateFrame	INT	ACS, MCS, PCS...	MyAXES_GROUP_REF.ActiveCoordinateFrame
C	ActiveCoordinateSystem	INT	Cartesian, Polar, Spherical. (Only Cartesian is supported.) Others for future use.	MyAXES_GROUP_REF.ActiveCoordinateSystem
C	ActiveTool	INT	As entered in the UserApplicationData structure.	MyAXES_GROUP_REF.ActiveTool
C	ActivePartFrame	INT	As entered in the UserApplicationData structure.	MyAXES_GROUP_REF.ActivePartFrame



Data Type: AXIS_REF

The AXIS_REF data type identifies an axis and thus provides the interface to the hardware or virtual axes. AXIS_REF is referenced as a VAR_IN_OUT in all function blocks. It is represented as an input and an output connected by a horizontal line in the graphical representation of a function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAxisRef	AXIS_REF		
U	AxisNum	UINT	The value of AxisNum is determined by the logical axis number assigned by the Hardware Configuration. See the Configuration tab under each axis.	MyAxisRef.AxisNum

Example

```
MyServo.AxisNum:=UINT#3;
```



AxisParameterStruct

For use with the [CamSlave_FeedToLength](#) and [CamSlave_WindowCheck](#) function blocks.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAxisParameterStruct	AxisParameterStruct		
C	ActualPosition	LREAL	1000	MyAxisParameterStruct.ActualPosition
C	ActualPositionCyclic	LREAL	1005	MyAxisParameterStruct.ActualPositionCyclic
C	ActualPositionNonCyclic	LREAL	1006	MyAxisParameterStruct.ActualPositionNonCyclic
C	ActualTorque	LREAL	1004	MyAxisParameterStruct.ActualTorque
C	ActualVelocity	LREAL	1001	MyAxisParameterStruct.ActualVelocity
C	AtVelocity	BOOL	1141	MyAxisParameterStruct.AtVelocity
C	BufferedMotionBlocks	LREAL	1600	MyAxisParameterStruct.BufferedMotionBlocks
C	CamMasterCycle	LREAL	1512	MyAxisParameterStruct.CamMasterCycle
C	CamMasterPosition	LREAL	1500	MyAxisParameterStruct.CamMasterPosition
C	CamMasterShiftedCyclic	LREAL	1502	MyAxisParameterStruct.CamMasterShiftedCyclic
C	CamMasterShiftedPosition	LREAL	1501	MyAxisParameterStruct.CamMasterShiftedPosition
C	CamMasterScale	LREAL	1510	MyAxisParameterStruct.CamMasterScale
C	CamMasterShift	LREAL	1511	MyAxisParameterStruct.CamMasterShift
C	CamOffset	LREAL	1531	MyAxisParameterStruct.CamOffset
C	CamScale	LREAL	1530	MyAxisParameterStruct.CamScale
C	CamShiftRemaining	LREAL	1513	MyAxisParameterStruct.CamShiftRemaining
C	CamState	LREAL	1540	MyAxisParameterStruct.CamState
C	CamTableIDEngaged	LREAL	1541	MyAxisParameterStruct.CamTableIDEngaged
C	CamTableOutput	LREAL	1520	MyAxisParameterStruct.CamTableOutput
C	CommandedAcceleration	LREAL	1012	MyAxisParameterStruct.CommandedAcceleration
C	CommandedPosition	LREAL	1010	MyAxisParameterStruct.CommandedPosition
C	CommandedPositionCyclic	LREAL	1015	MyAxisParameterStruct.CommandedPositionCyclic
C	CommandedPositionNonCyclic	LREAL	1016	MyAxisParameterStruct.CommandedPositionNonCyclic
C	CommandedTorque	LREAL	1014	MyAxisParameterStruct.CommandedTorque
C	CommandedVelocity	LREAL	1011	MyAxisParameterStruct.CommandedVelocity
C	InPosition	BOOL	1140	MyAxisParameterStruct.InPosition
C	LatchPositionNonCyclic	LREAL	1031	MyAxisParameterStruct.LatchPositionNonCyclic
C	PositionError	LREAL	1130	MyAxisParameterStruct.PositionError
C	PositionWindow	LREAL	1120	MyAxisParameterStruct.PositionWindow



Data Type: COORD_SYS_LIMITS_6

COORD_SYS_LIMITS_6 is a sub structure of [AXES_GROUP_REF](#).Limits.WorldLimits and PartLimits. This structure is for the MCS and PCS Coordinate systems.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAxesGroup.Limits.AxisLimits	COORD_SYS_LIMITS		
C	MinPosition	MC_COORD_REF		MyAxesGroup.Limits.AxisLimits.MinPosition[2]
C	MinVelocity	MC_COORD_REF		MyAxesGroup.Limits.AxisLimits.MinVelocity[1]
C	MinAcceleration	MC_COORD_REF		MyAxesGroup.Limits.AxisLimits.MinAcceleration[4]
C	MinJerk	MC_COORD_REF		MyAxesGroup.Limits.AxisLimits.Jerk[1]
C	MaxPosition	MC_COORD_REF		MyAxesGroup.Limits.AxisLimits.MaxPosition[3]
C	MaxVelocity	MC_COORD_REF		MyAxesGroup.Limits.AxisLimits.MaxVelocity[2]
C	MaxAcceleration	MC_COORD_REF		MyAxesGroup.Limits.AxisLimits.MaxAcceleration[2]
C	MaxJerk	MC_COORD_REF		MyAxesGroup.Limits.AxisLimits.MaxJerk[5]



Data Type: GROUP_LIMITS

GROUP_LIMITS is a sub structure of [AXES_GROUP_REF](#) and contains the following information.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAxesGroup.Limits	GROUP_LIMITS		
C	AxisLimits	COORD_SYS_LIMITS		MyAxesGroup.Limits.AxisLimits
C	MachineLimits	COORD_SYS_LIMITS		MyAxesGroup.Limits.MachineLimits
C	WorldLimits	COORD_SYS_LIMITS_6		MyAxesGroup.Limits.WorldLimits
C	PartLimits	COORD_SYS_LIMITS_6		MyAxesGroup.Limits.PartLimits



Data Type: GROUP_STATUS

GROUP_STATUS is a sub structure of [AXES_GROUP_REF](#) and contains the following information.

Data Type Declaration

*	Element	Data Type	Description	Usage
	GROUP_STATUS			
C	Active	BOOL	Indicates if the group is made active via the MC_GroupEnable function block.	MyAXES_GROUP_REF.Status.Active
C	Padding	BYTE	Reserved for future use	
C	State	UINT	Reserved for future use	MyAXES_GROUP_REF.Status.State
C	Alarm	UDINT	Indicates if there is an alarm for the group or any axis of the group.	MyAXES_GROUP_REF.Status.Alarm
C	NumMotionSegments	UINT	The number of motion segments currently in the motion queue.	MyAXES_GROUP_REF.Status.NumMotionSegments
C	FreeMotionSegments	UINT	The number of motion segments available to be added to the motion queue.	MyAXES_GROUP_REF.Status.FreeMotionSegments
C	AxisStatus	GRP_AXIS_STATUS_ARRAY	Array which reports the power status of each axis. For Mechatrolink Servo axes, this is equivalent to the PON flag.	MyAXES_GROUP_REF.Status.AxisStatus[1]
	<i>Reserved</i>	<i>256 bytes of reserved space.</i>	---	---



Data Type: VECTOR

The VECTOR data type specifies a coordinate position for a grouped system. This datatype has other applications also, such as [TransitionParameter](#).

Data Type Declaration

*	Element	Data Type	Description	Usage
U/C	Vector	Array [1..32] of LREAL	This datatype is generically used in several instances where an array of LREAL values is required.	MyVector [1]

Example

Consider the following code for a simple XYZ gantry mechanism. MyVector and AnotherVector can be connected directly to the MC_MoveLinearAbsolute or MC_MoveLinearRelative function blocks "Position" VAR_INPUT.

```
MyVector[1]:=LREAL#25.0; (* Specify the X axis position *)
```

```
MyVector[2]:=LREAL#25.0; (* Specify the Y axis position *)
```

```
MyVector[3]:=LREAL#3.5; (* Specify the Z axis position *)
```

```
AnotherVector[1]:=LREAL#1..765; (* Specify the X axis position *)
```

```
AnotherVector[2]:=LREAL#2.131; (* Specify the Y axis position *)
```

```
AnotherVector[3]:=LREAL#0.220; (* Specify the Z axis position *)
```



RTC Struct

This datatype is for use with the [Y_SetRTC](#) function block from the Y_Motion firmware library and the RealTimeClock function block from the Yaskawa Toolbox.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyRTCStruct	RTC_STRUCT		
U/C	Year	INT		MyRTCStruct.Year
U/C	Month	INT		MyRTCStruct.Month
U/C	Day	INT		MyRTCStruct.Day
U/C	Hour	INT		MyRTCStruct.Hour
U/C	Minute	INT		MyRTCStruct.Minute
U/C	Second	INT		MyRTCStruct.Second
U/C	Millisecond	INT		MyRTCStruct.Millisecond



Data Type: TRIGGER_REF

This data type is for use with the MC_TouchProbe and MC_AbortTrigger function blocks.

MC_TouchProbe requires a trigger referenced via a variable of the type TRIGGER_REF.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyTriggerRef	TRIGGER_REF		
U	Bit	UINT	See chart below for required value based on the device and physical input pin used.	MyTriggerRef.Bit
U	ID	UINT	Unique ID for each trigger on the same axis; used as a link between MC_TouchProbe and MC_AbortTrigger. Typically the value should be zero as only one trigger per axis is supported on Mechatrolink II and the MP2000iec style option cards. If using simultaneous MC_TouchProbe function blocks with a Mechatrolink-III Servopack, specify unique IDs.	MyTriggerRef.ID
U	Input	Input_REF	Reserved for future use	MyTriggerRef.Input.ID
U	Pattern	DETECTION_PATTERN	Reserved for future use	MyTriggerRef.Pattern

Notes

Prior to firmware 2.5, the TRIGGER_REF.ID element was not used by the MP2000iec controllers and any value would be accepted. In 2.5, the TRIGGER_REF.ID field must be a zero for MP2000iec series controllers. In older help documentation, an example showed the ID set to 1, and many users set TRIGGER_REF.ID to 1 in their projects, which causes MC_TouchProbe to output ErrorID 4630 when using firmware 2.5 or higher. The solution is to set MyTriggerRef.ID to zero which will work for all firmware versions. Only on the MP3000iec controllers can another value be used for the case of multiple latches configured on the same axis.

The following chart details the correct values for the TRIGGER_REF structure based on the hardware latch to be detected.

Device	Signal	Hardware Pin #	Software Default Variable Name	TRIGGER_REF			
				Bit	ID	Input	Pattern
						Input_Ref ID	
UINT	UINT	UINT	ENUM				
LIO-01	Encoder C Channel	A3/B3	n/a	0	Set to 0 for LIO, MP2600iec, or Mechatrolink -II ServoPacks. If using Mechatrolink-III ServoPack, more than one trigger can be captured simultaneously. In this case, use a different ID for each MC_TouchProbe function block. For use with MC_AbortTrigger	Not used	For future use
	DI-01	A22	M□□_DI_01	1			
LIO-02	Encoder C Channel	A3/B3	n/a	0			
	DI-01	A22	M□□_DI_01	1			
LIO-06	Encoder C Channel	35	n/a	0			
	DI-01	39	M□□_DI_01	1			
MP2600	External C Channel	35	n/a	0			
	Cn13 DI-01	39	MO1_DI_01	1			
SGDH	C Channel	n/a	n/a	0			
	EXT1	44	AX□□_SI4_EXT1	1			
	EXT2	45	AX□□_SI5_EXT2	2			
	EXT3	46	AX□□_SI6_EXT3	3			
SGDS	C Channel	n/a	n/a	0			
	EXT1	10*	AX□□_SI4_EXT1	1			
	EXT2	11*	AX□□_SI5_EXT2	2			
	EXT3	12*	AX□□_SI6_EXT3	3			
SGDV	C Channel	n/a	n/a	0			
	EXT1	10*	AX□□_SI4_EXT1	1			
	EXT2	11*	AX□□_SI5_EXT2	2			
	EXT3	12*	AX□□_SI6_EXT3	3			
SGD7S	C Channel	n/a	n/a	0			
	EXT1	10*	AX□□_SI4_EXT1	1			
	EXT2	11*	AX□□_SI5_EXT2	2			
	EXT3	12*	AX□□_SI6_EXT3	3			
SGD7W (A)	C Channel	n/a	n/a	0			
	EXT_A1	6*	AX□□_SI4_EXT1	1			
	EXT_A2	7*	AX□□_SI5_EXT2	2			
	EXT_A3	8*	AX□□_SI6_EXT3	3			
SGD7W (B)	C Channel	n/a	n/a	0			
	EXT_B1	12*	AX□□_SI4_EXT1	1			
	EXT_B2	13*	AX□□_SI5_EXT2	2			
	EXT_B3	14*	AX□□_SI6_EXT3	3			

* denotes the default pin, which can be changed by setting Pn 511 in the drive.



Data Type: Y_DISENGAGE_DATA

This data type is for use with the [Y_CamOut](#) function block. Y_DisengageMethod#AtPosition is the only disengage method supported. To disengage the slave from a master when the machine is already stopped, use MC_Stop for the slave axis.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyYDisengageData	Y_DISENGAGE_DATA		
U	EndMode	Y_DisengageMethod	Enumeration: 0:Y_DisengageMethod#AtPosition 1:Y_DisengageMethod#Immediate 2:Y_DisengageMethod#EndOfProfile	MyYDisengageData.EndMode
U	RampOut	INT	Reserved for future use	MyYDisengageData.RampOut
U	RampOutData1	LREAL	Reserved for future use	MyYDisengageData.RampOutData1
U	RampOutData2	LREAL	Reserved for future use	MyYDisengageData.RampOutData2
U	RampOutData3	LREAL	Reserved for future use	MyYDisengageData.RampOutData3
U	RampOutData4	LREAL	Reserved for future use	MyYDisengageData.RampOutData4

Code Example:

Data Type: Y_ENGAGE_DATA



This data type is for use with the [Y_CamIn](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
---	---------	-----------	-------------	-------

	MyYEngageData	Y_ENGAGE_DATA		
U	StartMode	Y_EngageMethod	<p>Enumeration:</p> <p>0:Y_EngageMethod#AtPosition (Default)</p> <p>The slave will engage when the master position is within the range of [EngagePosition +/- (EngageWindow/2)]. Master-Relative is ignored. Use this setting for normal circumstances. The intended usage requires setting YCamIn.Execute:=TRUE at some point before the master and slave are to be synchronized. The motion engine, operating at the MECHATROLINK or dual port RAM update interval will monitor for the exact position to start the camming process.</p> <p>1:Y_EngageMethod#Immediate</p> <p>Y_CamIn does not wait for the master position to reach the EngagePosition. The EngagePosition and the EngageWindow inputs are ignored. This mode is intended for use when the master is not moving, such as during fault or E-Stop recovery in the middle of a cam cycle. In this scenario, the slave may be moved to the equivalent cam position of the master, then the cam can be re-engaged immediately using MasterRelative:=FALSE to preserve the original synchronization. If MasterRelative=TRUE, then CamMasterShift (Parameter 1511) is adjusted so that the master position at the time YCamIn.Execute changes to TRUE corresponds to the start of the table domain. This scenario would change the synchronization between the master and slave. Immediate Mode is not recommended for application scenarios where the master is in motion, as a position drift or phase lag may be introduced.</p> <p>2:Y_EngageMethod#Linked</p> <p>The new cam profile will be switched on the fly at the end of the current cam table. This mode is intended for use when cams with different Machine Cycles are to be run without stopping. Use the Linked mode or applications where the product size must be changed on the fly. Do not use linked mode if Y_CamShift is used for the same slave axis.</p> <p>3: Y_EngageMethod#AtAbsolutePosition (requires firmware 3.2.0 or higher)</p> <p>The slave will engage when the master axis absolute position crosses the position specified by the EngagePosition input of the function block. The intended usage requires setting YCamIn.Execute:=TRUE at some point before the master's absolute position reaches the specified position. The motion engine, operating at the MECHATROLINK or dual port RAM update interval will monitor for the exact position to start the camming process. The slave will start to follow the master when the master crosses the specified Engage position according to the Y_CamIn function block input.</p>	MyYEngageData.StartMode
U	MasterRelative	BOOL	Only MasterRelative:=FALSE is supported.	MyYEngageData.MasterRelative
U	SlaveAbsolute	BOOL	When SlaveAbsolute:=FALSE, the initial position of the slave will be used as the base offset for the camming operation. For example, if the cam slave table data goes from 0 to 20, back to 0, but the slave's commanded position when the engage event occurs is 4.5, the slave will travel from 4.5 to 24.5, and back to 4.5. If SlaveAbsolute:=TRUE for the same scenario, the slave will jump from 4.5 to 0 when the engage event occurs. (This is not desirable behavior.) If SlaveAbsolute mode is used, the application program must ensure that the slave axis is positioned at the proper position for camming to operate correctly.	MyYEngageData.SlaveAbsolute
	RampIn	INT	Reserved for future use	---
	RampInData1	LREAL	Reserved for future use	---
	RampInData2	LREAL	Reserved for future use	---
	RampInData3	LREAL	Reserved for future use	---
	RampInData4	LREAL	Reserved for future use	---

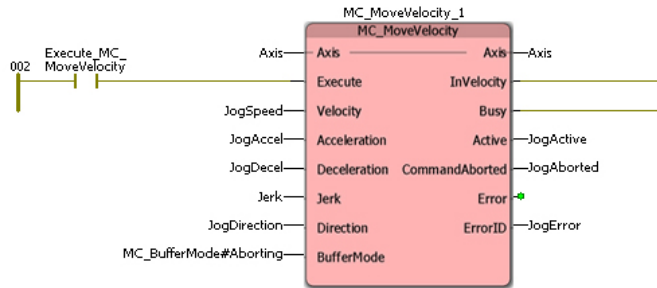
Code Example:

```
MyEngageData.StartMode:=Y_EngageMethod#AtAbsolutePosition;
```



Enumerated Types Shared from PLCopen Plus

Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to zero-based integers (INT). Therefore, the first value equates to zero, the second to 1, etc. The format for enumerated types is as follows: ENUM:(0, 1, 2...) as displayed in the example below (MC_BufferMode#Aborting).



The following is an alphabetical listing of the Enumerated Types defined in the MotionBlockTypes file:

Enumerated Type	#INT Value	Enum Value	Description
MC_BufferMode	0	Aborting	This is the Default mode. The FB aborts an ongoing motion and the command affects the axis immediately.
	1	Buffered	The FB affects the axis as soon as the previous movement is complete. The axis will stop between the movements.
	2	BlendingLow	The FB controls the axis after the previous FB has finished, but the axis will not stop between the movements. The velocity is blended with the lowest velocity of both commands.
	3	BlendingPrevious	The FB controls the axis after the previous FB has finished (equivalent to buffered), but the axis will not stop between the movements. Blending with the velocity of the previous move.
	4	BlendingNext	The FB controls the axis after the previous FB has finished, but the axis will not stop between the movements. Blending with velocity of this (next) function.
	5	BlendingHigh	The FB controls the axis after the previous FB has finished (equivalent to buffered), but the axis will not stop between the movements. Blending with highest velocity of the previous and this (next) function.
MC_Detection_Pattern	0	Rising_Edge	
	1	Falling_Edge	

Enumerated Type	#INT Value	Enum Value	Description
MC_Direction	0	Positive_Direction	In a rotary application, forces the axis to move in a positive direction.
	1	Shortest_Way	For use in applications where the Load Type is configured as a rotary or modularized axis.
	2	Negative_Direction	In a rotary application, forces the axis to move in a negative direction.
	3	Current_Direction	For use in applications where the Load Type is configured as a rotary or modularized axis. Only applies if an existing move is in progress and another function block such as MC_MoveAbsolute or MC_MoveRelative is executed. Once the axis is at StandStill, using MC_Direction_CurrentDirection will default to the positive direction
MC_ExecutionMode	0	Immediately	Functionality is immediately valid. May influence ongoing motion, but not the state.
	1	Delayed	Functionality is valid when the ongoing motion command set Done, Aborted, or Error outputs.
	2	Queued	Functionality is valid when ALL previous motion commands set Done, Aborted, or Error outputs.
MC_PathChoice	0	ClockWise	
	1	CounterClockWise	
MC_SwitchMode	0	On	Unsupported
	1	Off	Unsupported
	2	EdgeOn	
	3	EdgeOff	Unsupported
	4	EdgeSwitchPositive	Unsupported
	5	EdgeSwitchNegative	Unsupported
MC_TransitionMode	0	TMNone	No transition curve inserted. Motion blocks are not modified. The only TransitionMode usable with the "Buffered" BufferMode.
	1	TMStartVelocity	Transition with given start velocity. Adds a segment to the motion queue unless motion is tangent.
	2	TMConstantVelocity	Transition with given constant velocity. Adds a segment to the motion queue unless motion is tangent.
	3	TMCornerDistance	Transition with given corner distance. Adds a segment to the motion queue unless motion is tangent.
	4	TMMaxCornerDeviation	Geometry limited parabolic transition with the path defined by the programmed target point to the apex of the parabola. The system will enforce this exact path, but will adjust the curve if the input segments are too short. Note that the FMK implementation is logically "ExactCornerDeviation" even though the PLCOpen spec calls for "MaxCornerDeviation"
	10	TMMLXBlend	This mode is only user with MotomanSync remote hosted robots controlled by the YRC1000 series and related controllers. The transition parameter values are 0~8 with 0 defining a motion that stops at the segment end point and 1~8 apply blend distances defined in the robot controller.
	11	TMMaxCornerDeviation	Geometry limited parabolic transition with the path defined by specified corner distance from the start point of the curve to the programmed target point. The blend will begin no sooner than this distance to the programmed target point and can be started at any point closer without generating an error. The system is free to pick any paths that obey this constraint. This will limit the path to a maximum distance which is useful if there are obstacles in the way. It is also useful for applications that can tolerate a variable blend radius depending on the speed, motion sequence, or timing.

Enumerated Type	#INT Value	Enum Value	Description
Y_AdjustMode	0	MasterDistance	The cam adjustment starts immediately, and completes when the master has travelled the specified distance. If MasterDistance is 0.0, then the cam adjustment finishes in the same scan it starts.
	1	ElapsedTime	The cam adjustment starts immediately, and completes within the specified time. If Time=0.0, then the adjustment completes in the same scan it starts.
	2	WithinRange	The cam adjustment starts when the master is crosses the StartPosition, and completes when the master reaches the EndPosition. If the master position is already between StartPosition and EndPosition, then the adjustment starts immediately, but still completes at the EndPosition, which means that the correction speeds may be higher.
Y_ControlMode This enumerated type is for use with Y_DirectControl			
Y_ControlMode	0	NoControlMode	This is not a valid value.
	1	PositionMode	The ServoPack will be controlled in position mode. A new command target must be calculated in the controller at a frequency suitable for the application. No acceleration or speed profiling is added by the system, except for the command filtering provided by pn 811/812, or pn 216/217 if using the MP2600iec.
	2	VelocityTLMode	The ServoPack will be operated in velocity mode. The Torque input of the Y_DirectControl function block will serve as the torque Limit in this mode.
	3	TorqueVLMode	The ServoPack will be operated in torque mode. The Velocity input of the Y_DirectControl function block will serve as the velocity limit in this mode. If the velocity limit is reached, the servo will apply increased torque if necessary to keep the velocity within the limits.
	4	PositionTrqFFMode	The ServoPack will be operated in position mode. The Torque input of the Y_DirectControl function block will serve as the torque feed forward value in this mode.
	5	VelocityTrqFFMode	The ServoPack will be operated in velocity mode. The Torque input of the Y_DirectControl function block will serve as the torque feed forward value in this mode.
Y_TableType			
Y_TableType	0	Undefined	
	1	MasterSlaveArray	This is the only TableType supported.



Getting Started with Cam Toolbox

The Cam Toolbox contains function blocks that combine PLCopen blocks like Y_CamIn, Y_CamShift, Y_SlaveOffset, Y_CamSlave, Y_ReleaseCamTable etc. These toolbox function blocks provide enhanced application level functionality that can be used for cam applications like random rotary knife, linear flying shear, labeler, bottle filler etc.

Requirements for v374

To use the Cam Toolbox, the main project must also contain the following:

Firmware libraries:

- YMotion (only if using [CamSlave_FeedToLength2](#))

User libraries:

The following User Libraries must be listed above the Cam Toolbox and in the following order:

- Math_Toolbox (v300 or higher)
- DataTypes_Toolbox (v300 or higher)
- PLCopen_Toolbox (v300 or higher)

Using the Cam Toolbox

Cam Toolbox contains functions which provide enhanced support for the Y_Cam* PLCopen function blocks.

See Yaskawa's Youtube video - [Camming Demonstration with Yaskawa MP2300Siec](#) for more info.



Cam Toolbox Revision History

Current Version:

2022-06-23 v374 released

- 1) Labeler FB - Remove LabelerAxisType VAR_INPUT. DCR 6623.
- 2) Labeler FB - Add Busy VAR_OUTPUT. DCR 6624.
- 3) CamMaster_Lookup FB - Improved to support situations when slave goes out & back. DCR 6651.
- 4) CamMaster_Lookup / CamSlave_Lookup FB - Changed to Enable /Valid style behavior. DCR 6686.
- 5) MasterIndex_Lookup / SlaveIndex_Lookup FB - Changed to Enable / Valid style behavior. DCR 6695.

Previous Versions:

Note, there was no v373 release.

2021-07-07 v372 released

- 1) Labeler FB - New addition to the toolbox. DCR 4456.
- 2) CamShiftControl FB - Was not reporting all possible internal error conditions. DCR 4992.
- 3) SetCamMasterCycle FB - If an internal function wasn't allowed to finish (CommandAborted go low), then it would cause errors with other camming functions. DCR 3371.
- 4) SetCamMasterCycle FB - Did not 'wake up' parameter 1502. DCR 5332.

2020-02-06 v370 released (There were no changes when the December 2020 Toolbox installer was released, therefore there is no 371 release.)

- 1) CamGenerator - If an all cubic cam was generated more than once, spline coefficients were incorrect in the subsequent calculations because of improper initialization. DCR 3782.

2019-02-06 v352 released

- 1) CamGenerator.TableSize code change in v350 was wrong, caused false Error 10041. DCR 2057.
- 2) CamGenerator FB - Cubic spline calculations limited by the segment CamSegmentStruct index number. Cubic Splines no longer limited by their placement within the overall CamSegmentStruct. DCR 1740.

2018-06-26 v350 released

- 1) CAM_Analyzer FB - Did not output correct values for MaximumAcceleration and MaximumJerk. DCR 1266.
- 2) CamTableManager FB - Fixed bug introduced in last version when making the number of tables configurable. DCR 1629.
- 3) CamGenerator - Deprecated the use of the TableSize VAR_INPUT, it is no longer referenced in code in favor of using the UPPER_BOUND function to determine the size of the CamSegment structure. Also removed other hard coded size checks used for the Spline formula Type. DCR 1969.

2017-12-06 v340 released

- 1) CamTableManager - Added VAR_INPUT to specify the number of cam tables to keep in memory before removing. DCR 1104 .
- 2) CamGenerator - Improved reporting of a resolution ErrorID 10039. DCR 1171.

2016-09-18 v302 released

- 1) CamControl FB - Changed CamControl to set ControlData.Shifting based on CamShift.Done instead of Busy. DCR 780.
- 2) Added EngageWindow to CamSynchStruct. This will help to map user values into Y_CamIn for features like the new "Labeler" Function Block. DCR 782.

2015-01-31 v301 released

- 1) CamGenerator - DCR 766, fixed Tangent Blending to blend with non-zero slave starting position. Change also added for Cam Editor in MotionWorks IEC v3.1
- 2) CamShift_Control - Changed line 215 to ControlData.Shifting:=Y_CamShift_1.Busy OR Y_CamShift_1.Done. DCR 682.

2015-01-31 v300 created

- 1) Identical to v206, but recompiled specifically for MotionWorks IEC v3.x.

2014-11-14 v206 released. Developed using firmware 2.6

2015-01-31 v300 released. Identical to v206, but recompiled specifically for MotionWorks IEC v3.x.

- 1) CalcBezier - Improved code.
- 2) SlaveOffset_Control - Changed equation for first correction. Added manual offset input for adjusting while in motion.
- 3) CamGenerator - ParabolicVelocityBlend formula - Code added for blending improvement (Lines (132- 138)
- 4) CamTableUpdate - Changes made to prevent outputs from flickering (Refer to DCR 467)
- 5) CamControl - Changes made to iActive code to prevent flickering outputs in case of an Error and also lockup of function if data in CamControl structure changes after function block goes dormant.
- 6) SlaveRegistrationCheck - Added (DefaultSize * LREAL 1.1) to NextCheckPoint calculation on rising edge of Enable (line 27)
- 7) SetCamMasterCycle - Improvement by adding Y_CamShift with zero phaseshift to keep prm 1502 updating after this block executes.

2014-03-07 v205 released. Developed using firmware 2.5

- 1) CamBlend - Changed EngageData.SlaveAbsolute to FALSE. This is to support changing master cycles without changing the position scale in the Hardware Configuration.
- 2) CamBlend - Removed NC CamOutBusy from rung 11. This bit could cause the RampOut bit to fire twice.
- 3) CamBlend - Changes made to improve RampIn to RampOut transition without entering Running mode. Added NC RampOutBusy on rung 4. Added ActiveTable_RampIn in rung 11. Added ActiveTable_RampOut on Rung 4 to allow RampOut to RampIn transition in consecutive cycles. Added NC RampOutBusy and NC RampInBusy on rung 19 to prevent iActive from turning off prematurely.
- 4) CamBlend - Removed CamOut FB from CamBlend. Non periodic RampOut is sufficient. This eliminates false error outputs.
- 5) CamShift_Control - Added new datatype 'SynchPosition'. This is the position in the cam table where the master and slave become synchronized.
- 6) CamShift_Control - Simplified the equation for correction for initial shift for modes 1 and 2.
- 7) SlaveOffset_Control - New FB, similar to CamShift_Control. Buffered offsets on a slave axis can be accomplished by buffering registration marks. New datatype SlaveOffsetStruct accompanies SlaveOffset_Control FB.
- 8) SetCamMasterCycle - New FB in this version. Sets the cam master cycle the first time to change it from default of 1.0 to the Master cycle of the cam table to be used. Only necessary for applications that use Y_CamShift before engaging the cam.
- 9) CamGenerator - CalcSpline formula completely re written with new algorithm.

- 10) CamGenerator - New Bezier curve added. Bezier segment requires straight segments before and after the Bezier curve. This is a modified bezier which will never cause reverse motion.
- 11) CamSlaveFeedToLength - Further improvement based on customer feedback for the change made in v204. TestTrack DCR 7. SlaveRegistrationCheck is completely shut down if no cam is active, this prevents MissedLatchError from occurring.
- 12) CamControl, CamShift_Control - Added support for Multi Use Latches, which is a new feature of the PLCOpen Toolbox v206 ProductBuffer function block.

Starting in Cam Toolbox v204 – All firmware library DataType definitions were moved to a new toolbox called the DataTypes Toolbox. Formerly, the PLCOpen Toolbox contained the MotionInfoTypes and the PLCTaskInfoTypes datatype files. These were removed and are now included in the DataTypes Toolbox. If upgrading from an older version of Cam Toolbox, you must do the following:

- 1) Include the DataTypes Toolbox in the main project.
- 2) Remove any other Yaskawa supplied datatype files with firmware library definitions such as:
 - a. ControllInfoTypes
 - b. YDeviceCommTypes

Note: Compiler issues yielding the message "Error during generating native code" will be experienced under the following combined conditions: 1) Using Cam Toolbox v204 or higher AND using an MP3000iec series controller AND using MotionWorks IEC v2.x. The remedy is to downgrade to Cam Toolbox v203 or use MotionWorks IEC v3.x.

2013-09-01 v204 released. Developed using 2.4.0 firmware

- 1) CamBlend - Added ErrorID 10084. One of the Cam Tables has an invalid TableID.
- 2) CamBlend - Fixed ExecuteStandStill contact in RETURN rung to be normally closed.
- 3) CamGenerator - Corrected mistake with Tangent Match & Tangent Blend formulas introduced in v202 when CamGenerator was improved to allow blending segments.
- 4) CamBlend - Added check: If BlendData.Window = 0, then the code defaults the value to 1% of the CamMasterCycle.
- 5) CamGenerator - Added curve type 32 for Arc profile. Also added radius and direction to CamSegmentStruct
- 6) Removed references to Math Toolbox functions where possible. Now only the CamShiftControl function block requires the Math Toolbox.
- 7) Because of the reintroduction of functions with EN/ENO, the MP2600 requires firmware 2.1.
- 8) SlaveRegistrationCheck - Added ErrorID 10086 to report if the MaxPosCorrection or MaxNegCorrection are not set correctly.
- 9) CamSlaveFeedToLength - Added RecordedPosition as output. Also included interlock to prevent adjustments from occurring if the slave is not engaged.
- 10) CamGenerator - Added Parabolic with blended velocity as formula code 33. (for multi segment)
- 11) CamShift_Control - Consolidated Rotary Knife and Linear Flying shear math.

2013-01-16 v203 released. Created using 2.4.0 firmware

- 1) CamGenerator - Improved to support wrap around cubic spline segments at the beginning and the end of the cam. (YEU) 7 spline categories tested.
- 2) CamGenerator - Added TableShift support into the CamSegmentStruct. Initial shifts can be applied to the cam data without using the Y_CamShift function block.

2012-11-19 v203 created using 2.3.0 firmware

- 1) CamGenerator - Improved support for wrap around cubic spline segments at the beginning and the end of the cam. (YEU) 7 spline categories tested.

2) CamGenerator - Added TableShift support into the CamSegmentStruct for CamGenerator. Initial shifts can be applied to the data

without using the Y_CamShift function block.

2012-10-18 v202 released. Created using 2.2.0 firmware

1) CamGenerator - Improved to allow blending segments such as straight line, parabolic, modified sine without forcing a zero speed transition.

2) CamGenerator - Improved for blending of Cubic Spline segments to other segment types.

3) SlaveRegistrationCheck - Changed 'Missed Latch Error' to occur when the missed latch counter is \geq the MissedLatchLimit. Previously it was not causing error until the MissedLatchLimit was exceeded.

4) CamBlend - Added DisengageData to CamBlend's Y_CamOut for compatibility on MP2600iec and MP3200iec

2011-03-09 v201 released. Created using 2.1.0 firmware

1) CamGenerator - Added Cubic Spline CurveType as Type #31.

2) CamAnalyzer - Added new function block.

3) CamFileMgmt - CamTableMgmt renamed CamTableManager.

4) CamSlave_Lookup - Fixed false 10113 ErrorID from occurring.

5) CamSlave_Recover - Fixed unconnected line in the first rung.

6) DataTypes - Increased CamPair and CamSegmentArray from 200 to 400.

2011-07-29 v200 released. Created using 2.0.0 firmware

1) Built from v009beta for MotionWorks IEC 2.0

2011-04-02 v009 released. Created using 1.2.4 firmware

1) Added CamSlave_Lookup and CamSlave_Recover function blocks for e-stop recovery capability.

2) Added input 'ExecuteStandstill' to CamBlend. This input causes the running cam to engage immediately, which enhances the E-Stop recovery capability of CamBlend.

3) Removed SETCOIL from CamBlend CommandAborted.

2011-04-01 v008 released. Created using 1.2.4 firmware

1) Fixed Y_CamStructSelect in PathGenerator to comply with PLCopen rule to read TableID only on the scan.
when done is high. (Also to comply with firmware change made for 1.2.3.)

2) Reworked PathGenerator to support any variety of arcs beyond just simple 0,90,180,270 quadrants.

3) Removed spaces from project file name for improved usage with MotionWorks IEC 2.0.

4) Removed PathGenerator and MovePath, ported over to Gantry Toolbox.

5) Included YMotion firmware library in ZWT, required for CamSlaveFeedToLength2 function block.

NOTE: This toolbox will work with 1.2.3 firmware unless CamSlaveFeedToLength2 is used, which requires firmware 1.2.4.

6) Improved CamBlend's CommandAborted output behavior to ignore Commandaborted caused by itself.

2011-02-02 v007 released

1) Fixed incorrect parameter in CamBlend for checking the half way point of the cam cycle.

Step 5 had 1520, it is changed to 1512. Also streamlined the code to only include one check for Halfway instead of two.

2) Added CamSlaveFeedToLength2, which incorporates Y_ProbeContinuous from the Y_Motion firmware library and requires firmware 1.2.4 or higher. NOTE: After the 2.0 product release, Y_ProbeContinuous will be available in PLCopenPlus firmware library v2_3.

2010-11-15 v006 released

Moved on to v006, beta005 never released.

1) Increased flexibility of CamSlave_FeedToLength / SlaveRegistrationCheck by making Max Positive and Negative Correction inputs and outputs.

2) Added CamShift_Control FB for 'Rotary' and 'Out and Back' cam motions.

3) Added TB_CurveType#Polynomial345 to CamGenerator, Polynomial345.

4) Added Cam_Control FB which works with the Product Buffer for slaves that must stop when no product is coming.

2010-08-01 v005beta created)

Moved on to v005, beta004 never released.

1) Merged code changes with Doug Meyer, for CamSlavePullToLength and CamSlaveFeedToLength for MaxCorrection and Time based correction. NOTE: Function block interface changed for these functions.

2) Removed LatchError from occurring in CamSlavePullToLength and CamSlaveFeedToLength.

3) Moved window logic into the main Enable section of SlaveRegistrationCheck to allow on the fly updates.

2010-07-02 v004beta created

Moved on to v004, beta003 never released.

1) Added logic to SlaveRegistrationCheck to add one CamCycle if the LatchTableReference is negative.

2010-03-15 v003beta created

1) Fixed mistake in case statement to allow Simple Harmonic as one of the Valid Curve Types. Was 4, should be 3.

2) Changed Max CamSegmentArray size to 200 from 20.

3) Changed CamSlave_FeedToLength to use Stair Step method of latch lookup in cam table. Original method used an

interpolated latch algorithm.

- 4) Removed Y_EngageMethod#Linked as a StartMode inside CamBlend.
- 5) Changed the second and third Y_CamIn functions inside CamBlend to use StartMode = Absolute to eliminate drifting caused by switching tables while master in motion.
- 6) Added NOT(Error) contact to prevent the CamSlave_FeedToLength function from running if there was an error.
- 7) Added PathGenerator and MovePath for creating XY paths with straight line and circular interpolation.
- 8) Added CamSlavePullToLength and supporting function CS_PTL_ScaleCalc.

2010-03-12 v002 released

- 1) Changed CamGenerator straight line segment to include option for calculating points at spec'ed resolution.
- 2) Initial version would ignore resolution and just use beginning and end points for straight line.
- 3) Improved CamGenerator. It was recalculating the entire profile over and over each scan while execute was held high. Changed to F_TRIG to let initialize section run on the first scan, and the cam calcs on the second.
- 4) Improved CamBlend Output behavior. (Some bits remained on when both execute inputs were off.

2010-02-01 v001beta created

Created Cam Toolbox by moving the following Function blocks from PLCopen Toolbox v019beta:

- 1) CamBlend
- 2) CamMaster_Lookup
- 3) CamSlave_FeedToLength
- 4) CamSlave_WindowCheck
- 5) CamGenerator
- 6) CamTableUpdate
- 7) SlaveRegistrationCheck
- 8) SlaveIndex_Lookup



Data Type: BlendStruct

Used by the [CamBlend](#) function block

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyBlendStruct	BlendStruct		
U	RampInTableID	UINT	The TableID of the Cam profile which accelerates the slave to synchronize with the master.	MyBlendStruct.RampInTableID
U	RampInSwitchOverPos	LREAL	A position where the slave has the same position in both the RampIn and Running table, typically near the last 90 to 100% of the profile.	MyBlendStruct.RampInSwitchOverPos
U	RunningTableID	UINT	The TableID of the Cam profile is used in normal operation.	MyBlendStruct.TableID
U	StandStillEngagePos	LREAL	This input can be used if the slave is being engaged to the master at stand-still. (E-Stop recovery where the slave engages to a stationary master). This input will engage the slave to the running table.	MyBlendStruct.StandStillEngagePos
U	RampOutTableID	UINT	TableID of the Cam profile which decelerates the slave to a stop at a desciied location.	MyBlendStruct.RampOutTableID
U	RampOutSwitchOverPos	LREAL	Specify a position where the slave would be at the same position in both the RampIn and Running table, typically near the last 90 to 100% of the profile.	MyBlendStruct.RampOutSwitchOverPos
U	Window	LREAL	Switchover / Engage window in master units. The cam will engage at any master position from EngagePosition +/- EngageWindow. Units are those of the cam master.	MyBlendStruct.Window



Data Type: CamPairs

Used by the [CamGenerator](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
MyCamPairs		CamPairs		
U	CamPairs	ARRAY[0..400] OF UDINT		MyCamPairs[0]



Data Type: CamParameters

Supporting structure for [CamSegmentStruct](#). For use with the [CamGenerator](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCamParameters	CamParameters		
U	MasterEnd	LREAL	Position of the master at the end of the current segment.	MyCamSegmentStruct.MyCamParameters[x].MasterEnd
U	SlaveEnd	LREAL	Position of the slave at the end of the current segment.	MyCamSegmentStruct.MyCamParameters[x].SlaveEnd
U	CurveType	INT	Formula code to indicate the motion profile for the segment.	MyCamSegmentStruct.MyCamParameters[x].CurveType
U	Resolution	REAL	Determines how many data points are calculated along this segment. If the master delta for this segment is 10 units, and the resolution is 0.5, then 20 points calculated. For CurveType=StraightLine, there is no need to specify a resolution for most applications. Set the Resolution value to zero to conserve on calculated cam data points.	MyCamSegmentStruct.MyCamParameters[x].Resolution



Data Type: CamSegmentArray

Supporting structure for [CamSegmentStruct](#). For use with the [CamGenerator](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCamSegmentArray	CamSegmentArray		
U	CamSegmentArray	ARRAY[0..400] OF CamParameters		MyCamSegmentArray[0]

Notes:

This is an internal sub structure for CamSegmentStruct and is not intended to be referenced directly by the user.

Data Type: CamSegmentStruct



For use with the [CamGenerator](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCamSegmentStruct	CamSegmentStruct		

*	Element	Data Type	Description	Usage
U	CamParameters	CamSegmentArray		

*	Element	Data Type	Description	Usage
U	MasterEnd	LREAL	Location of the master at the end of the current segment	MyCamSegmentStruct.CamParameters [x].MasterEnd
U	SlaveEnd	LREAL	Location of the slave at the end of the current segment	MyCamSegmentStruct.CamParameters [x].SlaveEnd
U	CurveType	INT	Formula code to indicate the motion profile for this segment	MyCamSegmentStruct.CamParameters [x].CurveType
U	Resolution	REAL	Determines how many data points are calculated along this segment	MyCamSegmentStruct.CamParameters [x].Resolution
U	ArcRadius	LREAL	If CurveType = Arc, this element describes the radius. Not used for any other CurveTypes.	MyCamSegmentStruct.ArcRadius
U	ArcDirection	INT	If CurveType = Arc, (1=ccw, -1=cw). Not used for any other CurveTypes.	MyCamSegmentStruct.ArcDirection
U	SplineStartSlope	LREAL	If the first Segment is CurveType = CubicSpline, this value is the positional slope of the profile as the cam begins. Typically this value is zero unless the cam is blended with other cams, and the slope cannot be automatically determined by the CamGenerator.	MyCamSegmentStruct.SplineStartSlope
U	SplineEndSlope	LREAL	If the last Segment is CurveType = CubicSpline, this value is the positional slope of the profile as the cam completes. Typically this is zero unless the cam is blended with other cams, and the slope cannot be automatically determined by the CamGenerator.	MyCamSegmentStruct.SplineEndSlope
U	TableShift	LREAL	If non zero, this value represents the amount of initial shift in the master slave values that is applied to the table data.	MyCamSegmentStruct.TableShift

*	Element	Data Type	Description	Usage
U	LastSegment	INT	Informs the CamGenerator which element of the CamSegmentStruct contains the last segment of cam data to be applied.	MyCamSegmentStruct.LastSegment
C	OutAndBackCam	BOOL	Flag which indicates if the first and last slave position are the same (out and back.) If they are different, the cam is reciprocating, and the slave will move away from the initial start position with each passing cycle.	MyCamSegmentStruct.OutAndBackCam
U	UseSplineSlope	BOOL	Flag to indicate to use the SplineSlope parameters in this structure.	MyCamSegmentStruct.UseSplineSlope

Example

RampInCam.SlaveStart:=LREAL#0.5; (* Slave home position at 12 O'Clock *)

RampInCam.LastSegment:=INT#2;

RampInCam.CamParameters[1].CurveType:=TB_CurveType#TangentBlending;

RampInCam.CamParameters[1].MasterEnd:=LREAL#0.9;

RampInCam.CamParameters[1].SlaveEnd:=LREAL#0.9; (* Slave moves SlaveEnd - SlaveStart during RampIn *)

RampInCam.CamParameters[1].Resolution:=REAL#0.01;

RampInCam.CamParameters[2].CurveType:=TB_CurveType#StraightLine;

RampInCam.CamParameters[2].MasterEnd:=LREAL#1.0;

RampInCam.CamParameters[2].SlaveEnd:=LREAL#1.0;

RampInCam.CamParameters[2].Resolution:=REAL#0.01;



Data Type: CamStruct

For use with Y_CamIn and Y_CamOut function blocks

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCamStruct	CamStruct		
U	FileName	STRING	Filename that will be used by Y_CamFileSelect	MyCamStruct.FileName
U	TableType	INT	0=Undefined, 1=M/S pair, 2=reserved, 3=reserved	MyCamStruct.TableType
U	TableSize	UDINT	The size of the cam table in bytes (Don't forget, 16 bytes per M/S pair)	MyCamStruct.TableSize
U	TableID	UINT	Number returned from Y_CamFileSelect	MyCamStruct.TableID
U	EngagePosition	LREAL	Master location where slave must start synchronization (Reference prm 1502 - CamMasterShiftedCyclic)	MyCamStruct.EngagePosition
U	EngageData	Y_ENGAGE_DATA		MyCamStruct.
U	DisengagePosition	LREAL	Master location where slave must discontinue synchronization. (Reference prm 1502 - CamMasterShiftedCyclic)	MyCamStruct.DisengagePosition
U	DisengageData	Y_DISENGAGE_DATA		MyCamStruct.
U	Window	LREAL	Size of the window in master units where the engage or disengage event must take place.	MyCamStruct.Window
U	MasterCycle	LREAL	Defines the total Master axis travel over which the slave synchronizes with the master. As the master goes beyond the MasterCycle, the synchronization profile repeats.	MyCamStruct.MasterCycle
U	SlaveCycle	LREAL	Slave positions which correspond to the Master at prescribed intervals along the Cam profile.	MyCamStruct.SlaveCycle



Data Type: CamSyncStruct

For use with the [CamControl](#) , [CamShift_Control](#) and [Labeler](#) function blocks.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCamSyncStruct	CamSyncStruct		
U	Mode	INT	Describes the application so the function blocks can apply the correct logic. 1 = Rotary Knife 2 = Linear Flying Shear 3 = Rotary Placer or Reciprocating Drill	MyCamSyncStruct.Mode
U	StartSyncPosition	LREAL	The first master position where the slave must be synchronized with the master.	MyCamSyncStruct.StartSyncPosition
U	EndSyncPosition	LREAL	The final master position where the slave must be synchronized with the master.	MyCamSyncStruct.EndSyncPosition
U	DecisionPosition	LREAL	Key location in the process where the controller must decide to disengage the slave from the process or continue camming and CamShift to the next product.	MyCamSyncStruct.DecisionPosition
U	MaxShift	LREAL	If Mode = 3, this value helps the CamShift_Control function block determine whether the slave should advance or retard to synchronize with the next product. For other modes, this input is not used.	MyCamSyncStruct.MaxShift
U/C	SafeEngageDistance	LREAL	The distance the master travels from the sensor until the product is less than one machine cycle away from the synchronization position. If zero is entered, the CamShift_Control function block will calculate the value automatically. If the MachineCycle is greater than the distance from the sensor to the synchronization point, enter LREAL#0.0.	MyCamSyncStruct.SafeEngageDistance
C	Shifting	BOOL	Status flag set by the CamShift_Control function block to signal the CamControl function block.	MyCamSyncStruct.Shifting
C	Pause	BOOL	Status flag set by the CamControl function block to signal the CamShift_Control function block.	MyCamSyncStruct.Pause



Data Type: LabelStruct

For use with the [Labeler](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	Label	LabelStruct		
U	TriggerInput	MC_TRIGGER_REF	Reference to the trigger signal source. This structure identifies which latch signal on the hardware will be latched.	Label.TriggerInput.ID
U	LabelSize	LREAL	Label pitch. Includes label length and label gap. This is also the slave cycle in the cam profile.	Label.LabelSize
U	DistanceAfterlatch	LREAL	The desired additional travel distance after the registration mark is detected.	Label.EndSyncPosition
U	MaxNegCorrection	LREAL	The most negative correction that can be applied per label, in user units of the label axis.	Label.MaxNegCorrection
U	MaxPosCorrection	LREAL	The most positive correction that can be applied per label, in user units of the label axis.	Label.MaxPosCorrection
U	SensorMaximum	LREAL	The latest position in the label index where the registration mark is expected under normal conditions, in label axis units.	Label.SensorMaximum
U	SensorMinimum	LREAL	The earliest position in the label index where the registration mark is expected under normal conditions, in label axis units.	Label.SensorMinimum
U	StartCorrection	LREAL	The start of the area on the label where it is OK to apply the micro correction to adjust the label placement, in label axis units.	Label.StartCorrection
U	EndCorrection	LREAL	The end of the area on the label where it is OK to apply the micro correction to adjust the label placement, in label axis units.	Label.EndCorrection
U	MissedLatchLimit	UINT	The # of consecutive default distances allowed to occur without seeing a registration mark and not cause an error.	Label.MissedLatchLimit
U	LatchData	CONTINUOUS_REF	The structure containing data for configuring and operating continuous latch mode.	Label.BufferSize



Data Type: Matrix

For internal use by the [CamGenerator](#) for [Cubic Spline](#) calculations.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyMatrix	Matrix		
U	Matrix	ARRAY[0..20] OF SubMatrix		MyMatrix[0]



Data Type: SlaveOffsetStruct

For use with the [SlaveOffset_Control](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MySlaveOffsetStruct	SlaveOffsetStruct		
U	StartSyncPosition	LREAL	The first master position where the slave must be synchronized with the master	MySlaveOffsetStruct.StartSyncPosition
U	SyncPosition	LREAL	The master position that represents the center of the sync zone. Usually SyncPosition = (EndSyncPosition - StartSyncPosition)/2.	MySlaveOffsetStruct.SyncPosition
U	EndSyncPosition	LREAL	The final master position where the slave must be synchronized with the master, adjustments can start after.	MySlaveOffsetStruct.EndSyncPosition



Data Type: TableIDStruct

For use with the [CamTableUpdate](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyTableIDStruct	TableIDStruct		
U	Inactive	UINT	The CamTableID that is NOT currently being accessed to control motion.	MyTableIDStruct.Inactive
U	Active	UINT	The CamTableID that IS currently being accessed to control motion.	MyTableIDStruct.Active



Data Type: UINTArray

For use with the [CamTableManager](#) Function Block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyUINTArray	UINTArray		
U	UINTArray	ARRAY[0..4] OF UINT	An array for CamTableIDs that are released from memory in a FIFO method.	MyUINTArray [0]



Data Type: Y_MS_CAM_STRUCT

This data type is for use with the Y_CamStructSelect, Y_ReadCamTable, and Y_WriteCamTable function blocks. Y_MS_CAM_STRUCT consists of the sub-structures found below. Refer to the Internally Created Cam Data diagram in the Cam Data Management section.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCam	Y_MS_CAM_STRUCT		
	Header	Y_CAM_HEADER		
U	TableType	INT	INT#1 = Master/Slave pair. If using the Y_ReadCamTable function block, this value must be set by the user before executing the function.	MyCam.Header.TableType
	Reserved1	UINT	---	---
U	DataSize	UDINT	Total used size of MS_Data in bytes. (Each Y_MS_PAIR is 16 bytes.)	MyCam.Header.DataSize
	MS_Header	Y_MS_HEADER		
U	SlaveIncremental	BOOL	If TRUE, the slave data from pair to pair is relative.	MyCam.MS_Header.SlaveIncremental
U	MasterIncremental	BOOL	If TRUE, the master data from pair to pair is relative.	MyCam.MS_Header.MasterIncremental
	Reserved1	UINT	---	---
	Reserved2	UINT	---	---
	Reserved3	INT	---	---
	MS_Data	MS_Array_Type	Array of all master / slave data pairs used for the cam. ARRAY [0..2880] OF Y_MS_PAIR	
U/C	Master	LREAL	Master position	MyCam.MS_Data[0].Master
U/C	Slave	LREAL	Slave position	MyCam.MS_Data[0].Slave

Notes

MS_Data[x].Master and MS_Data[x].Slave can be set by either the user or a function block depending on whether this data type is used with Y_ReadCamTable or Y_WriteCamTable in the PLCopen Plus firmware library.

Code Example



Enumerated Types for Cam Toolbox

Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to a zero-based integer (INT) list.

Enumerated Types Declaration

Enumerated Type	#INT Value	Enum Value	Description
TB_Mode	ENUM Type for CamShift_Control to specify the application type.		
	0	n/a	
	1	RotaryKnife	Rotary Knife, Rotary Punch, etc.
	2	LinearFlyingShear	Out and Back, like linear flying shear, walking beam, bottle filler
	3	RotaryPlacer	

Enumerated Type	#INT Value	Enum Value	Description
TB_CurveType	Indicates the Cam formula to be applied between to positions.		

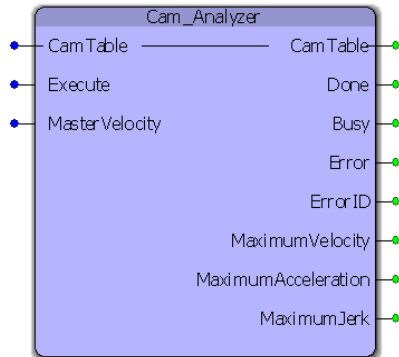
Enumerated Type	#INT Value	Enum Value	Description
	0	n/a	Not a valid CurveType

Enumerated Type	#INT Value	Enum Value	Description
	1	StraightLine	

Enumerated Type	#INT Value	Enum Value	Description
	2	Parabolic	
	3	SimpleHarmonic	
	4	Cycloidal	
	5	ModifiedTrapezoid	
	6	ModifiedSine	
	7	ModifiedConstVelocity	
	8	AsymmetricalCycloidal	
	9	AsymmetricalModifiedTrapezoid	
	10	Trapezoid	
	11	OneDwellCycloidal_1	
	12	OneDwellCycloidal_2_3	
	13	OneDwellTrapezoid_1	
	14	OneDwellTrapezoid	
	15	OneDwellTrapezoid_2_3	
	16	OneDwellModifiedSine	
	17	OneDwellTrapezoid	
	18	NoDwellSimpleHarmonic	
	19	NoDwellModifiedTrapezoid	
	20	NoDwellModifiedConstVelocity	
	21	NC2Curve	
	22	TangentMatching	
	23	ReverseTrapezoid	
	24	DoubleHarmonic	
	25	ReverseDoubleHarmonic	
	26	TangentBlending	
	27	Unsupported27	Unsupported
	28	Unsupported28	Unsupported
	29	UserModifiedConstVelocity	User specifies the accel / decel distances
	30	Polynomial345	5th order polynomial with C3 = 10, C4 = -15, C5 = 6
	31	CubicSpline	Cubic spline interpolation
	32	Arc	
	33	ParabolicVelocityBlend	Parabolic curve with velocity blending
	34	Bezier	Non reversing profile between two straight lines.



Cam_Analyzer



The Cam_Analyzer function block provides the slaves maximum velocity, acceleration, deceleration and jerk values for a specific cam profile based on a maximum expected master velocity.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	CamTable	Y_MS_CAM_STRUCT	This structure contains the resulting master/slave information for each data point and can be downloaded to the motion engine using Y_CamStructSelect.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	MasterVelocity	LREAL	Master axis maximum velocity (in master user units/sec.)	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	

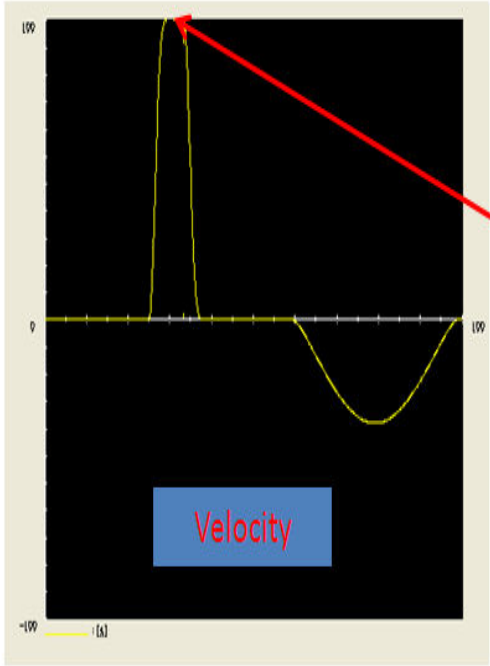
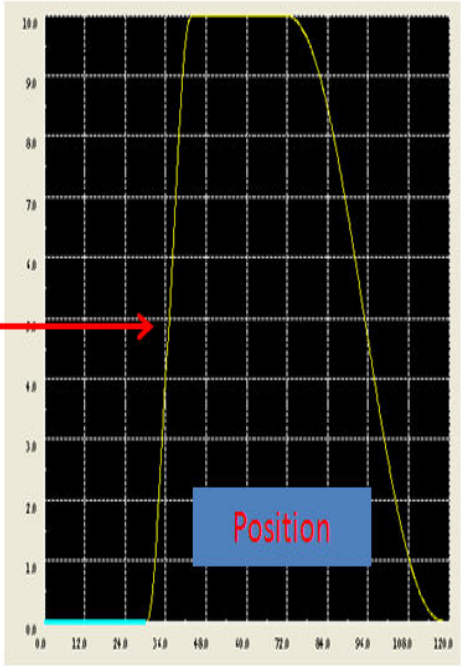
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	MaximumVelocity	LREAL	Peak slave velocity for the given cam profile at the maximum master velocity.
V	MaximumAcceleration	LREAL	Peak slave acceleration for the given cam profile at the maximum master velocity.
V	MaximumJerk	LREAL	Peak slave jerk for the given cam profile at the maximum master velocity.

Error Description

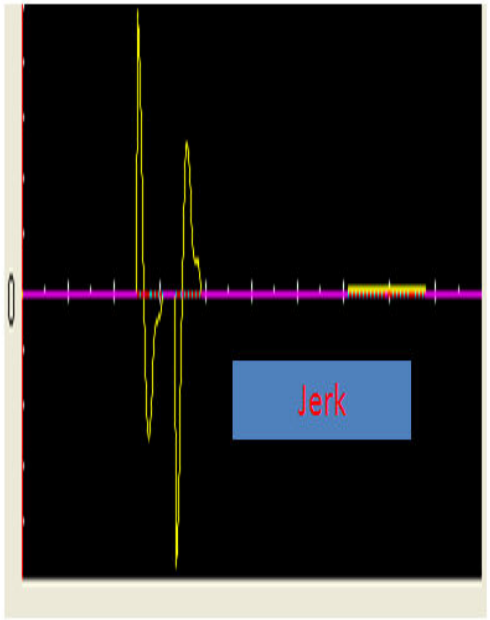
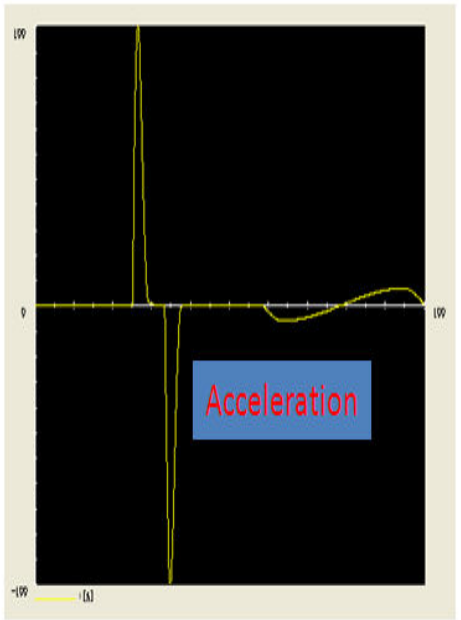
See the [Function Block ErrorID](#) list.

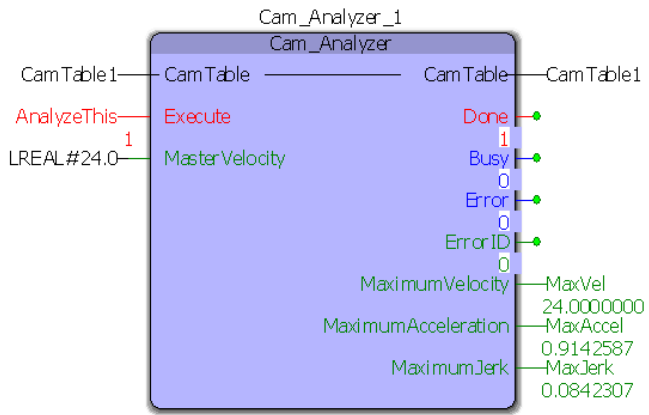
Example

Consider a linear flying shear application. The maximum slave velocity of the profile is in the speed matching region. The master maximum velocity was given as 24 units/sec and the maximum velocity output of the CamAnalyzer is 24.



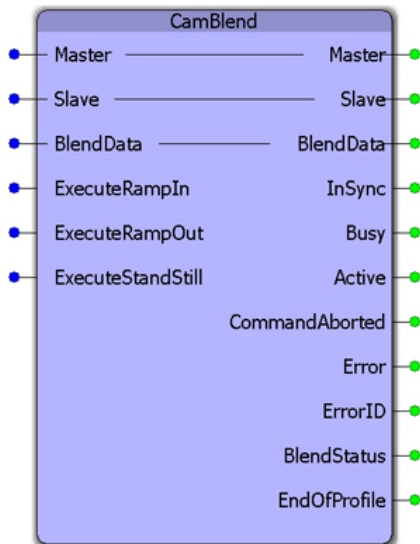
Maximum velocity at speed matching region of cam profile







CamBlend



This function block was designed for applications that require a one way cam profile, and the slave must be able to engage or disengage smoothly from a moving master. It requires three separate cam tables with a portion of equivalent slave data, so an on-the-fly changeover from one table to the next can occur. This function block uses three [Y_CamIn](#) functions blocks and one [Y_CamOut](#) function block.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Master	AXIS_REF	A logical reference to the master axis.	
B	Slave	AXIS_REF	A logical reference to the slave axis.	
V	BlendData	BlendStruct	Structure containing the information required for engaging, disengaging, ramping in, and ramping out. See help for the EngageWindow for additional information.	
VAR_INPUT				Default
V	ExecuteRampIn	BOOL	Upon the rising edge, this function block will prepare to engage the RampIn cam profile at the master position specified in the BlendData structure.	FALSE

V	ExecuteRampOut	BOOL	Upon the rising edge, this function block will prepare to switch to the RampOut cam profile at the SwitchOver position specified in the BlendData structure.	FALSE
V	ExecuteStandStill	BOOL	Upon the rising edge, this function block will prepare to engage the slave to the Running cam profile at the StandstillEngage position (calculated after an E-Stop recovery routine) in the BlendData structure	FALSE
VAR_OUTPUT				
E	InSync	BOOL	Set high when the axis or group is synchronized with the axis or group it is commanded to follow. Synchronized means that the two are position locked, any transitional period required to achieve synchronization has been completed.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Active	BOOL	For buffered modes, this output is set high at the moment the block takes control of the axis. For non buffered modes, the outputs Busy and Active have the same value.	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	BlendStatus	UINT	Outputs a value of 1 to indicate the RampIn Cam is Active, 2 indicates the Running cam is Active, and 3 indicates the RampOut cam is Active. Refer to the slaves CamState parameter [1540] to determine the state the active cam.	
E	EndOfProfile	BOOL	Pulsed output signaling the cyclic end of a CAM Profile	

Notes

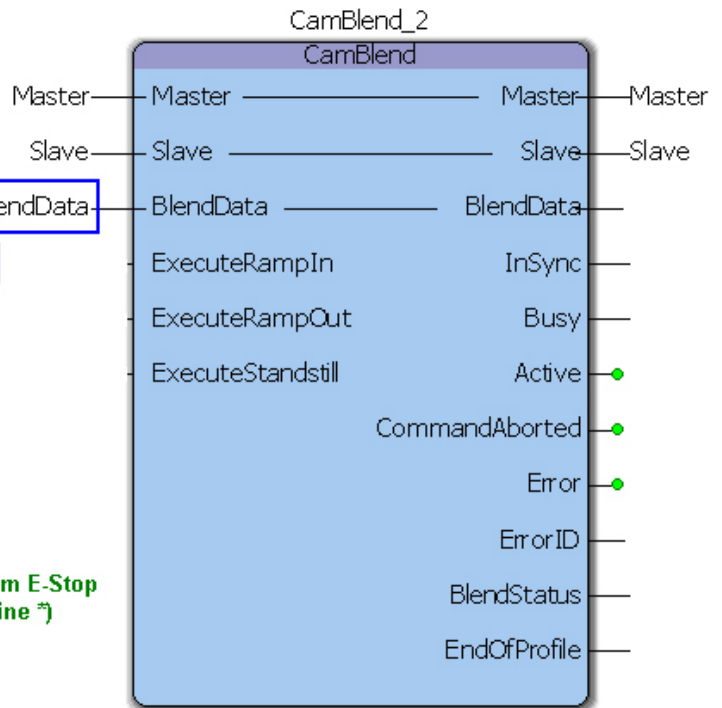
- Typically the RampInSwitchOverPos and the RampOutSwitchOverPos will be fixed at some predetermined position suitable for the application. Typically the RampInSwitchOverPos will occur very late in the cycle, and the RampOutSwitchOverPos will occur very early in the cycle. This will provide for the optimum performance by allowing the maximum possible time for the slave to accelerate up to the master's velocity and decelerate back to zero.
- If using the ExecuteStandStill mode, use the [CamMaster_Lookup](#) and [CamSlave_Recover](#) function blocks to determine the master position that corresponds to the current slave position, and set BlendData.StandStillEngagePos accordingly to preserve synchronization. The ExecuteStandStill mode was added to provide the capability of re-synchronizing after an E-Stop.

See the [CamBlend eLearning Module](#) on Yaskawa's YouTube Channel.

Error Description

See the [Function Block ErrorID](#) list.

Example 1



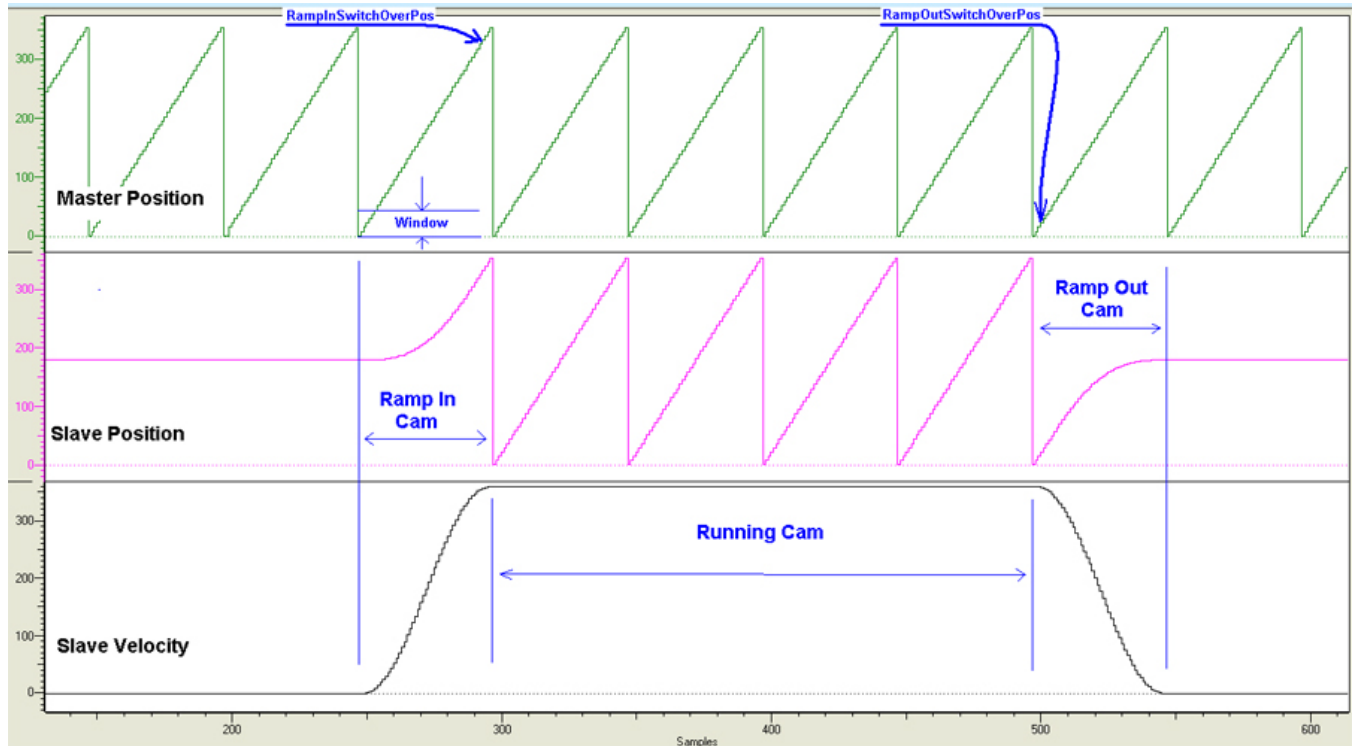
```

CamBlendData.RampInTableID := UINT#1;
CamBlendData.RunningTableID := UINT#2;
CamBlendData.RampOutTableID := UINT#3;

CamBlendData.RampInSwitchOverPos := LREAL#355.0;
CamBlendData.RampOutSwitchOverPos := LREAL#5.0;
CamBlendData.Window := LREAL#3.6;

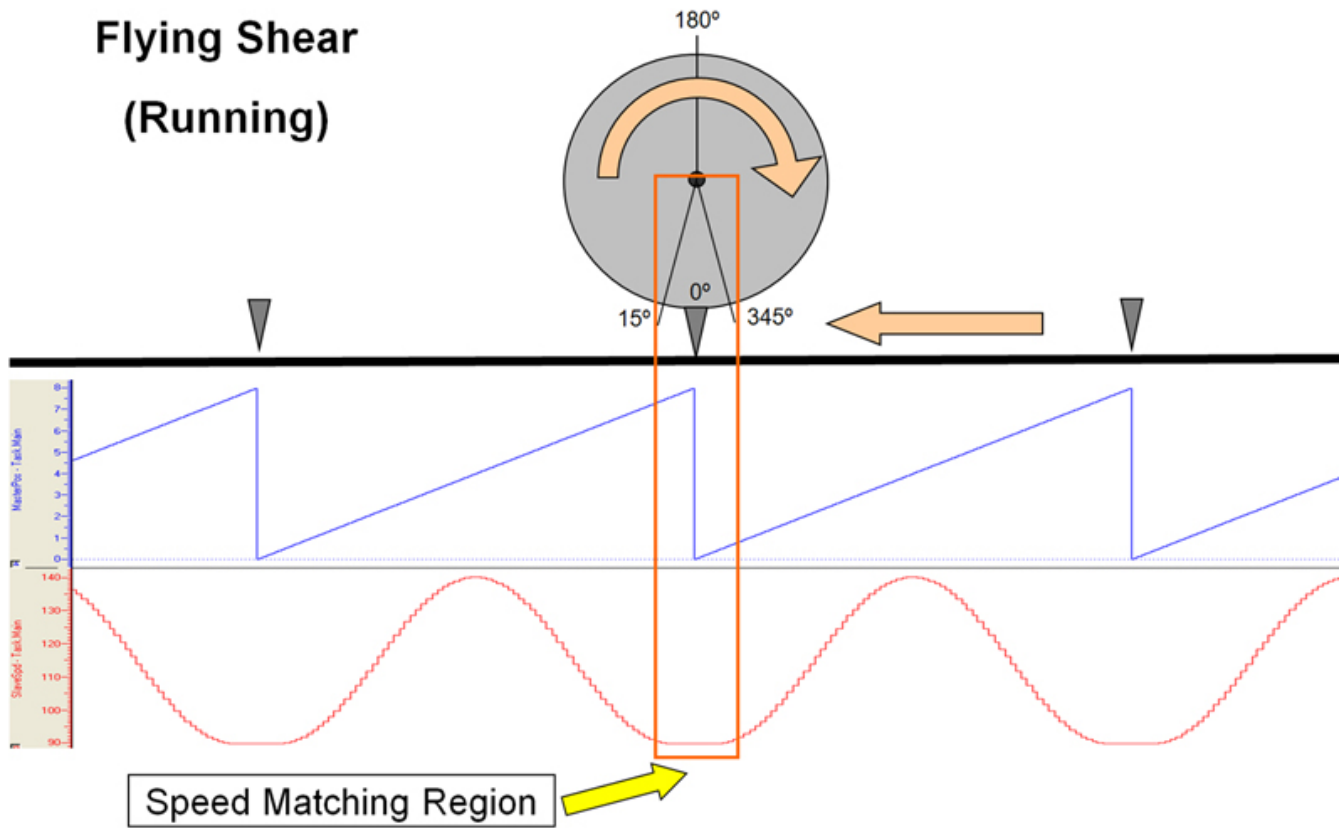
CamBlendData.StandstillEngagePos := (* Calculated from E-Stop
recovery routine *)
    
```

Timing Diagram



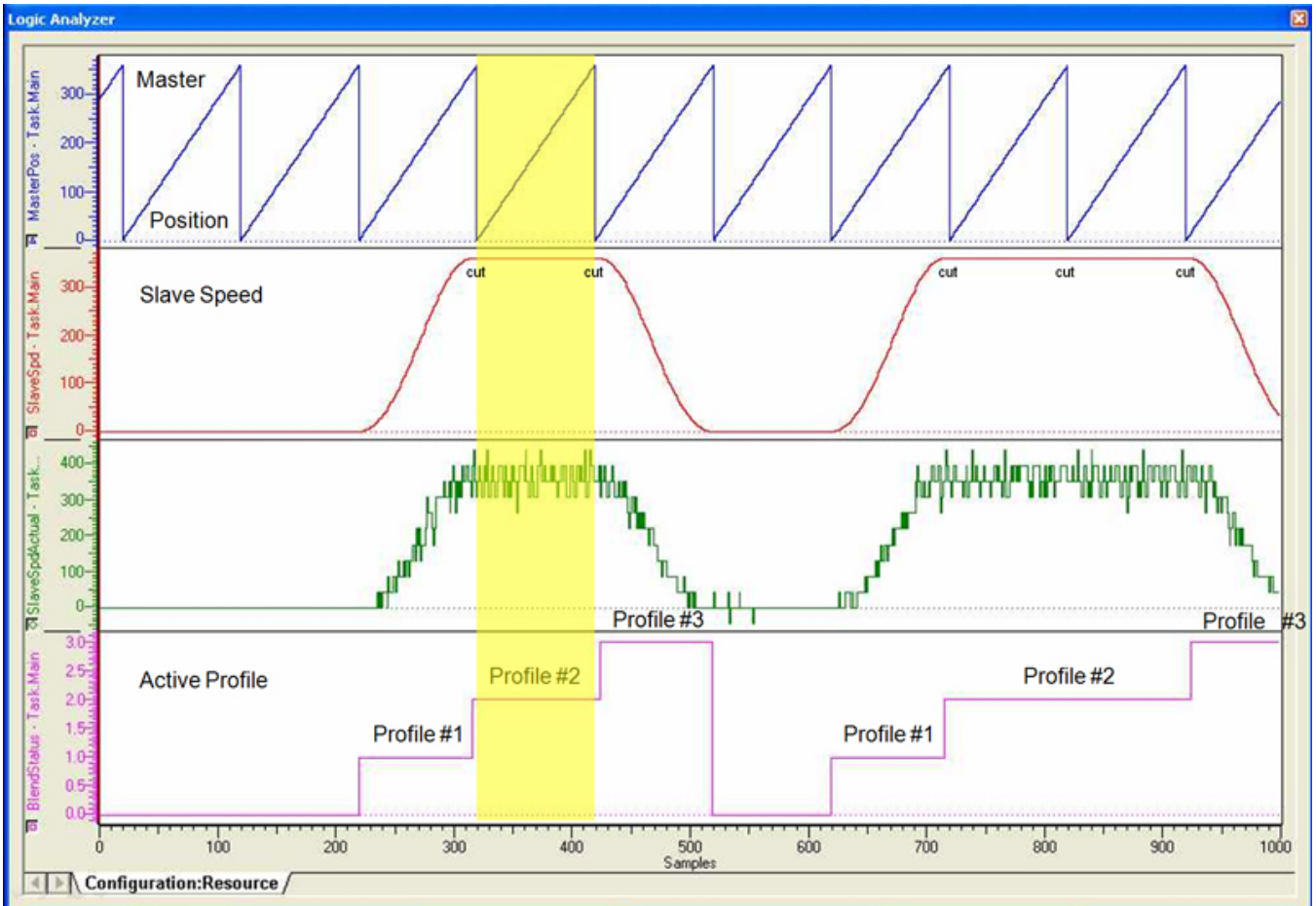
Application Example

Flying Shear (Running)



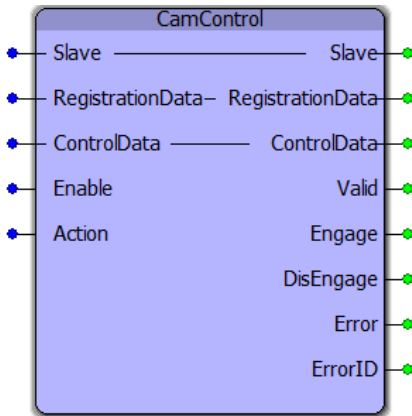
Timing Diagram

The speed matching, or normal running cam is designated as Profile #2. Profile #1 and Profile #3 will only run once, but Profile #2 will run indefinitely. A simple straight line profile for Profile #2 is not required, and reasonable motion can be used if the application requires it, keeping in mind that CamBlend was designed for one way slave motion that never stops while in normal operation, thus making it difficult to synchronize with the master smoothly without blending from one profile to another.





CamControl



The CamControl function makes decisions regarding Engage and Disengage logic for applications where products are buffered and processed at random intervals. This function block requires the [ProductBuffer](#) function block from the PLCopen Toolbox and the [CamShift_Control](#) block from the Cam Toolbox. The main inputs that feed the CamControl block are RegistrationData and ControlData. This function block was designed for applications such as a Linear Flying Shear, Random Rotary Placer, Knife, Drill, etc.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
V	RegistrationData	ProductBufferStruct	Structure containing all information for the circular buffer to operate.	
V	ControlData	CamSyncStruct	Structure containing all information to allow both the CamControl and CamShiftControl to make decisions to run the cam function effectively.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Action	INT	Designates this instance of this function block as one of the several activities to occur based on the registration sensor. For applications that have only one action, such as a cut or a stamp, this input can be left unconnected. This input is required for applications that have more than one action associated with a sensor input, such as pick and place.	INT#0

VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	Engage	BOOL	Set high when the externally located Y_Cam_In function block(s) must be executed.
V	Disengage	BOOL	Set high when the externally located Y_Cam_Out function block(s) must be executed.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

- The Engage output is to be used with a Y_CamIn function block placed external to this function block. This design allows for one or more cam slaves to be operated via the logic provided.
- The Disengage output is to be used with a Y_CamOut function block placed external to this function block. This design allows for one or more cam slaves to be operated via the logic provided.
- This function block is designed to work with the [CamShift_Control](#) function block. It waits for an initial Camshift will occur before the first Engage event should take place. If the application requires the slave to become synchronized with the master without a Camshift, simply use an R_TRIG of the CamControl.Valid to cause the CamData.Shifting bit to go high and low.

Error Description

See the [Function Block ErrorID](#) list.

Example

The operation of CamControl in deciding when to engage and disengage a cam is shown in the logic analyzer illustration below. The rising edge of the CamControl.Shifting variable denotes the "first" product to be processed. First product in this implementation means the cam is disengaged, the ProductBuffer was empty, and a product arrived. Shifting starts immediately if it is the first product in the ProductBuffer. CamControl waits for the falling edge of the Shifting bit to set the CamControl.Engage output. While the cam is engaged, the CamControl block continues to monitor the product buffer for new products. When the ProductBuffer indicates that no products have arrived and the cam cycle has past the 'Decision Position,' the CamControl.Disengage output is turned on.

```

(*Initializing the ProductBufferStruct for Registration Data *)
(*-----*)
20 Products.BufferSize:=INT#20;           (* Maximum size of buffer*)
10.0000000 Products.LockoutDistance:=LREAL#10.0;  (* Looks for a new part only after conveyor has travelled LockOutDistance after previous part
0.0000000 Products.ManualOffset:=LREAL#0.0;
16.5000000 Products.ProductAwayDistance:=LREAL#16.5;  (* Distance from sensor that corresponds to last sync point on the cam profile*)
1 Products.Sensor.Bit:=UINT#1;          (* Equates to EXT1 on a Sigma-5 amplifier, see MC_TouchProbe help for details *)
14.0000000 Products.SensorDistance:=LREAL#14.0;      (* Distance from sensor to centre of sync area in cam profile *)
14.0000000 Products.SensorOffset:=REM(Products.SensorDistance,LREAL#18.0); (* 18 is the cam master cycle *)

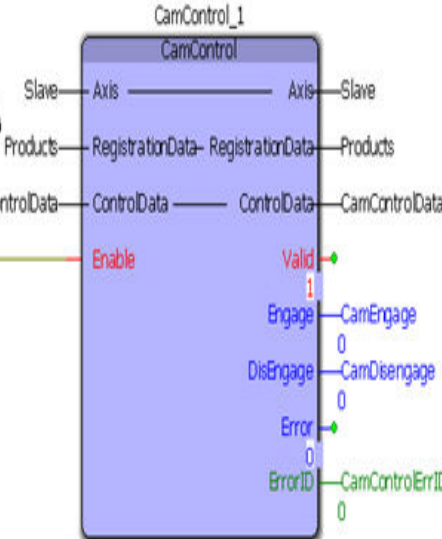
```

Variable Properties

Name: **Products**

Data Type: ProductBufferStruct

Usage: VAR_GLOBAL RETAIN



```

13.0000000 CamControlData.DecisionPosition := LREAL#13.0; (*Position in the cam profile where decision to cam out can be made,
9.0000000 CamControlData.EndSyncPosition := LREAL#9.0;
2 CamControlData.Mode := 2;
4.0000000 CamControlData.StartSyncPosition:= LREAL#4.0;

```

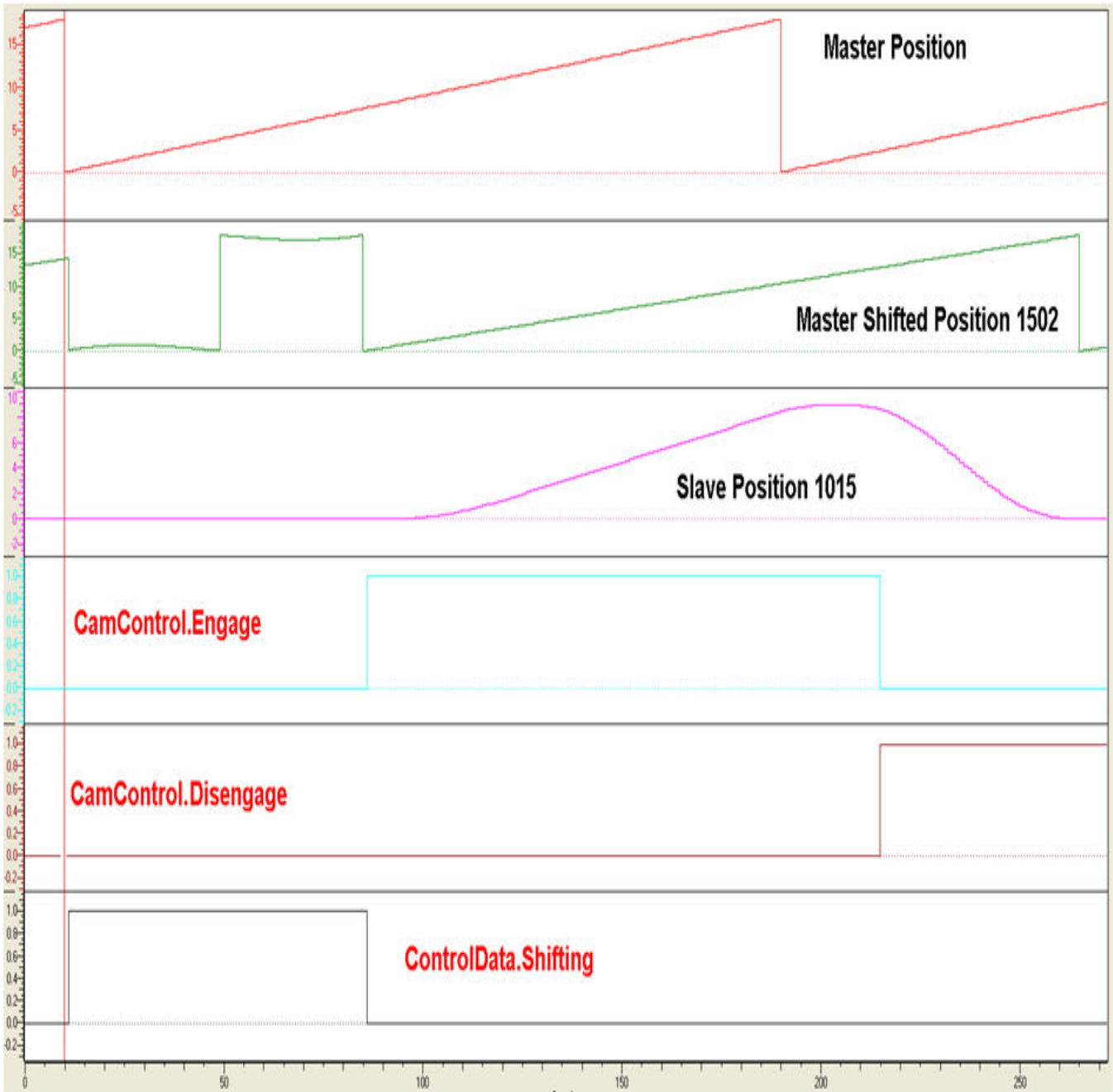
Variable Properties

Name: **CamControlData**

Data Type: CamSyncStruct

Usage: VAR_GLOBAL RETAIN

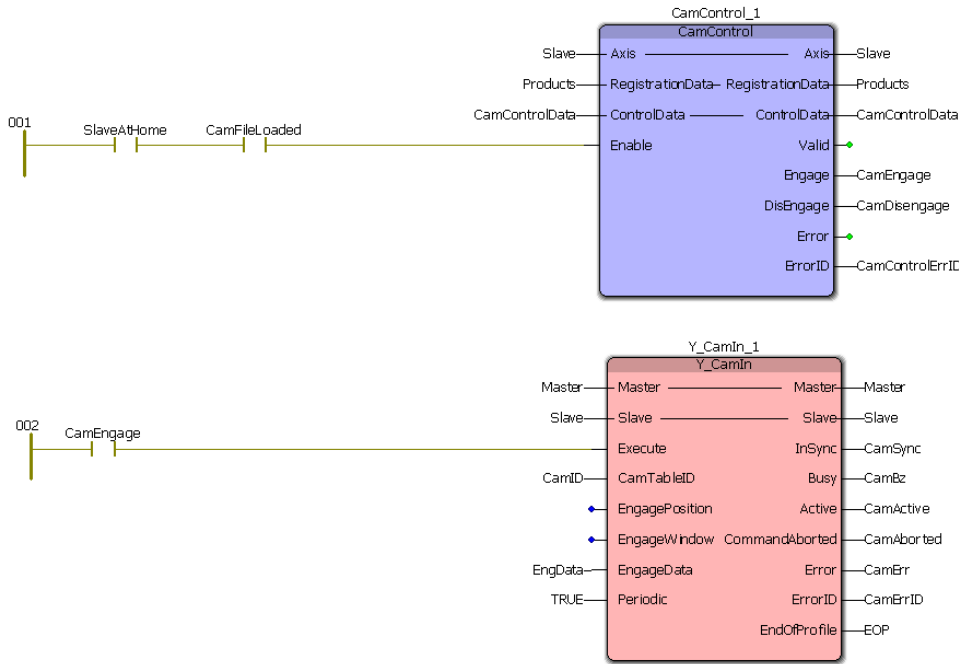
ControlData.Shifting is updated in CamShift_Control
ControlData.Pause is updated in CamControl

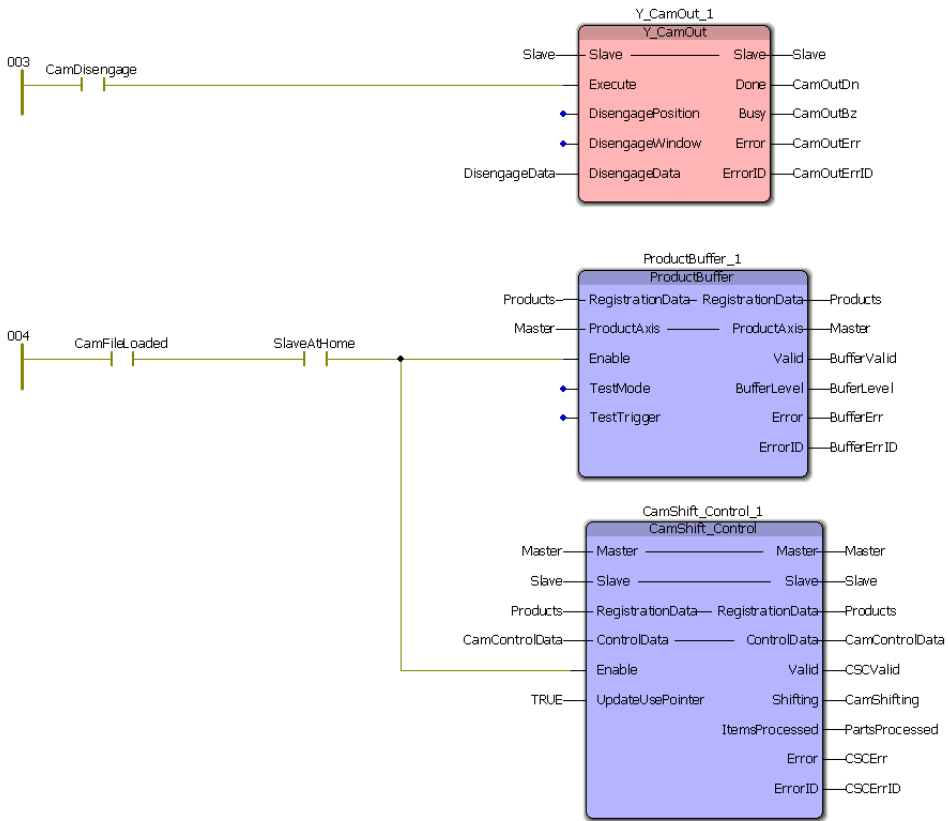


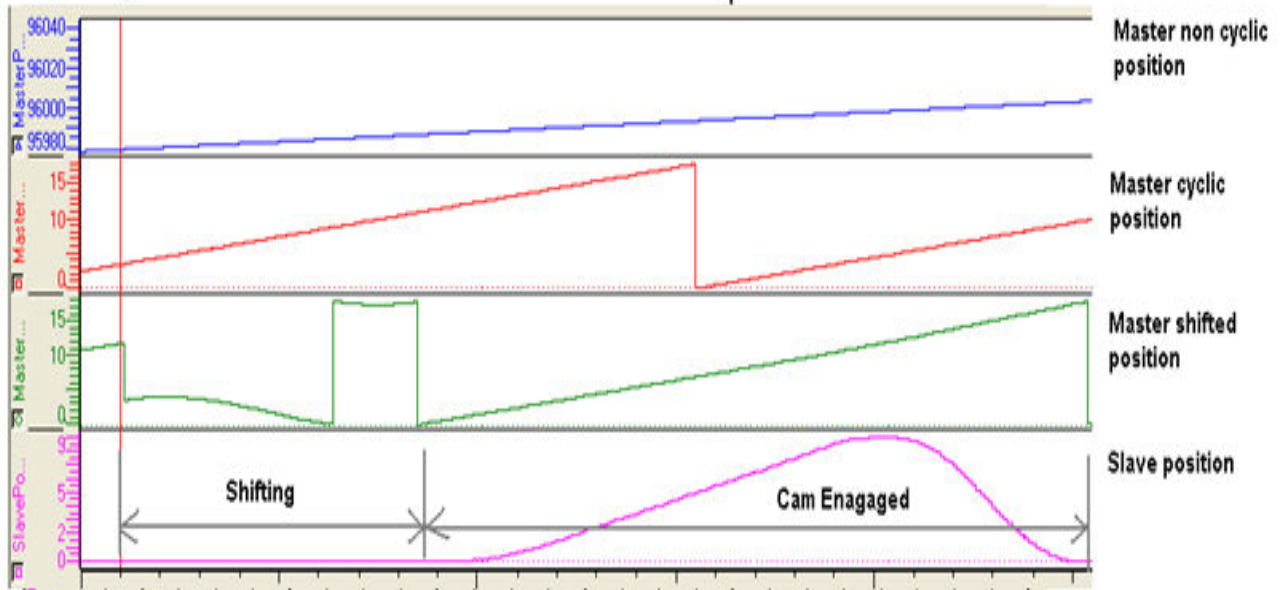
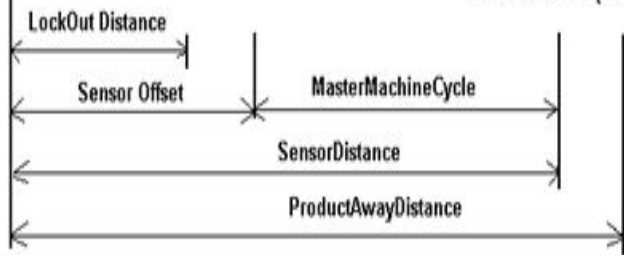
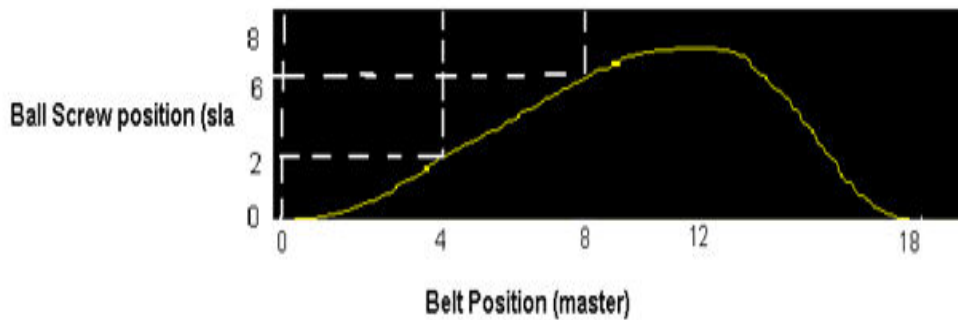
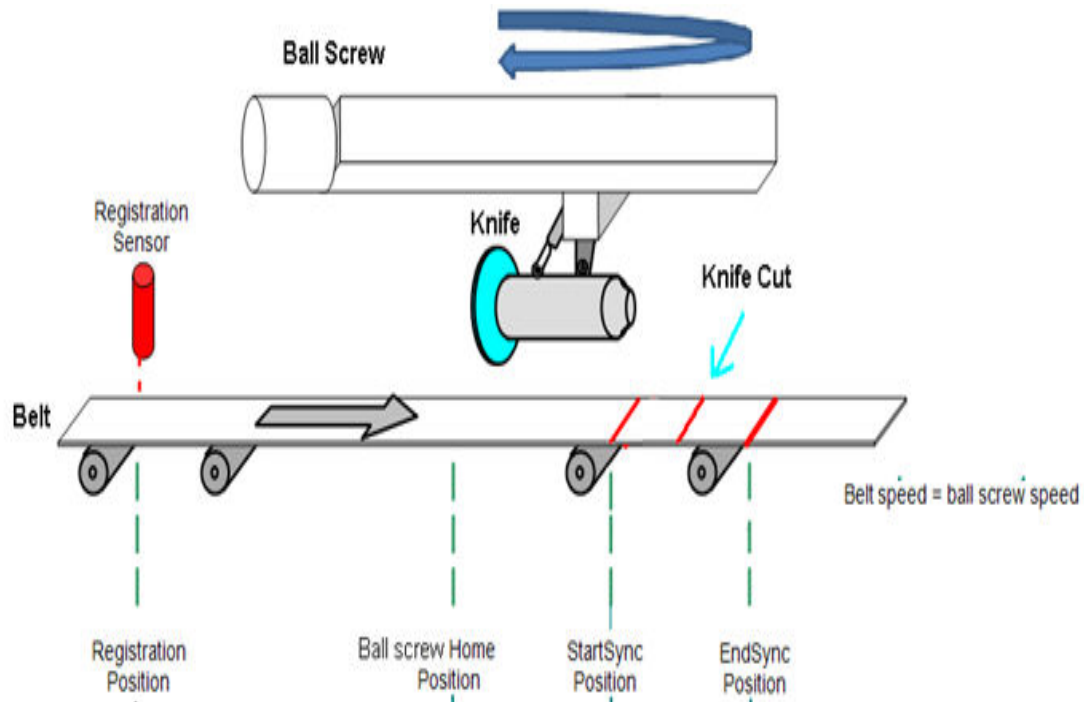
Application Example

This example illustrates how the CamControl block can be applied in a linear flying shear application. In this application, the items to be cut are defective areas (knots) in a piece of wood. The code shown here performs the following actions:

1. The [ProductBuffer](#) stores the position of each defect where a cut must be made.
2. The [CamShift_Control](#) synchronizes the master (conveyor moving the wood) and slave (saw).
3. The CamControl.Engage output must be connected to Y_CamIn.Execute. (Other logic requirements may be included if necessary.)
4. Key Point: When defects are close together, the goal is to remain engaged, and use the CamShift function during the slave (saw) retraction stroke while not in contact with the wood to re-synchronize with the next defect (or knot) to be cut.
5. The CamControl.Disengage output must be connected to Y_CamOut.Execute. In this application, it will cause the slave (saw) to disengage when the ProductBuffer indicates that there are no more defects to be cut.

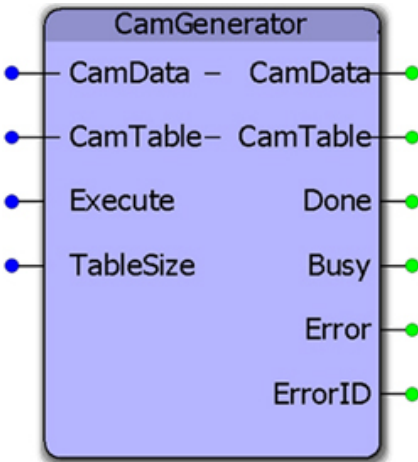








CamGenerator



This function calculates positions required for various master / slave motion profiles. It was designed to replicate the formulas available in Yaskawa’s CamTool & Cam Editor Windows software and includes additional curve types. The CamData input is a structure of key data points required by the application, including a formula code for generating a pair of master / slave data points at the resolution specified. The output CamTable is a [Y_MS_CAM_STRUCT](#) which can be downloaded to the Motion Engine using the Y_CamStructSelect function block.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	CamData	CamSegmentStruct	This structure must be populated with the key datapoints required for the cam profile.	
V	CamTable	Y_MS_CAM_STRUCT	Cam data structure. Can be downloaded to the motion engine using Y_CamStructSelect.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	TableSize	UDINT	This has been deprecated in v350. The function will now determine the TableSize using the UPPER_BOUND function. There is no need to connect a variable.	Calculated using UPPER_BOUND function.

VAR_OUTPUT			
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

- In MotionWorks IEC, array sizes must be hard coded at design time. The default size of the CamSegmentArray DataType can be changed if more segments are required. Edit the Cam Toolbox's DataType definition if necessary. The practical limit on the number of segments, however also depends on the size allocated for Y_MS_PAIR.

The default size of a Y_MS_CAM_STRUCT is defined in the PLCopen Toolbox as:

MS_Array_Type:ARRAY[0..2880] OF Y_MS_PAIR.

If the application requires more than 2880 master / slave pairs, this value can be increased by editing the DataTypes Toolbox > DataTypes > MotionBlockTypes definition.

- The resolution specified for each point in the CamData STRUCT is resolution of the master. For example, if MasterEnd = 100.0, and the previous segment's MasterEnd = 80.0, and the Resolution = 1.0, then 20 data points will be calculated along the CurveType specified.
- See the [Cam Curve Types](#) for further details about creating cam profiles.
- See the [CamGenerator eLearning Module](#) on Yaskawa's YouTube Channel.

Error Description

See the [Function Block ErrorID](#) list.

Examples

Structured text to load a [CamSegmentStruct](#):

Example 1

```

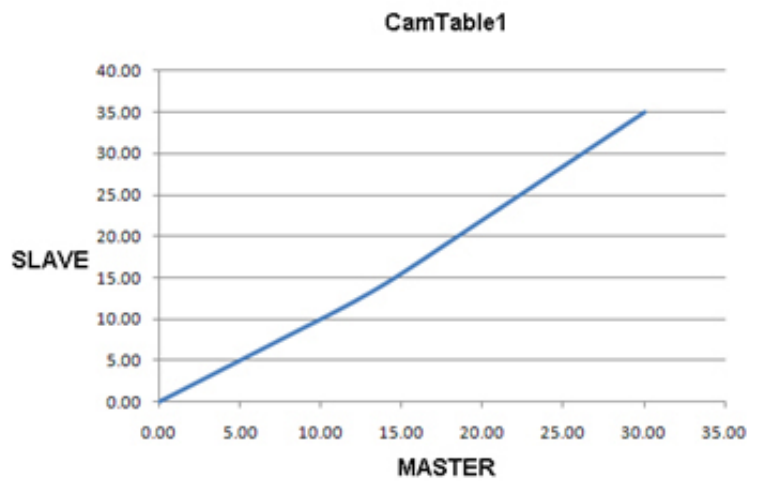
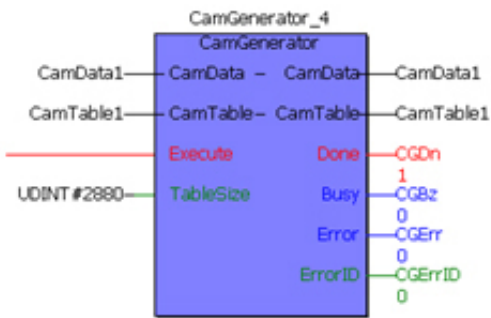
3 CamData1.LastSegment:=INT#3;
0.0000 CamData1.SlaveStart:=LREAL#0.0;

1 CamData1.CamParameters[1].CurveType:=TB_CurveType#StraightLine;
10.0000 CamData1.CamParameters[1].MasterEnd:=LREAL#10.0;
10.0000 CamData1.CamParameters[1].SlaveEnd:=LREAL#10.0;
0.5000 CamData1.CamParameters[1].Resolution:=REAL#0.5;

22 CamData1.CamParameters[2].CurveType:=TB_CurveType#TangentMatching;
20.0000 CamData1.CamParameters[2].MasterEnd:=LREAL#20.0;
22.0000 CamData1.CamParameters[2].SlaveEnd:=LREAL#22.0;
0.5000 CamData1.CamParameters[2].Resolution:=REAL#0.5;

1 CamData1.CamParameters[3].CurveType:=TB_CurveType#StraightLine;
30.0000 CamData1.CamParameters[3].MasterEnd:=LREAL#30.0;
35.0000 CamData1.CamParameters[3].SlaveEnd:=LREAL#35.0;
0.5000 CamData1.CamParameters[3].Resolution:=REAL#0.5;

```



Example 2

```

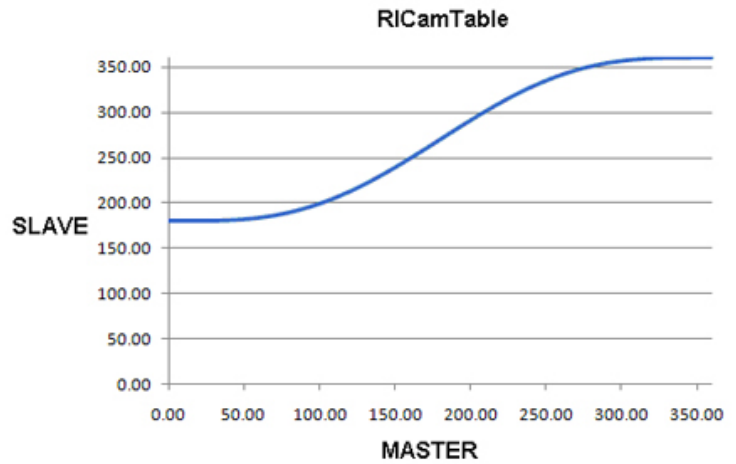
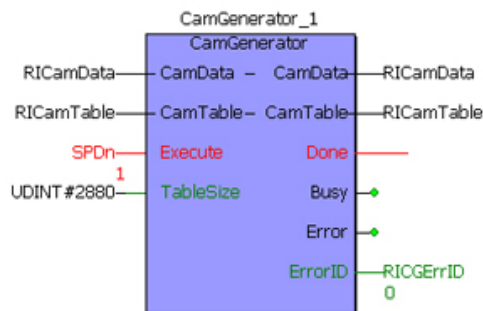
3 RICamData.LastSegment:=INT#3;
180.0000 RICamData.SlaveStart:=LREAL#180.0;

1 RICamData.CamParameters[1].CurveType:=TB_CurveType#StraightLine;
10.0000 RICamData.CamParameters[1].MasterEnd:=LREAL#10.0;
180.0000 RICamData.CamParameters[1].SlaveEnd:=LREAL#180.0;
1.0000 RICamData.CamParameters[1].Resolution:=REAL#1.0;

22 RICamData.CamParameters[2].CurveType:=TB_CurveType#TangentMatching;
350.0000 RICamData.CamParameters[2].MasterEnd:=LREAL#350.0;
350.0000 RICamData.CamParameters[2].SlaveEnd:=LREAL#350.0;
1.0000 RICamData.CamParameters[2].Resolution:=REAL#1.0;

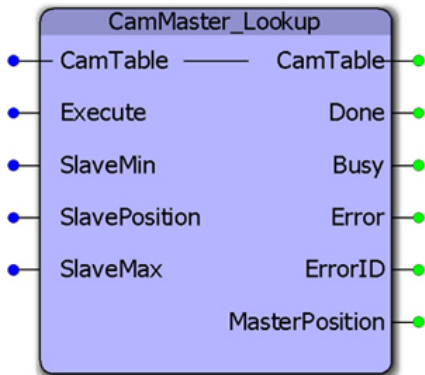
1 RICamData.CamParameters[3].CurveType:=TB_CurveType#StraightLine;
360.0000 RICamData.CamParameters[3].MasterEnd:=LREAL#360.0;
360.0000 RICamData.CamParameters[3].SlaveEnd:=LREAL#360.0;
1.0000 RICamData.CamParameters[3].Resolution:=REAL#1.0;

```





CamMaster_Lookup



This function block provides the master position given a slave position by searching the referenced CamTable. If there may be two or more master positions for the slave, as in the case of out and back slave motion, a range of slave positions can be specified to limit the search for the corresponding master position. This function block is useful for E-Stop recovery routines.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	CamTable	Y_MS_CAM_STRUCT	Cam data structure	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	SlaveMin	LREAL	The smallest slave position to include when searching for the master.	LREAL#0.0
V	SlavePosition	LREAL	The current slave position.	LREAL#0.0
B	SlaveMax	LREAL	The largest slave position to include when searching for the master.	LREAL#0.0
VAR_OUTPUT				

B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
B	MasterPosition	LREAL	The master position which corresponds to the SlavePosition.

Notes

This function provide the exact master position that corresponds to the SlavePostion input by interpolating the CamTable. Consider the following CamTable:

M	S
0	0
10	0
20	5
30	10
40	20

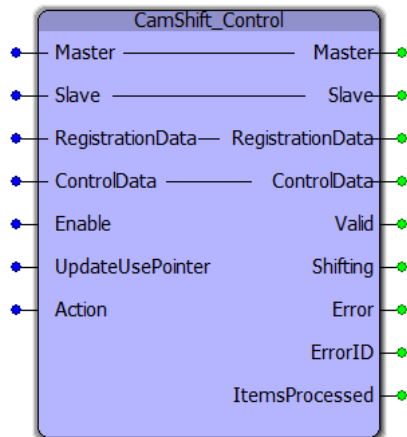
If the SlavePosition is 15, the corresponding MasterPosition is 35.

Error Description

See the [Function Block ErrorID](#) list.



CamShift_Control



The CamShift_Control block manages cam shifting for applications that buffer random products such as Linear Flying Shear or Random Rotary Placer/Knife/Drill, etc. The purpose is to re-synchronize the slave for each item or product arriving on the master axis.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Master	AXIS_REF	A logical reference to the master axis	
B	Slave	AXIS_REF	A logical reference to the slave axis	
V	RegistrationData	ProductBufferStruct	Structure containing all information for the circular buffer to operate.	
V	ControlData	CamSyncStruct	Structure containing all information about the cam profile that will be used to calculate and implement cam shifts	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	UpdateUsePointer	BOOL	RegistrationData.UsePointer will be updated when a product has been processed only if this input is TRUE. If more than one slave follow the master, only the last slave must update the UsePointer.	FALSE

V	Action	INT	Designates this instance of this function block as one of the several activities to occur based on the registration sensor. For applications that have only one action, such as a cut or a stamp, this input can be left unconnected. This input is required for applications that have more than one action associated with a sensor input, such as pick and place.	INT # 1
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	Shifting	BOOL	Set high if the function block is active and Y_CamShift is Busy.	
V	ItemsProcessed	UDINT	Provides a count of the number of products processed since this function was enabled.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

- This function block includes a Y_CamShift block, and will execute shifts at the appropriate position based on data provided by the user via the ControlData structure.
- The shifted master position is available by reading slave axis parameter 1502.
- This function block requires the [ProductBuffer](#) function block from the PLCopen Toolbox and the CamControl block from the Cam Toolbox. These three blocks work together to provide cam engage/disengage control as well as cam shifting (synchronization) logic.
- The 'Shifting' bit is held high when a Y_CamShift is in progress.
- The CamShift_Control block uses data from RegistrationData and ControlData to make decisions on when to shift the master position and by how much to shift the position. The user must provide valid data in the RegistrationData and ControlData structures.
- In cases where multiple slaves are synchronized to a single master, the slaves can share the same ProductBuffer. Set the last slave (last CamShift_Control function block) to update the UsePointer for the ProductBuffer.

Error Description

See the [Function Block ErrorID](#) list.

Code Example

The role of CamShift_Control in master / slave synchronization for each product is illustrated below.


```

(*Initializing the ProductBufferStruct for Registration Data *)
(*-----*)
20 Products.BufferSize:=INT#20;          (* Maximum size of buffer*)
10.0000000 Products.LockoutDistance:=LREAL#10.0;  (* Looks for a new part only after conveyor has travelled LockOutDistance after previous part
0.0000000 Products.ManualOffset:=LREAL#0.0;
16.5000000 Products.ProductAwayDistance:=LREAL#16.5;  (* Distance from sensor that corresponds to last sync point on the cam profile*)
1 Products.Sensor.Bit:=UINT#1;          (* Equates to EXT1 on a Sigma-5 amplifier, see MC_TouchProbe help for details *)
14.0000000 Products.SensorDistance:=LREAL#14.0;      (* Distance from sensor to centre of sync area in cam profile *)
14.0000000 Products.SensorOffset:=REN(Products.SensorDistance.LREAL#18.0); (* 18 is the cam master cycle *)

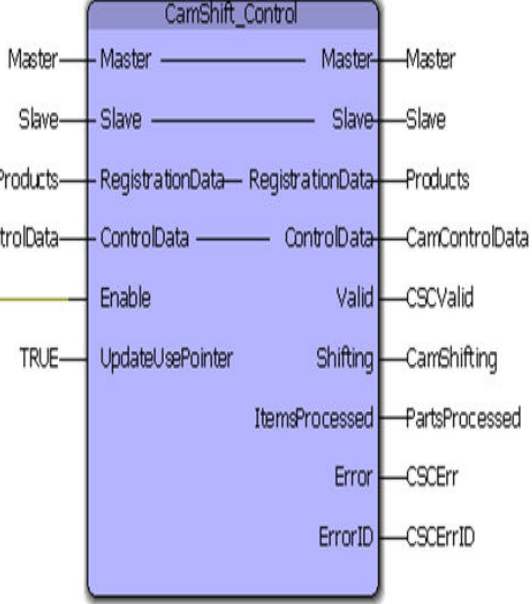
```

Variable Properties

Name: **Products**

Data Type: ProductBufferStruct

Usage: VAR_GLOBAL RETAIN



```

13.0000000 CamControlData.DecisionPosition := LREAL#13.0; (*Position in the cam profile where decision to cam out can be made,
9.0000000 CamControlData.EndSyncPosition := LREAL#9.0;
2 CamControlData.Mode := 2;
4.0000000 CamControlData.StartSyncPosition:= LREAL#4.0;

```

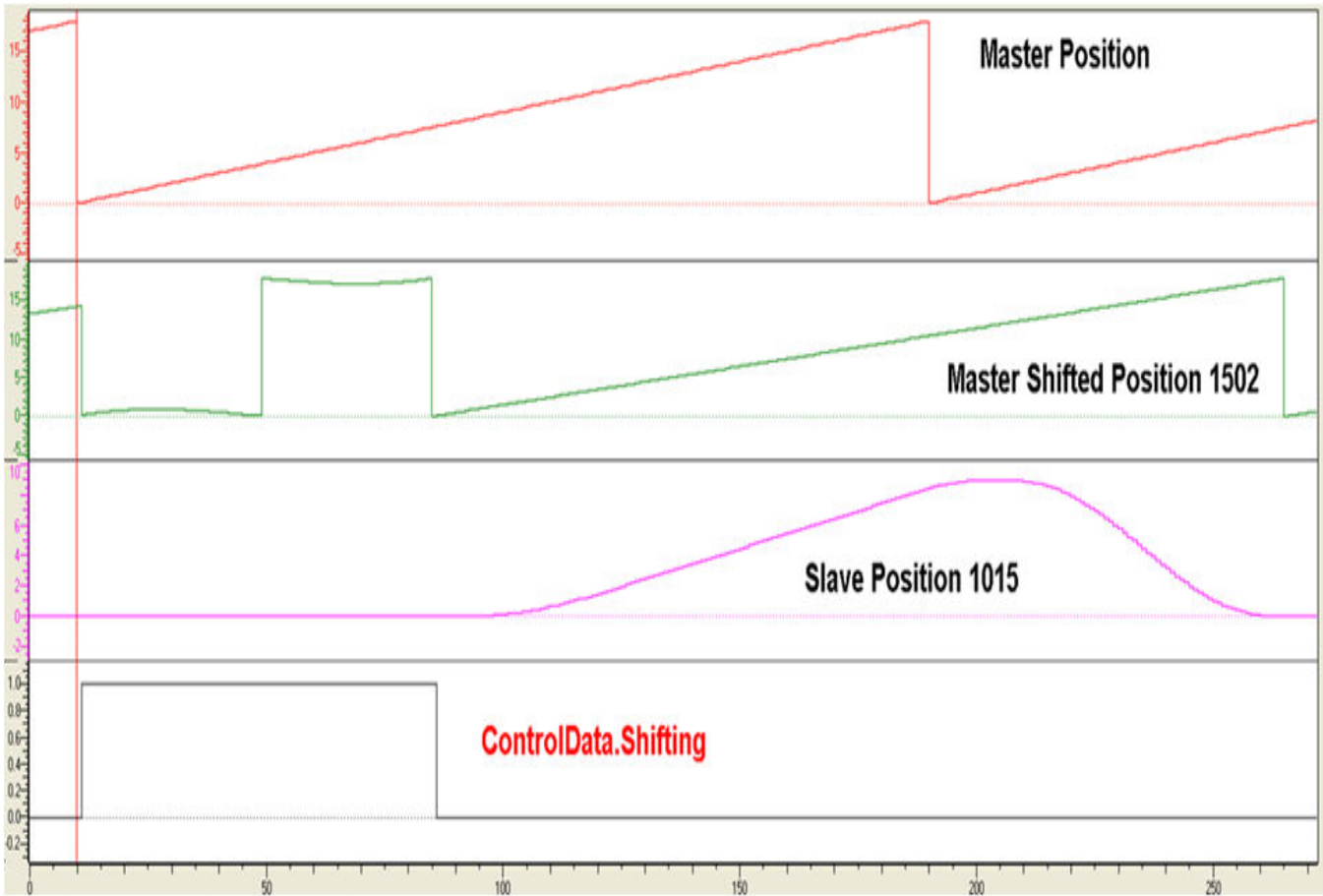
Variable Properties

Name: **CamControlData**

Data Type: CamSyncStruct

Usage: VAR_GLOBAL RETAIN

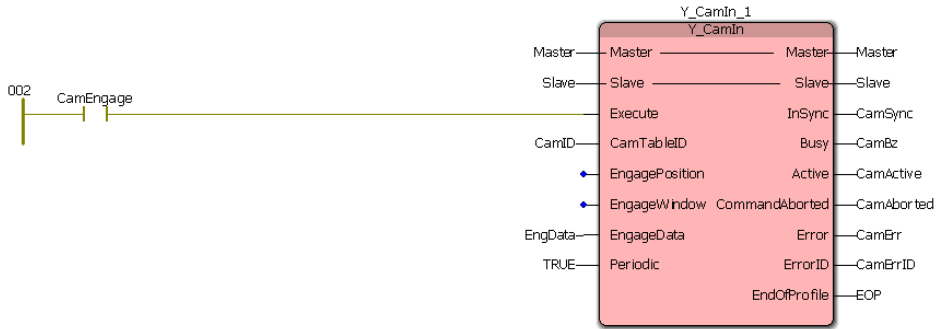
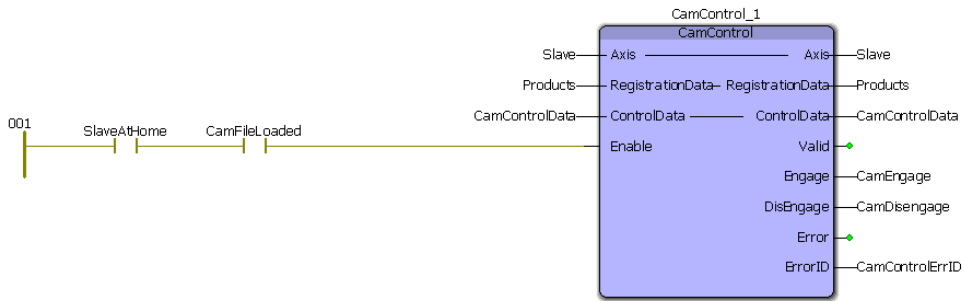
ControlData.Shifting is updated in CamShift_Control
ControlData.Pause is updated in CamControl

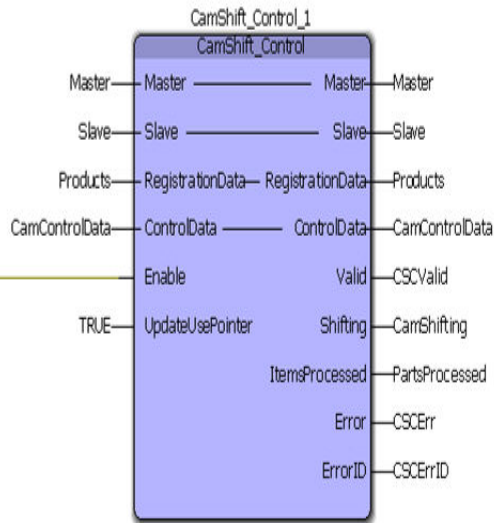
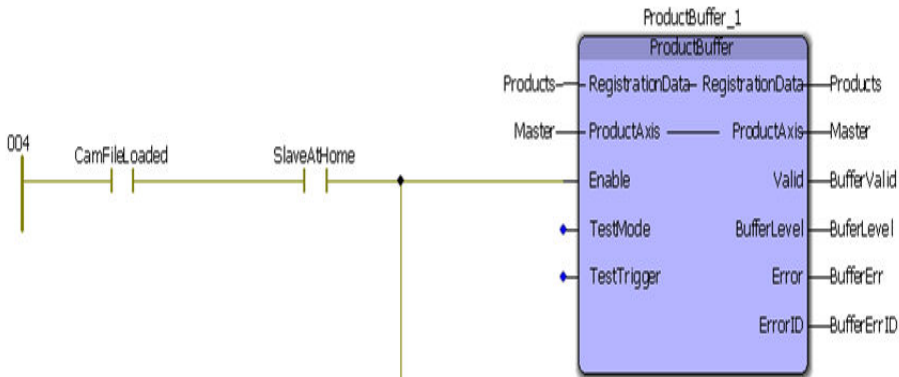
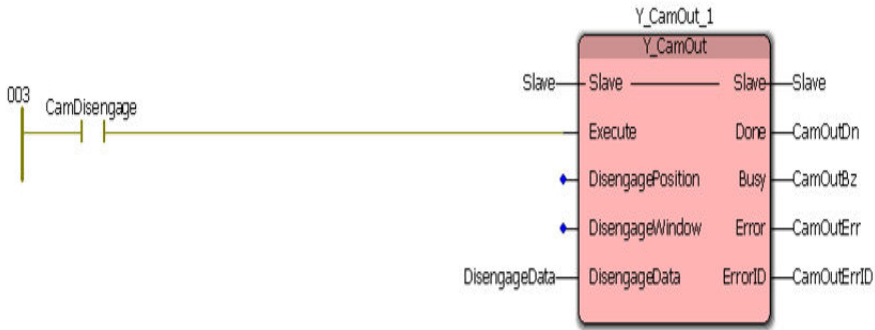


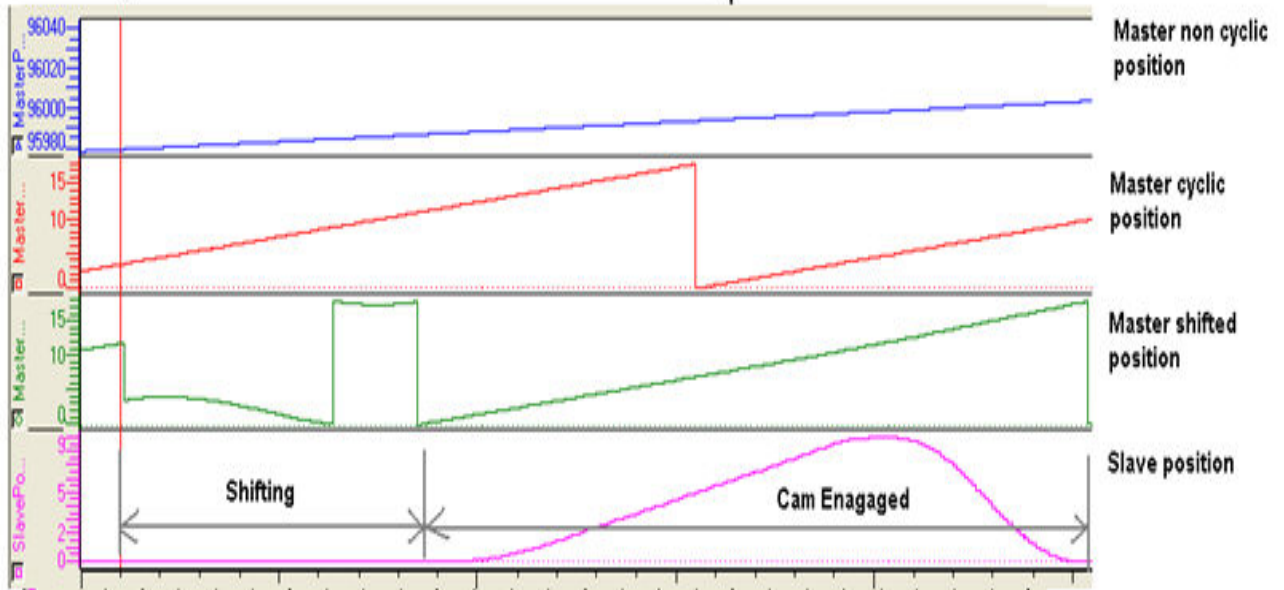
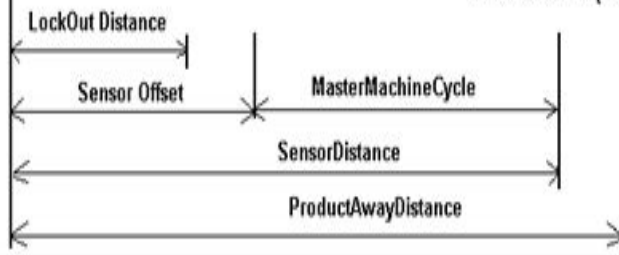
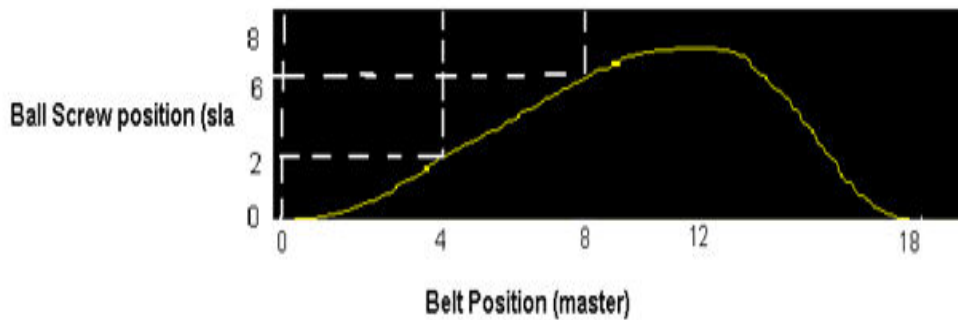
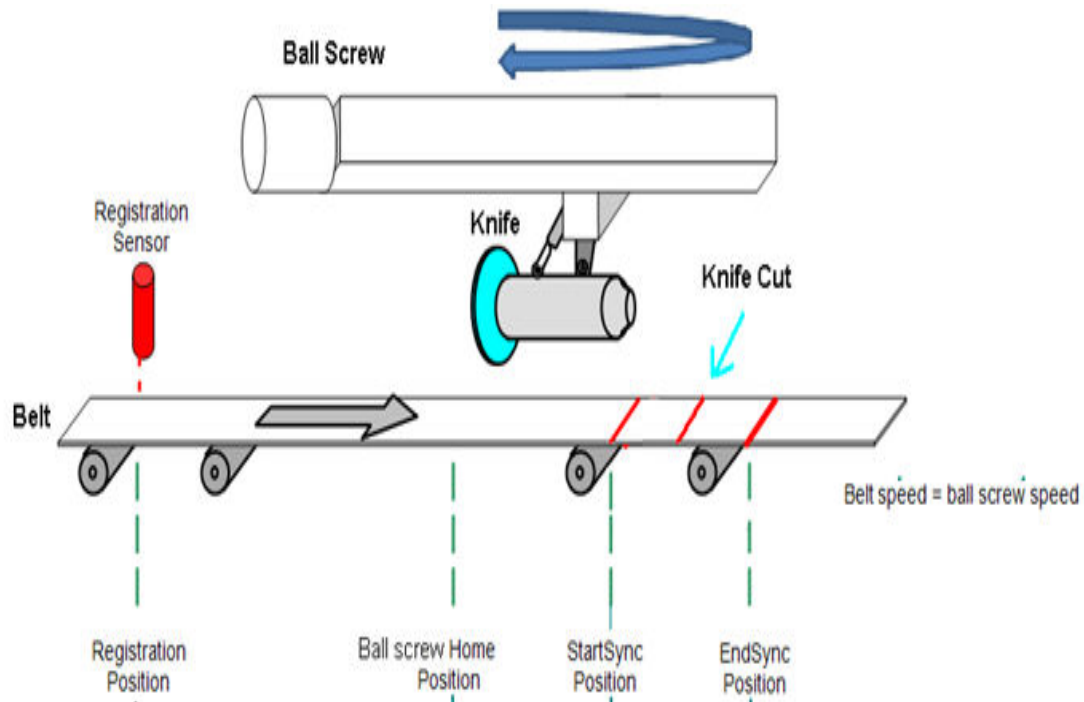
Application Example

This example illustrates how the CamControl block can be applied in a linear flying shear application. In this application, the items to be cut are defective areas (knots) in a piece of wood. The code shown here performs the following actions:

1. The [ProductBuffer](#) stores the position of each defect where a cut must be made.
2. The [CamShift_Control](#) synchronizes the master (conveyor moving the wood) and slave (saw).
3. The CamControl.Engage output must be connected to Y_CamIn.Execute. (Other logic requirements may be included if necessary.)
4. Key Point: When defects are close together, the goal is to remain engaged, and use the CamShift function during the slave (saw) retraction stroke while not in contact with the wood to re-synchronize with the next defect (or knot) to be cut.
5. The CamControl.Disengage output must be connected to Y_CamOutExecute. In this application, it will cause the slave (saw) to disengage when the [ProductBuffer](#) indicates that there are no more defects to be cut.

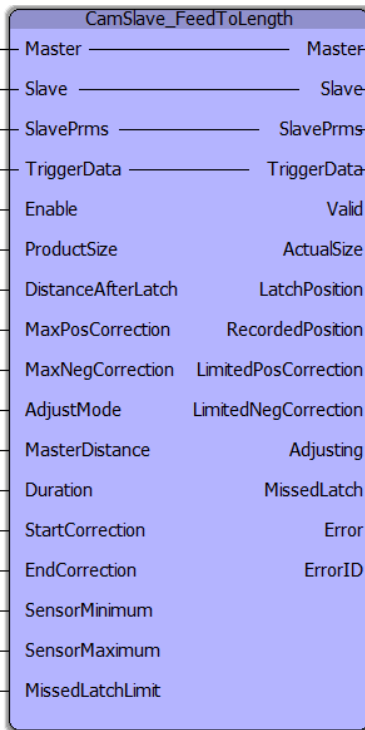








CamSlave_FeedToLength



CamSlave_FeedToLength was designed for use with camming applications that index a slave axis forward in one direction, and require on the fly adjustments of the actual index length based on a sensor input that occurs while the slave is moving. The sensor input is on the slave axis.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Master	AXIS_REF	A logical reference to the master axis.
B	Slave	AXIS_REF	A logical reference to the slave axis.
V	SlavePrms	AxisParameterStruct	User Defined DataType declared in the PLCopen Toolbox.
E	TriggerData	TRIGGER_REF	Reference to the trigger signal source.. Refer to the PLCopen Plus Function Block Manual for more details.
VAR_INPUT			Default

B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	ProductSize	LREAL	This value must be the same as the total one way index of the cam profile for this slave.	LREAL#0.0
V	DistanceAfterLatch	LREAL	The desired additional travel distance after the registration mark is detected	LREAL#0.0
V	MaxPosCorrection	LREAL	Limits the amount of positive correction that can be applied.	
V	MaxNegCorrection	LREAL	Limits the amount of negative correction that can be applied.	
V	AdjustMode	INT	An ENUM for TIME or range of master correction, with the following values:	
V	MasterDistance	LREAL	Relative amount the master will travel (in cam master units) from when the function block first executes until the correction is complete. Only used if AdjustMode = Y_AdjustMode#Master-Distance.	
V	Duration	LREAL	Time of the correction used if AdjustMode is set for TIME mode	
V	StartCorrection	LREAL	Earliest master position where the correction can begin.	LREAL#0.0
V	FinishCorrection	LREAL	Latest master position where the correction must be completed.	LREAL#0.0
V	SensorMinimum	LREAL	The earliest slave position where a sensor position is valid for correction.	LREAL#0.0
V	SensorMaximum	LREAL	The latest slave position where a sensor position is valid for correction.	LREAL#0.0 (function block defaults to ProductSize if unconnected.)
V	MissedLatchLimit	UINT	The number of consecutive DefaultDistances allowed to occur without seeing a registration mark in the window, and not cause an Error. Valid registration marks will reset the internal counter.	UINT#0 (interpreted as infinite)
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	ActualSize	LREAL	The actual indexed distance.	
V	LatchPosition	LREAL	The slave's position in the CamTable when the latch occurred.	
B	RecordedPosition	LREAL	The slaves latch position as reported by MC_TouchProbe.	
V	LimitedPosCorrection	BOOL	Indicates that the MaxPosCorrection is limiting the required correction.	
V	LimitedNegCorrection	BOOL	Indicates that the MaxNegCorrection is limiting the required correction.	
V	Adjusting	BOOL	Indicates that an adjustment is currently taking place (Busy output of Y_SlaveOffset)	
V	MissedLatch	BOOL	Indicates that a latch was detected, but it was outside of the window parameters specified.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

- This function block requires that the [ReadAxisParameters](#) function block from the PLCopen toolbox is also running, preferably in the same task as CamSlaveFeedToLength.
- This function block does not support buffering of products. It is recommended to place the sensor less than 1 part length away from where the correction must happen.
- Functionality differences between the CamSlave_FeedToLength and [SlaveOffset_Control](#) are given in the table shown below.

	CamSlave_FeedToLength	SlaveOffset_Control
BufferProducts (SensorDistance > 1 part length)	Not supported	Supported
Successive triggers > 2 part lengths	Reports missed latch	Makes large correction
Missed Part Indicator	Supported	Not supported
Registration within window check	Supported	Not supported
Correction limits	Supported	Not Supported

- See the [CamSlave_FeedToLength eLearning Module](#) on Yaskawa's YouTube Channel.

Missed Latch Detection feature:

There are two parts to this feature.

1) It will report an ErrorID 10021 if the user enters a non zero value for the MissedLatchLimit and a consecutive number of latches are not counted. (To detect a hardware failure or other problem with system such as a sensor blockage.)

2) If latches are detected, but are outside of the SensorMinimum and SensorMaximum range, it is not considered a missed latch in terms of counting up to the MissedLatchLimit. In this condition, the function block will pulse the MissedLatch output to indicate that no correction will be made because the latch is not in the specified area. The user can track the MissedLatch output pulses to make adjustments to the machine, or open the window for first time synchronization of the master and slave.

In Cam Toolbox v204, this function block was modified to report the RecordedPosition as a new output so that applications can use this information to re position or re home the axis after a manual operation without adding a separate MC_TouchProbe function block in the application. The function was also modified to prohibit its internal Y_SlaveOffset from executing if no cam is engaged.

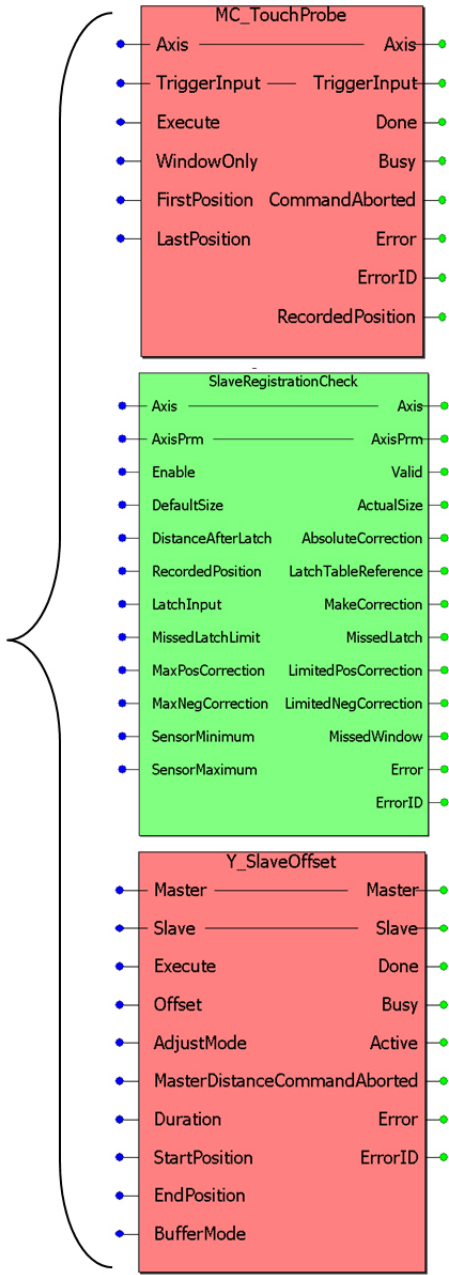
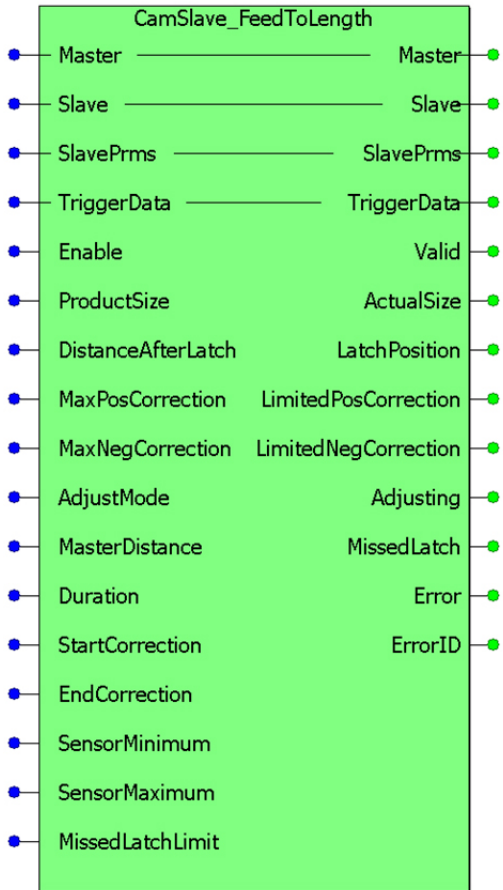
Error Description

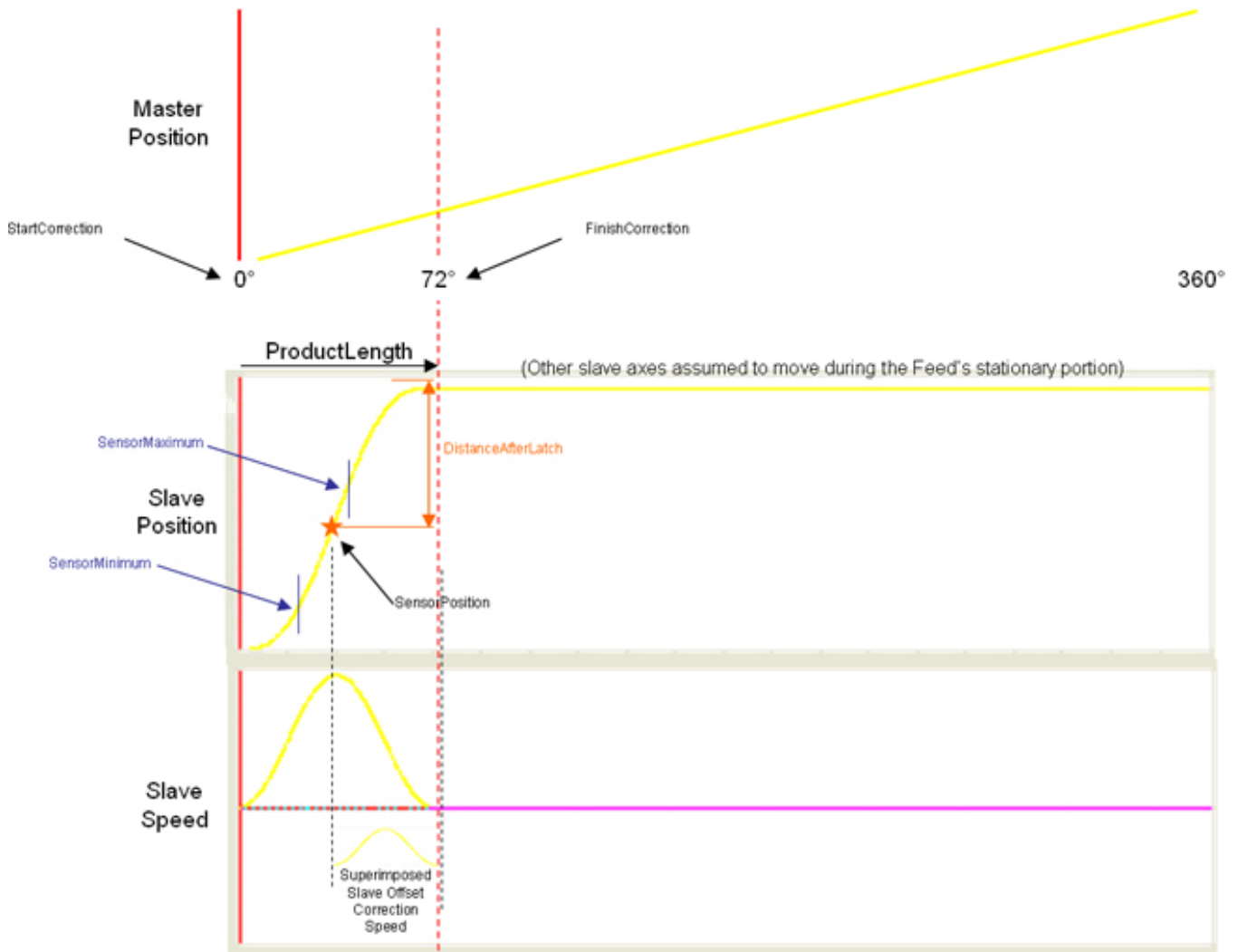
See the [Function Block ErrorID](#) list.

Applications

- Label Feeder
- Punch Press
- Intermittent Form Fill and Seal

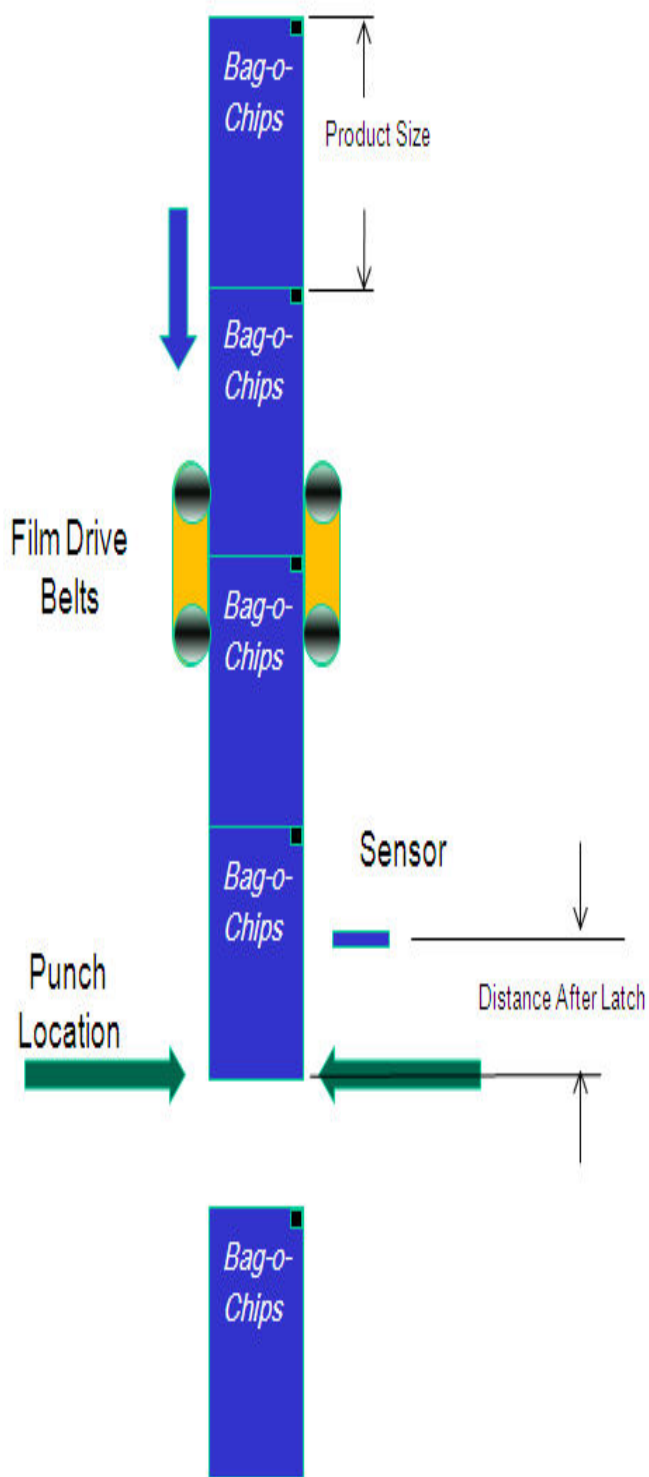
Overview of Supporting Function Blocks



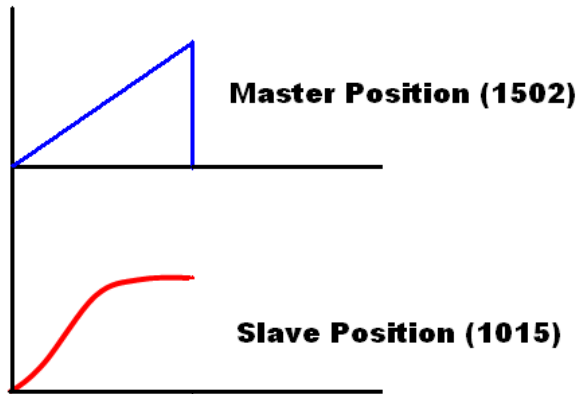


Application Example

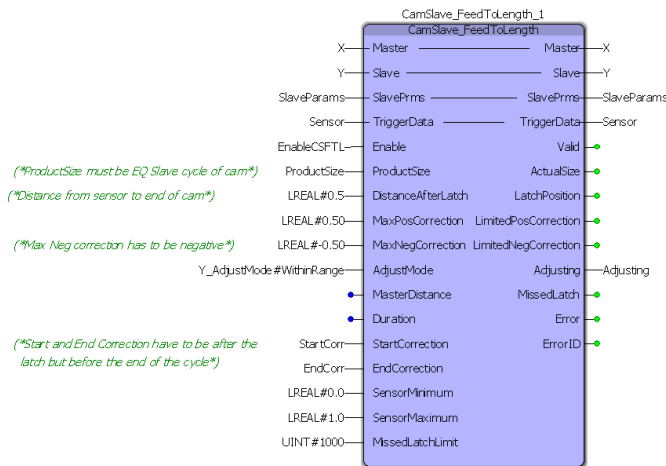
Consider a form fill and seal application as shown below. Feed belts control payout of film for the form fill and seal machine. Distance After Latch is set to align the end of bag with the cutter/punch



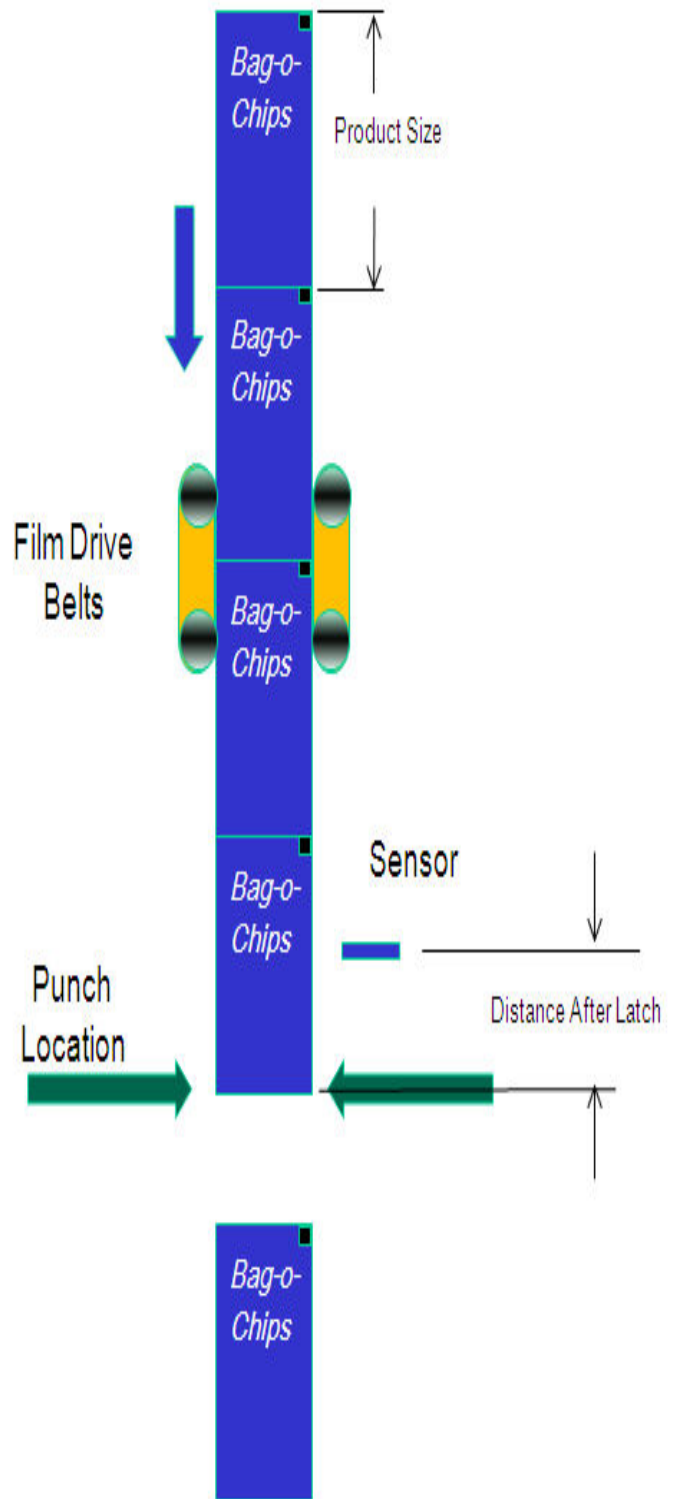
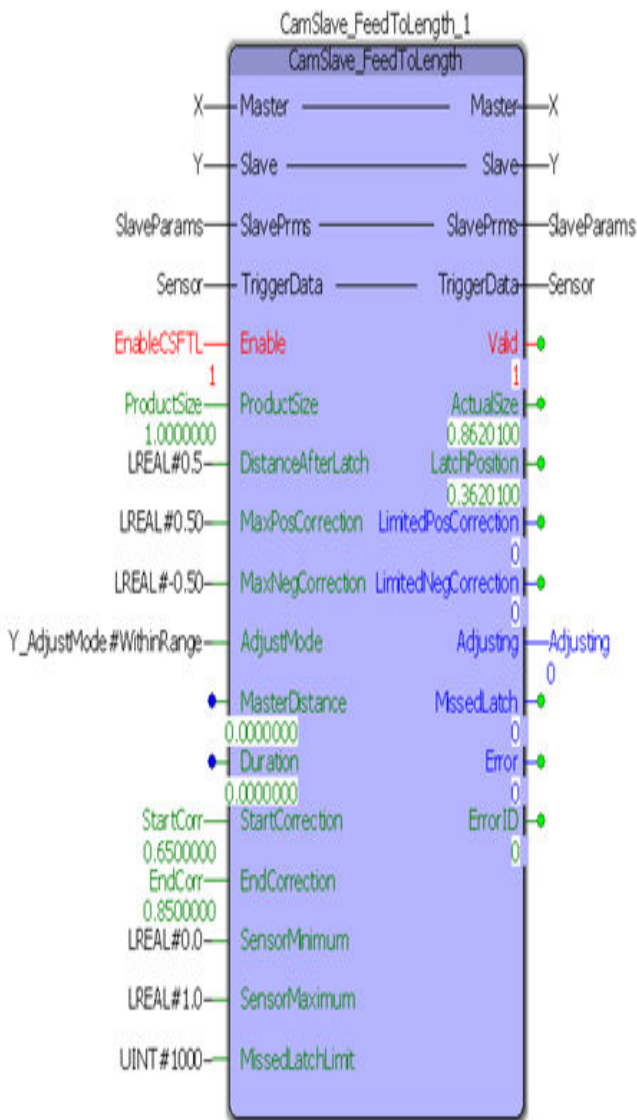
The film drive belt is the slave to a constantly running master. The nominal cam table is shown below. The master cycle is 0 - 1 units and the slave cycle is also between 0 and 1 units.



A sample screen shot of data that must be entered for the system described above is shown in the figure below. Care should be taken to ensure that the input parameters will generate motion that is physically achievable by the slave axis.



In the screen shot of the CamSlave_FeedToLength block shown below, the sensor detects a registration mark at 0.36201 units of the slave cycle. Assuming that the previous registration mark was captured at 0.5 units of the slave cycle, the distance between two successive registrations is 0.86201 units (0.5 + 0.36201). The actual bag length in this case is 0.86201 units.



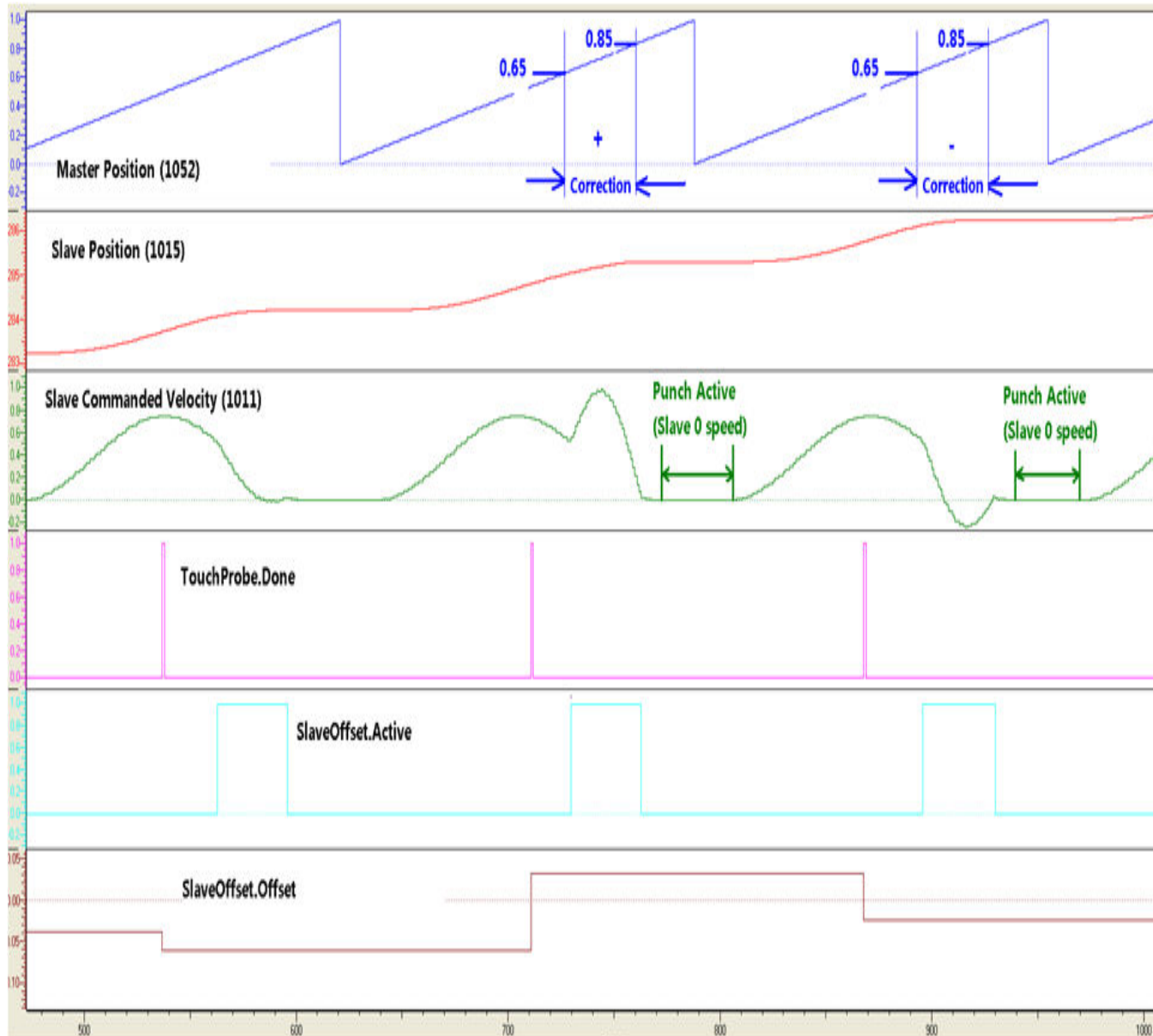
The calculation for the adjustment required by the slave axis (film feed) to place the film exactly at the cutter/pinch location is explained below:

$$\text{Correction} = \text{Nominal part size} (1.0) - \text{Actual bag length} (0.86201) = -0.1379$$

This will be the offset added/subtracted (for this cycle) to any previous offsets in the slave position.

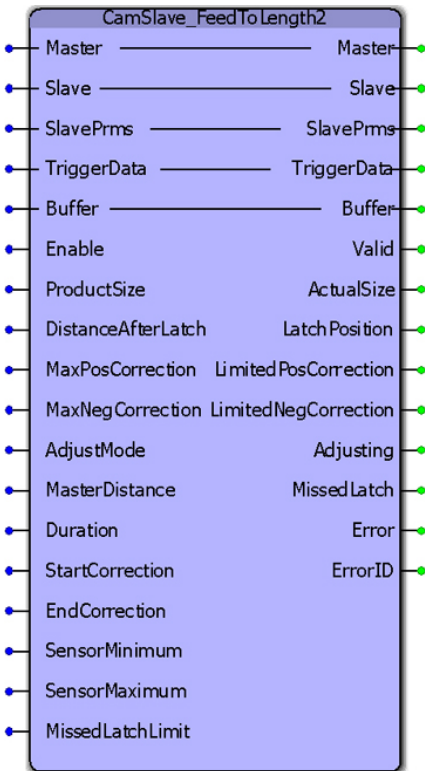
A continuous sequence of short, long, short bag lengths is illustrated in the logic analyzer plots below.

The first occurrence of TouchProbe.Done triggers a calculation that results in a short bag. A small negative offset is calculated and can be seen by the dip to negative velocity at the end of the first master cycle. The registration mark in the middle of the second master cycle triggers a calculation that results in a long bag and a positive offset. Notice the spike in slave velocity between 0.65 and 0.86 units of the master cycle. The last registration mark in the middle of the third master cycle triggers a calculation that results in a short bag and a negative offset. Notice the dip in slave velocity between 0.65 and 0.86 units of the master cycle.





CamSlave_FeedToLength2



CamSlave_FeedToLength2 is an enhancement of CamSlave_FeedtoLength. The only difference is the increased performance in capturing latches that occur at higher frequency by incorporating the Y_ProbeContinuous function block. As with CamSlave_FeedtoLength, this function block was designed for use with camming applications that index a slave axis forward in one direction, and require on the fly adjustments of the actual index length based on a sensor input. The sensor input is on the slave axis.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Master	AXIS_REF	A logical reference to the master axis.
B	Slave	AXIS_REF	A logical reference to the slave axis.

V	SlavePrms	AxisParameterStruct	User Defined DataType declared in the PLCopen Toolbox.
E	TriggerData	TRIGGER_REF	Reference to the trigger signal source.. Refer to PLCopen Plus Function Block Manual for more details.
V	Buffer	CONTINUOUS_REF	
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	ProductSize	LREAL	This value must be the same as the total one way index of the cam profile for this slave.
V	DistanceAfterLatch	LREAL	The desired additional travel distance after the registration mark is detected
V	MaxPosCorrection	LREAL	Limits the amount of positive correction that can be applied.
V	MaxNegCorrection	LREAL	Limits the amount of negative correction that can be applied.
V	AdjustMode	INT	An ENUM for TIME or range of master correction, with the following values:
V	MasterDistance	LREAL	Relative amount the master will travel (in cam master units) from when the function block first executes until the correction is complete. Only used if AdjustMode = Y_AdjustMode#MasterDistance.
V	Duration	LREAL	Time of the correction used if AdjustMode is set for TIME mode
V	StartCorrection	LREAL	Earliest master position where the correction can begin.
V	FinishCorrection	LREAL	Latest master position where the correction must be completed.
V	SensorMinimum	LREAL	The earliest slave position where a sensor position is valid for correction.
V	SensorMaximum	LREAL	The latest slave position where a sensor position is valid for correction.
V	MissedLatchLimit	UINT	The number of consecutive DefaultDistances allowed to occur without seeing a registration mark in the window, and not cause an Error. Valid registration marks will reset the internal counter.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	ActualSize	LREAL	The actual indexed distance.
V	LatchPosition	LREAL	The slave's position in the CamTable when the latch occurred.
V	LimitedPosCorrection	BOOL	Indicates that the MaxPosCorrection is limiting the required correction.
V	LimitedNegCorrection	BOOL	Indicates that the MaxNegCorrection is limiting the required correction.
V	Adjusting	BOOL	Indicates that an adjustment is currently taking place (Busy output of Y_SlaveOffset)
V	MissedLatch	BOOL	Indicates that a latch was detected, but it was outside of the window parameters specified.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.

E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
---	-------------------------	------	--

Notes

The slave axis must be Sigma-5 or Sigma-7 servo amplifier when using this function block.

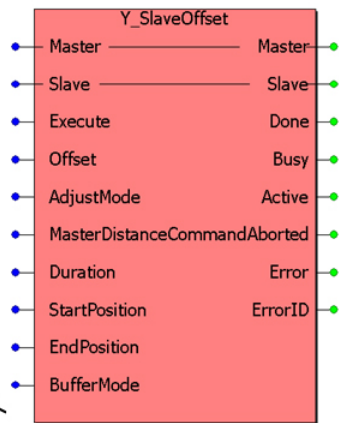
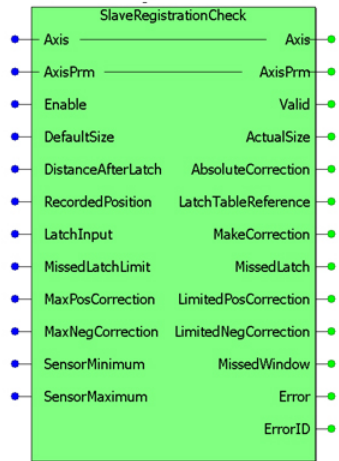
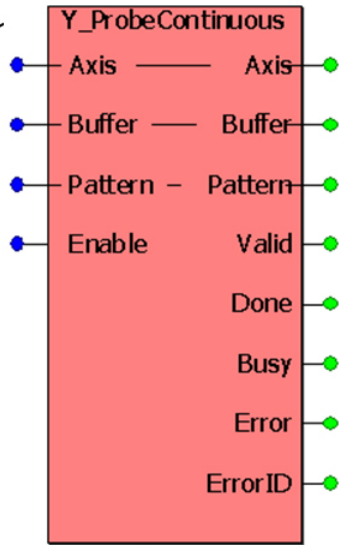
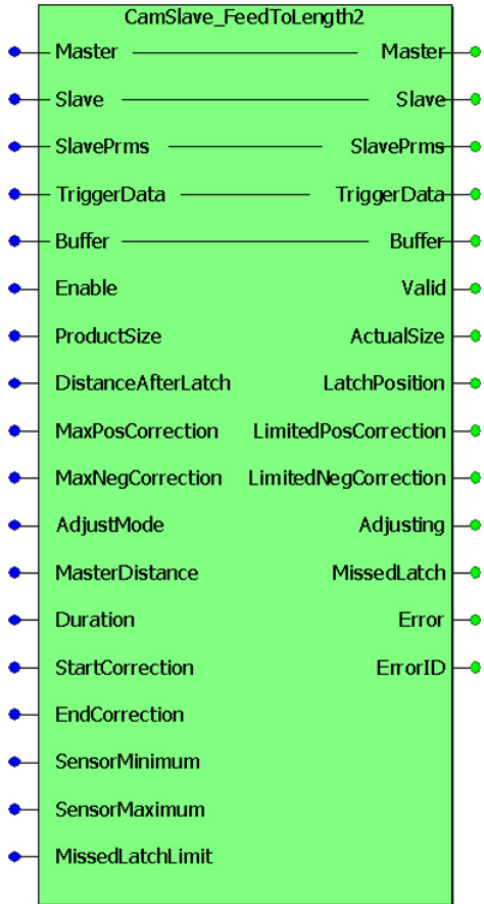
Error Description

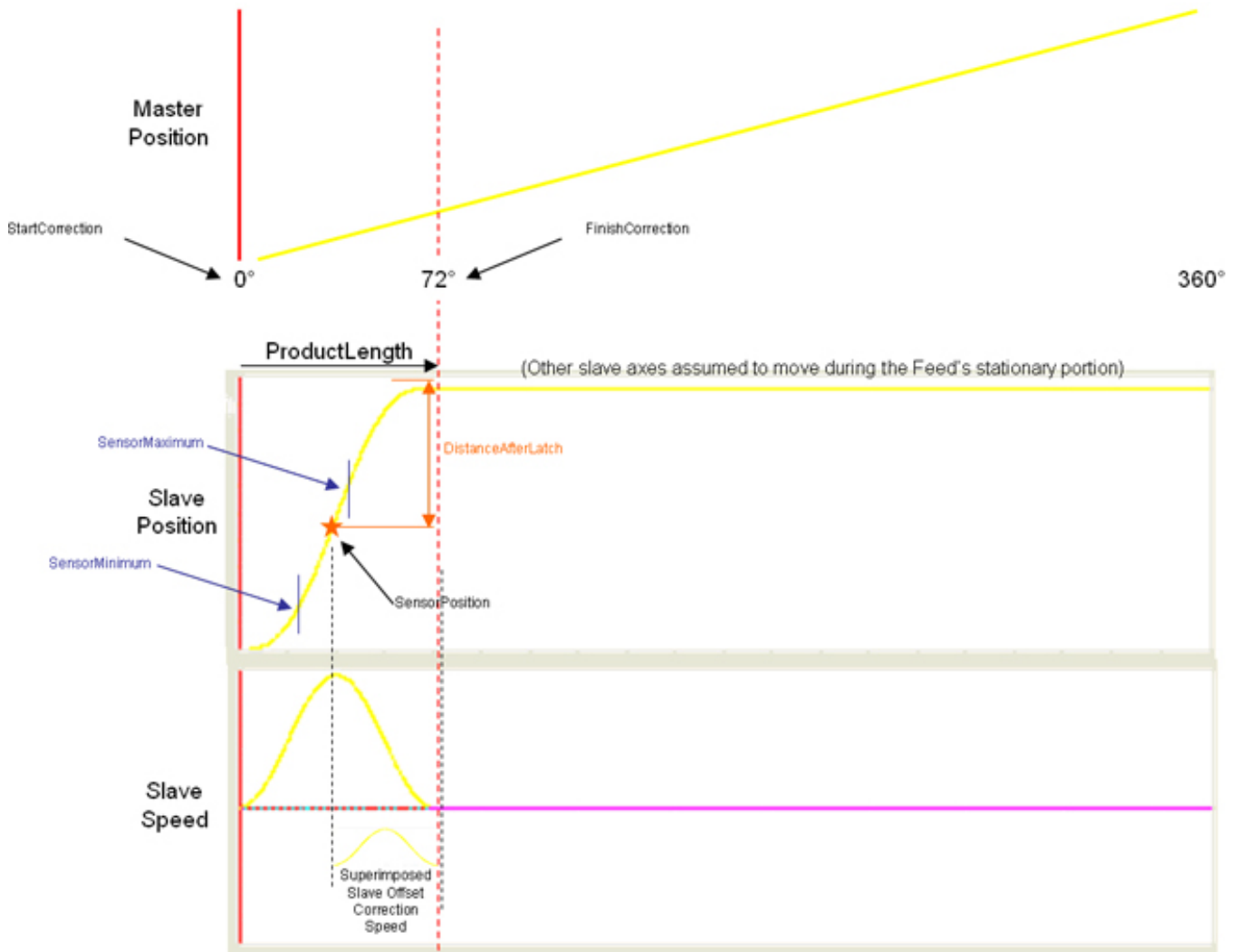
See the [Function Block ErrorID](#) list.

Applications

- Label Feeder
- Punch Press

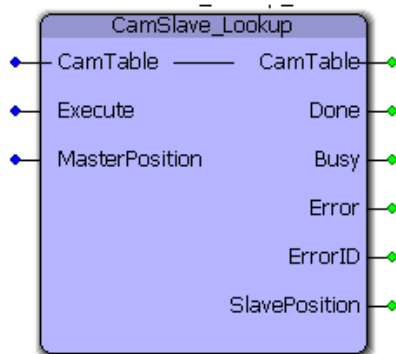
Overview of Supporting Function Blocks







CamSlave_Lookup



This function block returns the slave position corresponding to the given master position. This function block is used by [CamSlave_Recover](#).

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	CamTable	Y_MS_CAM_STRUCT	Cam data structure.
VAR_INPUT			Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
V	MasterPosition	LREAL	The position of the master axis for which the corresponding slave position is required.
VAR_OUTPUT			
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.

B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	SlavePosition	LREAL	The slave position that relates to the master as described in the CamTable.

Notes

This function provides the exact slave position that corresponds to the MasterPosition input by interpolating the CamTable. Consider the following CamTable:

M	S
0	0
10	0
20	5
30	10
40	20

If the MasterPosition is 15, the corresponding SlavePosition is 2.5. (50% of the value between two master points is used to determine the value 50% between the corresponding slave points.)

This function determines the equivalent slave position by looking in the CamTable only, It does not include any other cam adjustments that may have been applied using any of the Y_CamAdjust function blocks.

See the [CamSlave_Lookup eLearning Module](#) on Yaskawa's YouTube Channel.

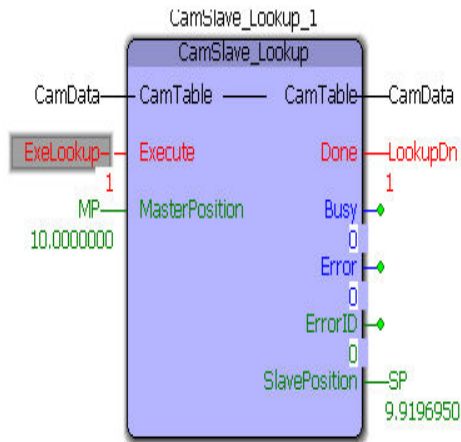
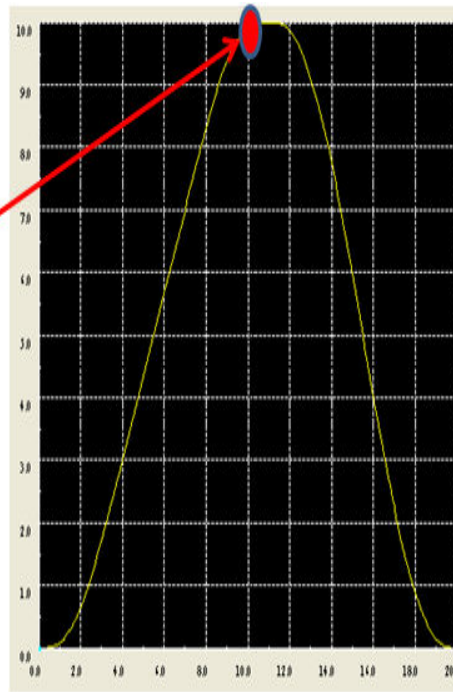
Error Description

See the [Function Block ErrorID](#) list.

Example

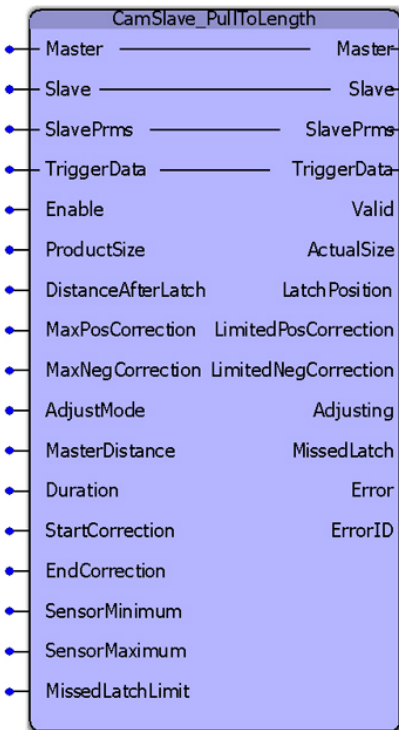
In the example shown below, the slave position corresponding to a master position of 10.0 is calculated. It can be seen that the slave position from the cam profile is 9.9196950.

[71]	Master	9.8000000
	Slave	9.8572890
[72]	Master	9.9000000
	Slave	9.8912510
[73]	Master	10.0000000
	Slave	9.9196950
[74]	Master	10.1000000
	Slave	9.9429420
[75]	Master	10.2000000
	Slave	9.9613810





CamSlave_PullToLength



CamSlave_PullToLength was designed for applications where the slave mechanism pulls material forward but the mechanism has a reciprocating stroke. This function block incorporates the ability to capture a registration mark on the material being pulled, and make on-the-fly adjustments to the stroke length by executing a Y_CamScale function block. This block has the same basic core operation as CamSlaveFeedToLength, which was designed for slaves that move in one direction but have the same requirement.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Master	AXIS_REF	A logical reference to the master axis.
B	Slave	AXIS_REF	A logical reference to the slave axis.
V	SlavePrms	AxisParameterStruct	User Defined DataType declared in the PLCopen Toolbox.

E	TriggerData	TRIGGER_REF	Reference to the trigger signal source.. Refer to PLCopen Plus Function Block Manual for more details.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	ProductSize	LREAL	This value must be the same as the total one way index of the cam profile for this slave.	LREAL#0.0
V	DistanceAfterLatch	LREAL	The desired additional travel distance after the registration mark is detected	LREAL#0.0
V	MaxPosCorrection	LREAL	Limits the amount of positive correction that can be applied.	LREAL#0.0
V	MaxNegCorrection	LREAL	Limits the amount of negative correction that can be applied.	LREAL#0.0
V	AdjustMode	INT	<p>An ENUM for TIME or range of master correction, with the following values:</p> <p>Y_AdjustMode#MasterDistance: The adjustment starts immediately and completes when the master has travelled the specified MasterDistance.</p> <p>Y_AdjustMode#ElapsedTime: The adjustment starts immediately and completes within the specified Time.</p> <p>Y_AdjustMode#WithinRange: The adjustment starts when the master first crosses the StartPosition and completes when the master reaches the EndPosition.</p>	<p>INT#0 (Y_AdjustMode#MasterDistance)</p>
V	MasterDistance	LREAL	Relative amount the master will travel (in cam master units) from when the function block first executes until the correction is complete. Only used if AdjustMode = Y_AdjustMode#MasterDistance.	LREAL#0.0
V	Duration	LREAL	Time of the correction used if AdjustMode is set for TIME mode	LREAL#0.0
V	StartCorrection	LREAL	Earliest master position where the correction can begin.	LREAL#0.0
V	FinishCorrection	LREAL	Latest master position where the correction must be completed.	LREAL#0.0
V	SensorMinimum	LREAL	The earliest slave position where a sensor position is valid for correction.	LREAL#0.0
V	SensorMaximum	LREAL	The latest slave position where a sensor position is valid for correction.	LREAL#0.0 (function block SensorMaxium to ProductSize if unconnected or set to zero.)

V	MissedLatchLimit	UINT	The number of consecutive DefaultDistances allowed to occur without seeing a registration mark in the window, and not cause an Error. Valid registration marks will reset the internal counter.	UINT#0 (interpreted as infinite)
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	ActualSize	LREAL	The actual indexed distance.	
V	LatchPosition	LREAL	The slave's position in the CamTable when the latch occurred.	
V	LimitedPosCorrection	BOOL	Indicates that the MaxPosCorrection is limiting the required correction.	
V	LimitedNegCorrection	BOOL	Indicates that the MaxNegCorrection is limiting the required correction.	
V	Adjusting	BOOL	Indicates that an adjustment is currently taking place (Busy output of Y_SlaveOffset)	
V	MissedLatch	BOOL	Indicates that a latch was detected, but it was outside of the window parameters specified.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

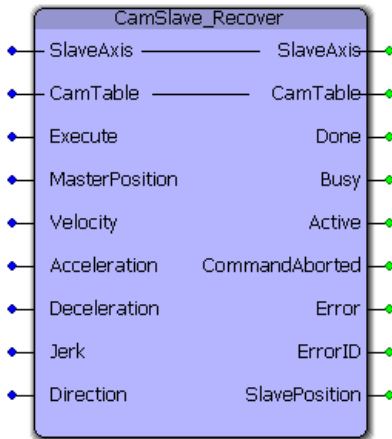
This function block is an adaptation of [CamSlave_FeedToLength](#). The main difference is that this function is designed for reciprocating slave motion, and uses the Y_CamScale function block instead of the Y_SlaveOffset function block.

Error Description

See the [Function Block ErrorID](#) list.



CamSlave_Recover



The CamSlave_Recover block moves a Slave back into sync with the master axis after camming was interrupted unexpectedly, such as E-Stop conditions, or alarms that disable the servo. This function block is particularly useful when resuming the cam motion from the position where it was interrupted is necessary to avoid wasting products in process, or if machine characteristics demand it, or if homing and re-starting the cycle is not feasible. The CamSlave_Recover function block can be used to bring the slave axis to the position in the cam table that corresponds to the current master axis position. Linear interpolation is performed for accuracy in case of coarse resolution between points in the cam table. Once CamSlave_Recover is Done, the camming motion can resume. This function block contains an MC_MoveAbsolute function.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	SlaveAxis	AXIS_REF	A logical reference to the slave axis.
B	CamTable	Y_MS_CAM_STRUCT	Cam data structure. Can be downloaded to the motion engine using Y_CamStructSelect.
VAR_INPUT			Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
			FALSE

B	MasterPosition	LREAL	Master axis' current position. The CamSlave_ Recover function block will command the slave axis to move to the slave position corresponding to this MasterPosition value.	LREAL#0.0
B	Velocity	LREAL	Velocity with which the slave axis recovers and moves to the position from the cam table corresponding to the master axis position	LREAL#0.0
B	Acceleration	LREAL	Acceleration with which the slave axis recovers and moves to the position from the cam table corresponding to the master axis position	LREAL#0.0
B	Deceleration	LREAL	Deceleration with which the slave axis recovers and moves to the position from the cam table corresponding to the master axis position	LREAL#0.0
E	Jerk	LREAL	<i>Not supported; reserved for future use. Value of the jerk in [user units / second^3].</i>	LREAL#0.0
B	Direction	MC_Direction	The position of the slave axis for which the corresponding master position is required.	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Active	BOOL	For buffered modes, this output is set high at the moment the block takes control of the axis. For non buffered modes, the outputs busy and active have the same value	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	SlavePosition	LREAL	Slave position in the cam profile that corresponds to the MasterPosition input to the function block	

Notes

After CamSlave_ Recover is Done, in most cases, the slave will be at a position different from the home position or dwell position. Care should be taken before re-engaging the slave to the master axis. Engage Position and Engage Data inputs on the Y_CamIn block should be verified to make sure that they are set correctly. Incorrect engage position and or engage method can cause abrupt motion on the slave axis.

Recommended steps to recover from a cam cycle interruption

- 1) Clear all alarms after an E-Stop.
- 2) Enable the slave.
- 3) Verify the MasterPosition input is the position of the master axis to where the slave must to move to re-synchronize the cam operation.
- 3) Execute CamSlave_ Recover with valid inputs.
- 4) Once CamSlave_ Recover.Done is TRUE, the slave is in position to continue the cam motion immediately.
- 5) Change the Y_CamIn.EngagePosition to the current master position. Set Y_CamIn.EngageData.SlaveAbsolute:= TRUE.
- 6) Execute Y_CamIn. The cam will engage and when the master axis starts motion, the slave will move in synchronization with the master.

See the [CamSlave_ Recover eLearning Module](#) on Yaskawa's YouTube Channel.

Error Description

See the [Function Block ErrorID](#) list.

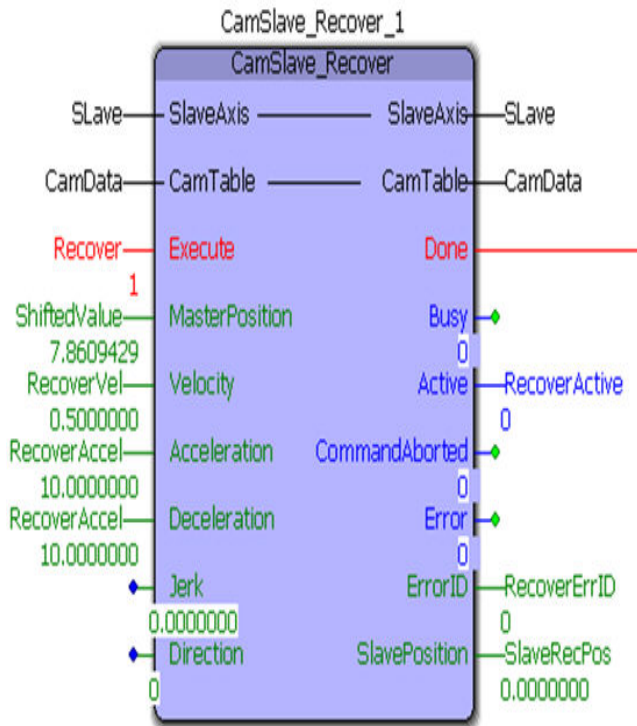
Example

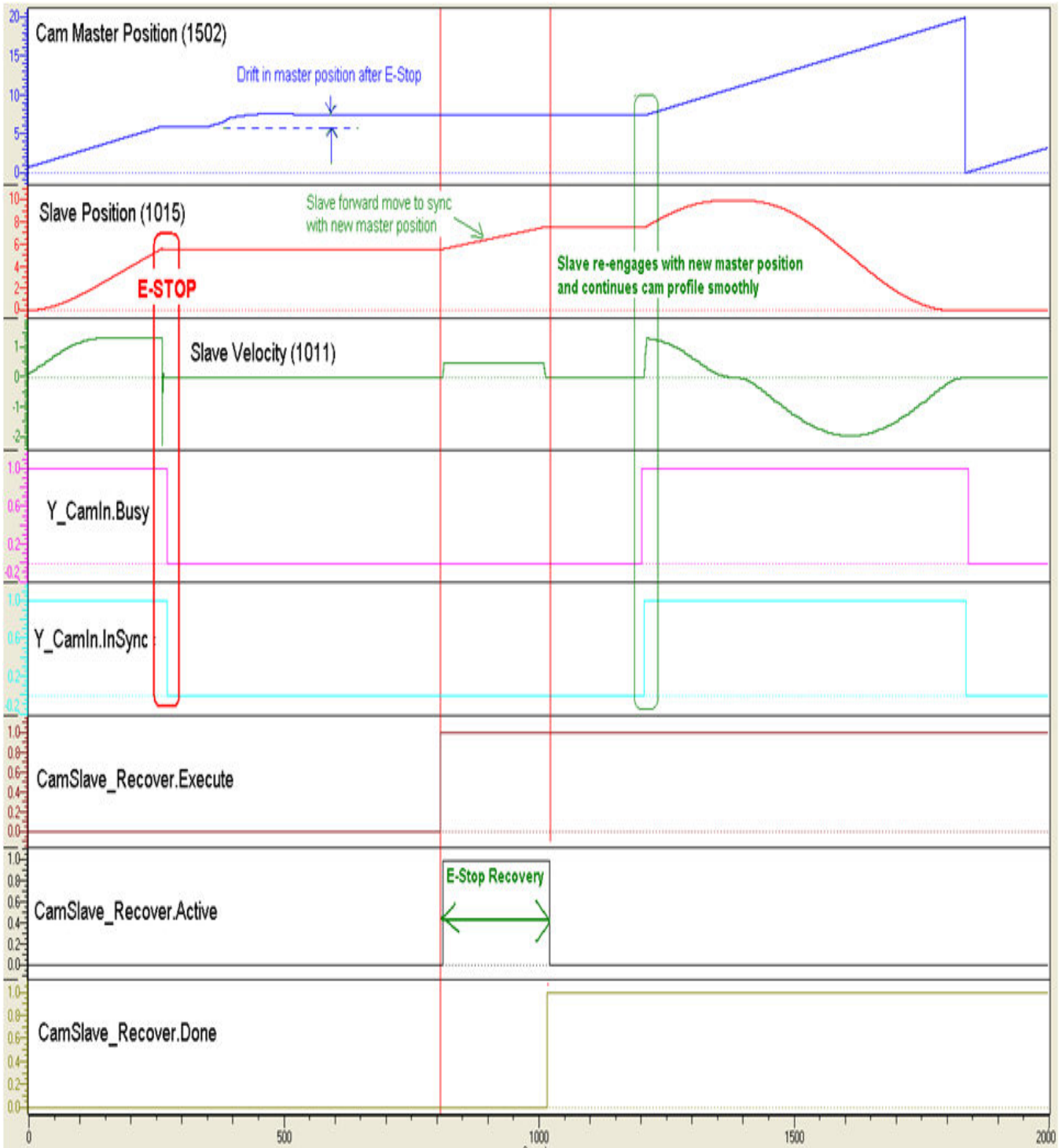
E-Stops can result in the instantaneous loss of control of the axes. Manually clearing debris or scrap from the machine and adjustments after E-Stops and alarms can cause a change in motor position, all resulting in a de synchronization of the master and slave.

The example given below illustrates how the CamSlave_Recover block can solve E-Stop recovery issues. The logic analyzer plot shows the axes when the E-Stop occurred. At this point, the Y_CamIn outputs InSync and Busy change to FALSE. A slight drift in the master axis position can be seen after the E-Stop. This can be due to axis inertia, or because of adjustments made to the machine. The CamSlave_Recover block is executed to physically move the slave to the position that corresponds to the master's current position as determined by looking in the CamTable.

The distance that the slave axis traverses in this process can be seen in the illustration. Once the CamSlave_Recover is Done, the slave can be re-engaged with the master using Y_Camin.

Important: In this recovery condition, the 'EngagePosition' must be set to the master axis' current position and the EngageData.SlaveAbsolute=TRUE must be applied.

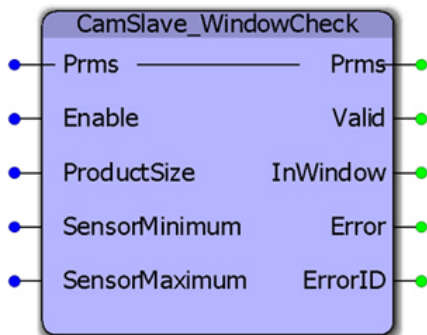




[41]	Master	5.0000000	
	Slave	4.3333330	E-Stop
[42]	Master	6.0000000	Master: 5.97
	Slave	5.6666670	Slave : 5.61
[43]	Master	7.0000000	
	Slave	7.0000000	
[44]	Master	7.1000000	
	Slave	7.1333060	
[45]	Master	7.2000000	
	Slave	7.2664440	
[46]	Master	7.3000000	
	Slave	7.3992430	
[47]	Master	7.4000000	
	Slave	7.5315240	
[48]	Master	7.5000000	
	Slave	7.6630960	
[49]	Master	7.6000000	
	Slave	7.7937530	
[50]	Master	7.7000000	
	Slave	7.9232730	
[51]	Master	7.8000000	After recovery
	Slave	8.0514140	
[52]	Master	7.9000000	Master: 7.8609
	Slave	8.1779140	Slave : 8.1285
[53]	Master	8.0000000	
	Slave	8.3024920	



CamSlave_WindowCheck



This function block is used by the CamSlave_FeedToLength function blocks to determine when the MC_TouchProbe output is valid and should be used for correction. It compares the CamTableOutput parameter 1520 to the SensorMinimum and SensorMaximum, not the actual slave feedback.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Prms	AxisParameterStruct	User Defined DataType declared in the PLCopen Toolbox.	
VAR_INPUT			Default	
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	ProductSize	LREAL	This value must be the same as the total one way index of the cam profile for this slave.	LREAL#0.0
V	SensorMinimum	LREAL	The earliest slave position where a sensor position is valid for correction.	LREAL#0.0
V	SensorMaximum	LREAL	The latest slave position where a sensor position is valid for correction.	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	InWindow	BOOL	Indicates the slave output.	

B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

If SensorMinimum and SensorMaximum are both zero, this function does not check for a window and reports InWindow as TRUE.

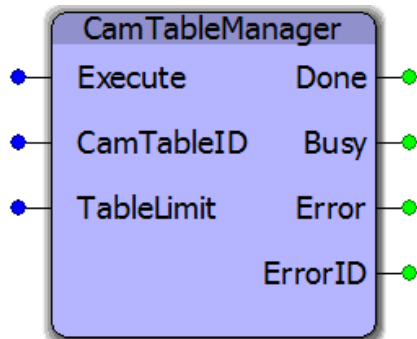
For the most accurate WindowCheck, this function block must be executed in a fast CYCLIC task. Since this function is used by CamSlave_WindowCheck, that block also should be executed in a fast CYCLIC task.

Error Description

See the [Function Block ErrorID](#) list.



CamTableManager



This function block serves as a FIFO buffer for CamTableIDs. Each time a new CamTableID is created, it will delete the cam memory allocated to the oldest CamTable by using the Y_RemoveCamTable function block from the PLCopenPlus firmware library. This function block cleans up cam memory for applications which routinely build new cam tables on the fly. A circular buffer of four cam tables is maintained in the CamTableManager. When the function block is executed a fifth time, it releases the memory area of the oldest cam table ID. Memory managed is that of the motion engine, not the PLC application.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	CamTableID	UINT	The most recent CamTableID create by Y_CamFileSelect or Y_CamStructSelect.	UINT # 0
V	TableLimit	UINT	The number of cam tables to leave in memory before they are removed.	UINT # 5
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	

B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

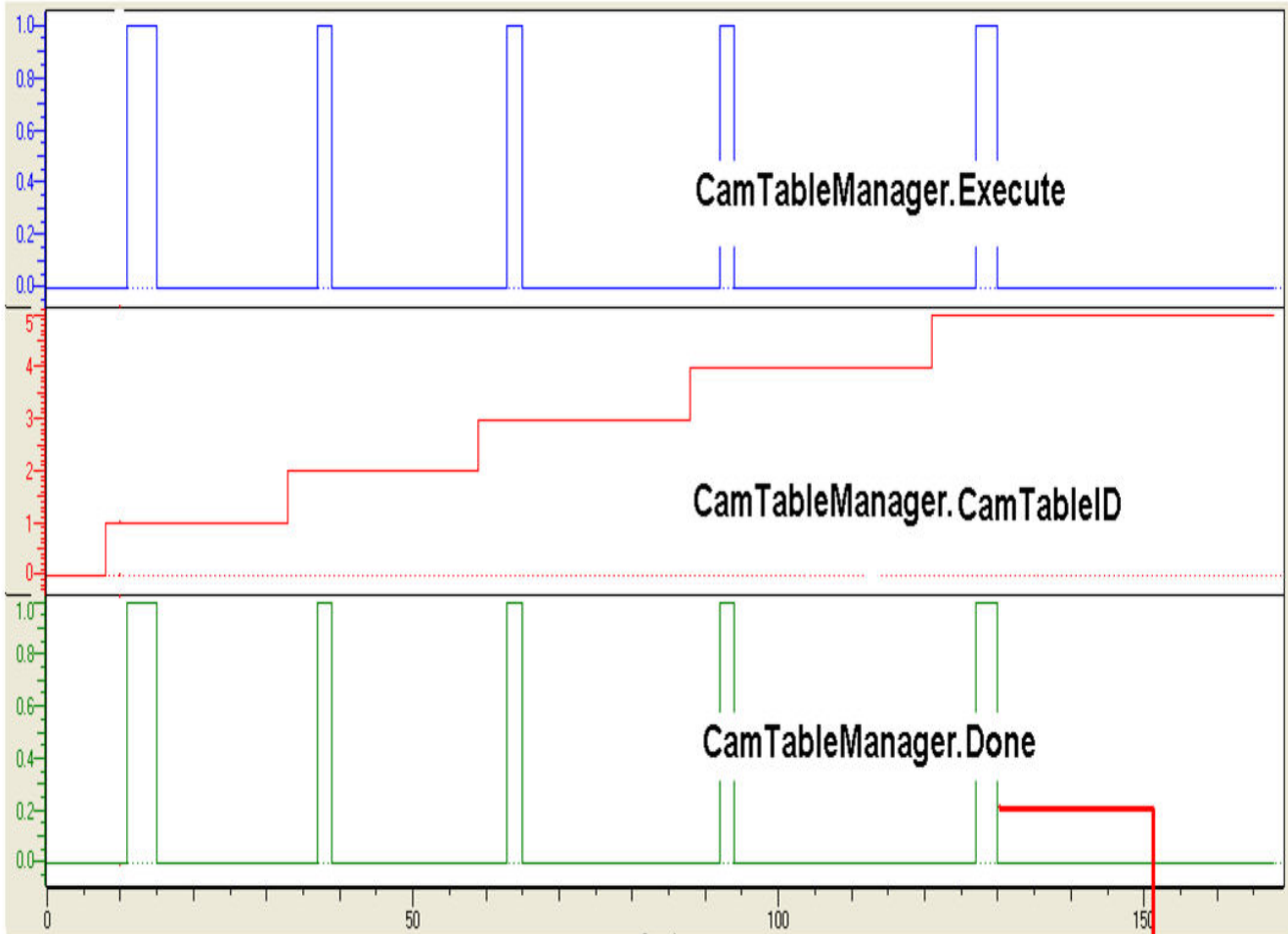
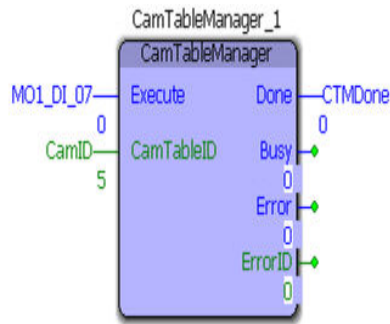
- This function block is unnecessary in applications which use a single, static cam table, or when there are several cams but once created, they are used permanently and not recalculated. There is capacity in the controller memory for dozens of cam tables. CamTableManager prevents the situation where a machine has been continuously running for weeks and enough cams were created that the memory becomes exhausted.
- Even though the memory for cam tables has been released, the Y_CamStructSelect function block will continue to allocate a new (increasing) CamTableID.
- See the [CamTableManager eLearning Module](#) on Yaskawa's YouTube Channel.

Error Description

See the [Function Block ErrorID](#) list.

Example 1

An example of using the CamTableManager is shown below. On the fifth execute of the CamTableManager block, the memory for the oldest CamTable ID gets released. In the example shown below, the memory for CamID 1 gets released. The next execution of the CamTableManager will release the memory for CamID 2.



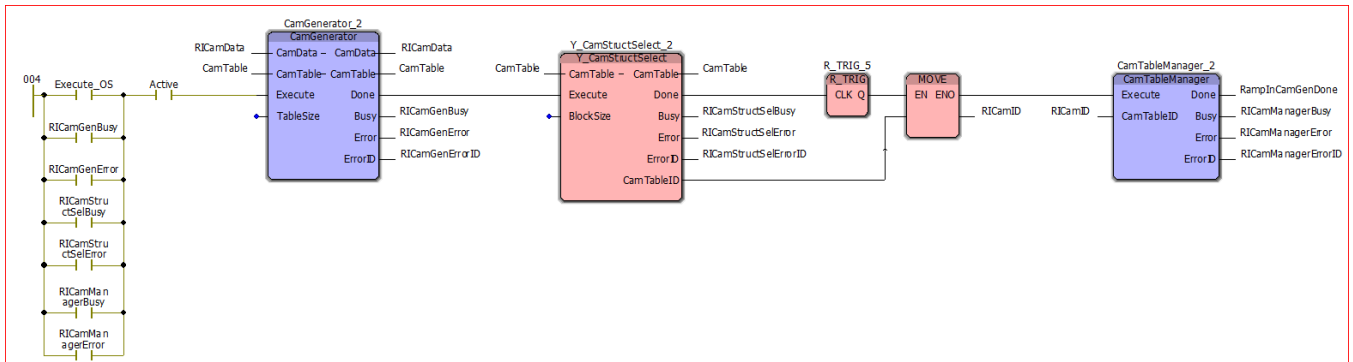
Watch Window

Variable	Value	De
CamTables		
[0]	1	
[1]	2	
[2]	3	
[3]	4	
[4]	5	

Watch 1 Watch 2 Watch 3 Watch 4

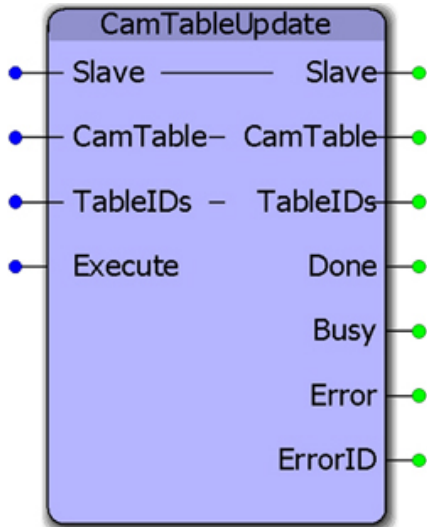
Memory for Cam ID 1 gets cleared

Application Example





CamTableUpdate



This function block aids with cam file management when on the fly changes to the table data are required. It supports two tables: one which may be actively running in the motion engine, and one that may be recalculated and transferred to the motion engine. It contains the Y_CamStructSelect and Y_WriteCamTable function blocks.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Slave	AXIS_REF	A logical reference to the slave axis.	
B	CamTable	Y_MS_CAM_STRUCT	Cam data structure. Can be downloaded to the motion engine using Y_CamStructSelect.	
V	TableIDs	TableIDStruct	Contains an Active and Inactive TableID	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				

B	Done	BOOL	Set high when the commanded action has been completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	BOOL	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

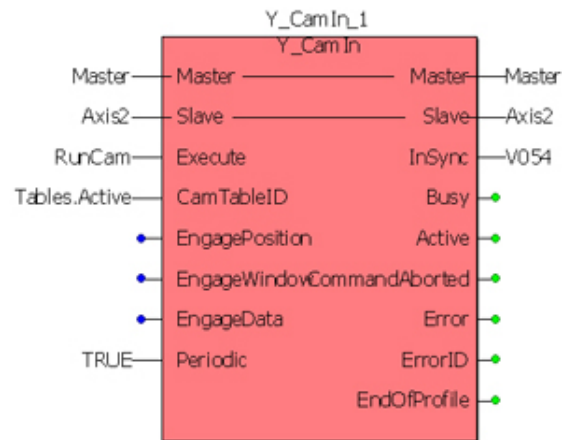
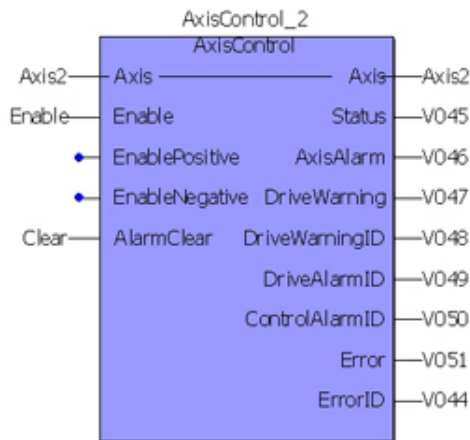
- If both TableIDs in the TableIDs input are zero, then this block automatically uses Y_CamStructSelect to send the first CamTable and obtain the CamTableID.
- If the event causing the cam tables to update is fired too frequently, this block limits the cam table transfer and swap by holding in a Busy state while the previous table transferred is still waiting to become the active table. In this way, it helps to stage the table swapping so that the application does not resort to writing over an active table, which can cause the slave to jump.

Error Description

See the [Function Block ErrorID](#) list.

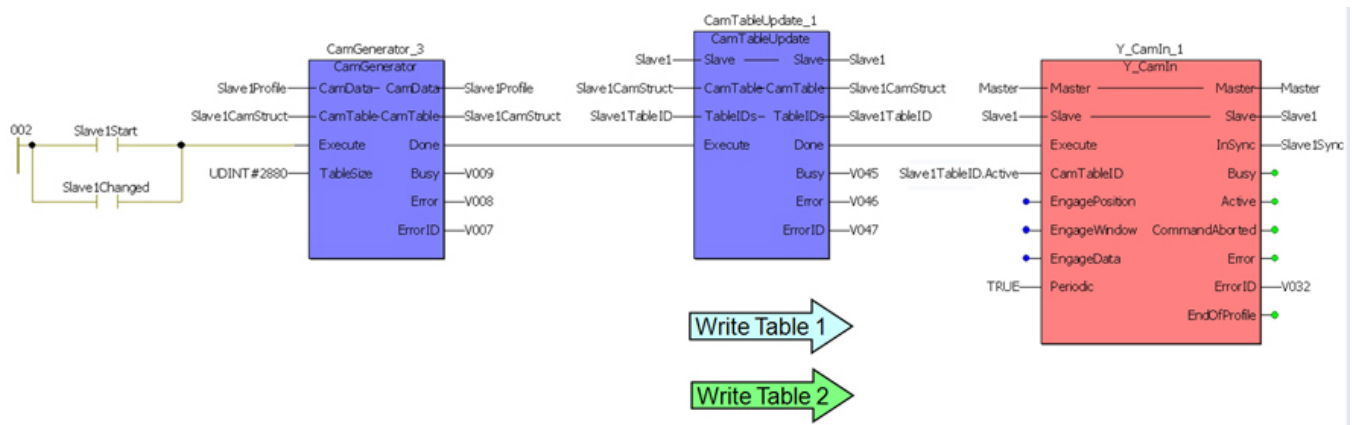
Example 1:

In this example, assume that some event has occurred which triggers the need for a new cam table to be generated using CamGenerator. CamGenerator in turn fires CamTableUpdate to send the new CamTable to the motion engine. CamTableUpdate manages the active and inactive TableIDs, which can then be used with Y_CamIn. The Table.Active variable will contain the TableID of the last table transferred, so the next time the rising edge of Y_CamIn is triggered, the new table will be used. This can be done while camming is currently engaged.



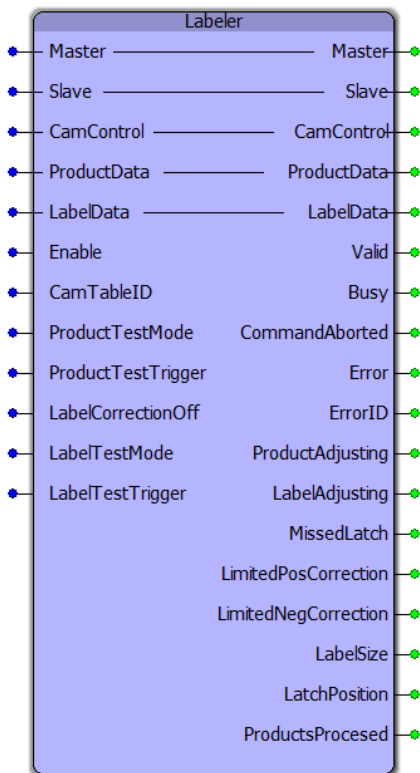
Example 2: Using Two Cam Tables

- One will be actively running the motion
- One will be "on deck" to take new changes





Labeler



The Labeler function block is designed to manage the entire automatic process for a labeling application. Products typically arrive at random intervals, so support is included for registration mark identification on the external axis, typically a conveyor. Registration mark adjustment on the slave (label) axis is also supported to account for potential slip or drift due to slight inaccuracies in the label lengths. In addition to this function block, the user must add logic to enable the servo, and any manual mode functionality required.

Library

Cam Toolbox

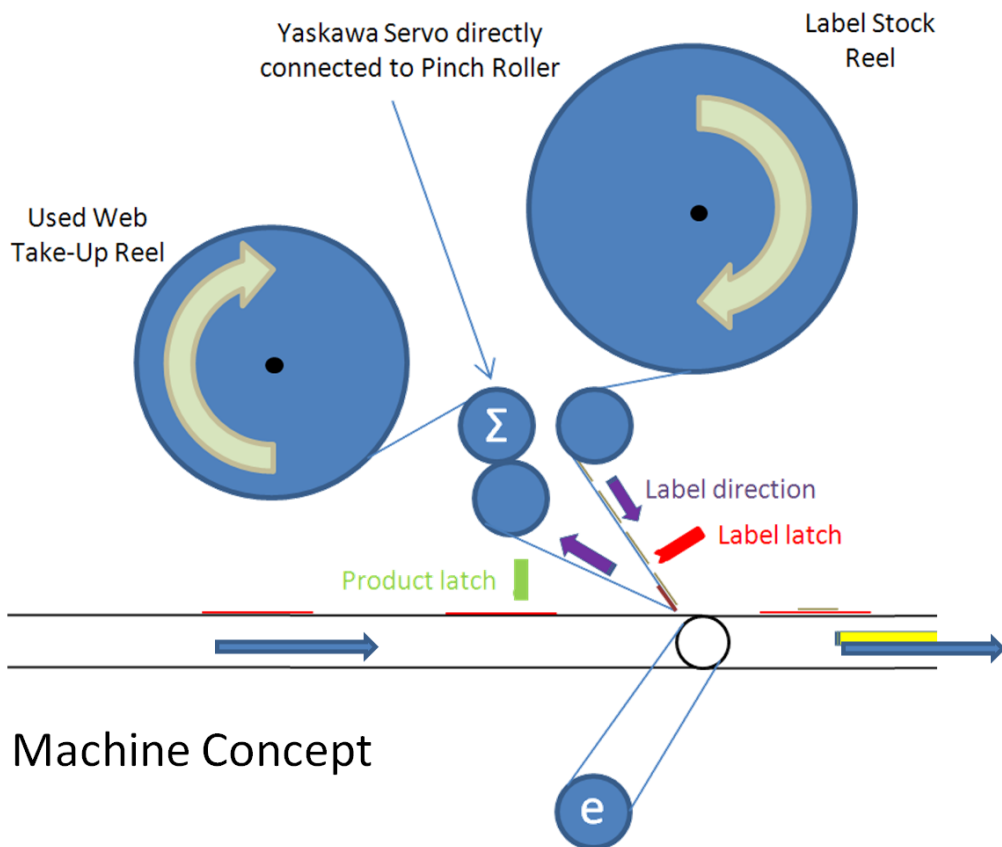
Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Master	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
B	Slave	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).

V	CamControl	CamSyncStruct	Structure containing all information to allow both the CamControl and CamShiftControl to make decisions to run the cam function effectively.
V	ProductData	ProductBufferStruct	Structure containing all information for the circular buffer to operate.
V	LabelData	LabelStruct	Structure containing all information to allow both the CamControl and CamShiftControl to make decisions to run the cam function effectively.
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
B	CamTableID	UINT	A reference to the cam memory in the motion engine. The cam table must be generated separately by the user.
V	ProductTestMode	BOOL	Set TRUE to manually provide "ProductTestTrigger"
V	ProductTestTrigger	BOOL	When ProductTestMode is TRUE, when this input transitions to TRUE, a simulated product latch is recorded.
V	LabelCorrectionOff	BOOL	When set TRUE, it disables all Label axis registration activities.
V	LabelTestMode	BOOL	Set TRUE to manually provide "LabelTestTrigger"
V	LabelTestTrigger	BOOL	When LabelTestMode is TRUE, when this input transitions to TRUE, a simulated Label latch is recorded.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
E	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	ProductAdjusting	BOOL	This output is set TRUE during the time when an adjustment is being made on the master axis. The product and the label are not in synchronization during this time.
V	LabelAdjusting	BOOL	This output is set TRUE during the time when an adjustment is being made on the Label axis. The product and the label are not in synchronization during this time.
V	MissedLatch	BOOL	This output will pulse if the master axis moves more than 110% of the ProductSize without detecting a registration mark. Use an R_TRIG and an ADD function block (or the Logic Analyzer) to monitor this output.

V	LimitedPosCorrection	BOOL	This output will pulse if the calculated correction required is larger than LabelData.MaxPosCorrection. For a properly configured system, this occurrence will be rare. Use an R_TRIG and an ADD function block (or the Logic Analyzer) to monitor this output.
V	LimitedNegCorrection	BOOL	This output will pulse if the calculated correction required is larger than LabelData.MaxNegCorrection. For a properly configured system, this occurrence will be rare. Use an R_TRIG and an ADD function block (or the Logic Analyzer) to monitor this output.
V	LabelSize	LREAL	Calculated as the position the label registration mark occurred in the cam table + DistanceAfterLatch.
V	LatchPosition	LREAL	The label's registration mark position from the start of the cam table.
V	ProductsProcessed	UDINT	A count of the number of items processed by this function block since Enable was set TRUE.

Typical Labeler Machine Concept



Machine Concept

Notes

- This function block requires a significant amount of parameter data (VAR_IN_OUT structures) to be accurately initialized for proper operation. Please study the structure datatypes connected to this function block and verify all necessary information.
- A Cam table to be used by this function block must be generated. Use the MotionWorks IEC Cam Editor or the [CamGenerator](#) function block to create this table.
- See additional Labeler Application Note on www.yaskawa.com for tips on wiring the registration sensors, and using the MotionWorks IEC Cam Editor to create the necessary cam table for the Labeler.

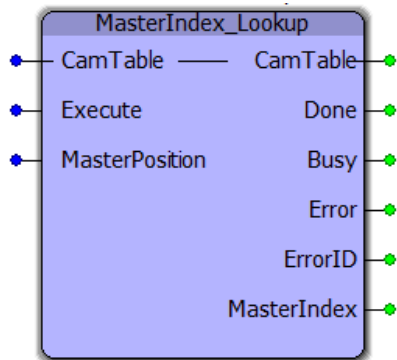
Error Description

See the [Function Block ErrorID](#) list.

Example



MasterIndex_Lookup



This function block returns the array index value corresponding to the given master position.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	CamTable	Y_MS CAM STRUCT	Cam data structure. Can be downloaded to the motion engine using Y_CamStructSelect.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	MasterPosition	LREAL	The position of the master axis for which the index in the cam table is required.	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	

B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	MasterIndex	UDINT	The array index corresponding to the master axis position in the Y_MS_CAM_STRUCT structure.

Notes

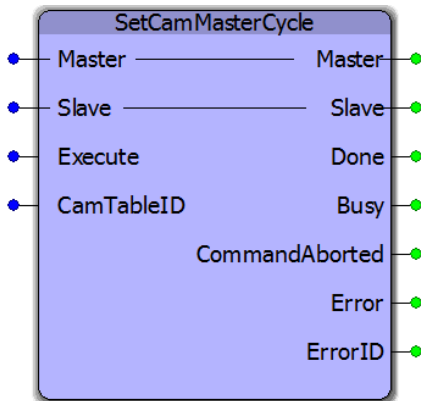
- The MasterPosition input should be a value between the maximum and minimum values of the master's position profile for the index value to be valid.

Error Description

See the [Function Block ErrorID](#) list.



SetCamMasterCycle



This function block prepares the motion engine with the cam table data so that cam adjustments involving blocks like Y_CamShift can be executed before Y_CamIn is executed. This is necessary for applications where calculations that involve the cam master cycle (parameter 1512) or cam master shifted cyclic position (parameter 1502) must be made before the Y_CamIn function block is executed.

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Master	AXIS_REF	A logical reference to the master axis.	
B	Slave	AXIS_REF	A logical reference to the slave axis.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
B	CamTableID	UINT	A reference to the cam memory in the motion engine.	UINT#0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	

B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

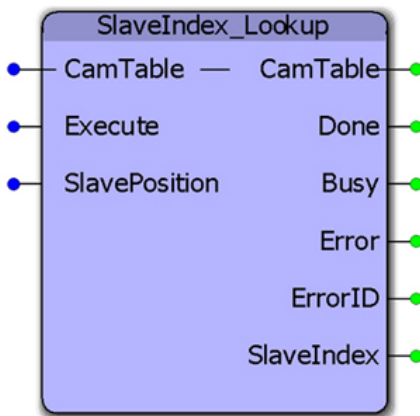
- 1) Although there will be no slave motion, the slave axis must be enabled using MC_Power before executing this function block.
- 2) A valid CamTableID must be input before executing SetCamMastercycle.

Error Description

See the [Function Block ErrorID](#) list.



SlaveIndex_Lookup



This function block returns the array index value corresponding to the given slave position. This function block is used by [CamMasterLookup](#) to determine the equivalent master location for a given slave position.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	CamTable	Y_MS_CAM_STRUCT	Cam data structure. Can be downloaded to the motion engine using Y_CamStructSelect.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	SlavePosition	LREAL	The position of the slave axis for which the corresponding master position is required.	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	

B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	SlaveIndex	UDINT	The array index of the Y_MS_CAM_STRUCT of the SlavePosition.

Notes

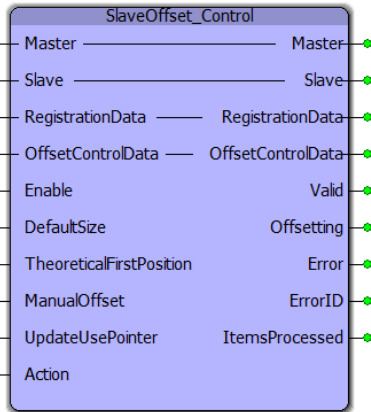
- The SlavePosition input should be a value between the maximum and minimum values of the slave's position profile for the index value to be valid.
- If the SlavePosition input is a value between two slave positions in the cam table, the SlaveIndex will return the lower index.

Error Description

See the [Function Block ErrorID](#) list.



SlaveOffset_Control



This function block makes corrections on a cammed slave axis for applications that require adjustments while the axis is in motion. Some applications, such as labeling, require on the fly adjustments based on sensor input that occurs while the label is moving. The actual pitch between consecutive labels may be different from the nominal pitch. The correction amount is the difference between the nominal part size and the actual part size measured by the sensor. In this type of application, the sensor input is wired to the slave axis.

The SlaveOffset_Control block is similar to [CamSlave_FeedToLength](#) in functionality. Both function blocks make corrections on a cammed slave axis based on sensor input. Both function blocks make corrections while the slave axis is in motion. The difference between the two blocks is that SlaveOffset_Control makes use of the [ProductBuffer](#) block while the [CamSlave_FeedToLength](#) does not. This allows the SlaveOffset_Control to buffer latched registration data. This can be used in applications where the sensor is more than one part length away from the point of action (SensorDistance > 1 part length). SlaveOffset_Control lacks the window check feature, correction limit feature and missed latch limit feature available in the [CamSlave_FeedToLength](#) block.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Master	AXIS_REF	A logical reference to the master axis
B	Slave	AXIS_REF	A logical reference to the slave axis
V	RegistrationData	ProductBufferStruct	Structure containing all information for the circular buffer to operate.

V	OffsetControlData	SlaveOffsetStruct	Structure containing all information to calculate and implement slave offsets.
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	DefaultSize	LREAL	Default pitch between consecutive parts
V	TheoreticalFirstPosition	LREAL	Ideal absolute position of the first part that will be detected by the sensor after homing is done.
V	ManualOffset	LREAL	One time adjustment that can be made while the slave is in motion. A change in the manual offset value will trigger the offset value to be added to the calculated correction.
V	UpdateUsePointer	BOOL	RegistrationData.UsePointer will be updated when a product has been processed only if this input is TRUE. If more than one slave follows the same master as a parallel activity, only one instance of this function block must update the UsePointer.
V	Action	INT	Designates this instance of this function block as one of the several activities to occur based on the registration sensor. For applications that have only one action, such as a cut or a stamp, this input can be left unconnected. This input is required for applications that have more than one action associated with a sensor input, such as pick and place.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	Offsetting	BOOL	Set high if the function block is active and Y_SlaveOffset is Busy.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	ItemsProcessed	UDINT	Provides a count of the number of products processed since this function was enabled.

Notes

- This function block includes the Y_SlaveOffset function block, and will execute Offsets at the appropriate position based on data provided by the user via the [SlaveOffsetStruct](#) structure.
- In cases where multiple slaves are synchronized to a single master, the slaves can share the same ProductBuffer . Set the last slave (last SlaveOffset_Control function block) to update the UsePointer for the ProductBuffer.
- SlaveOffset_Control provides similar functionality as [CamSlave_FeedToLength](#) as summarized in the table shown below.

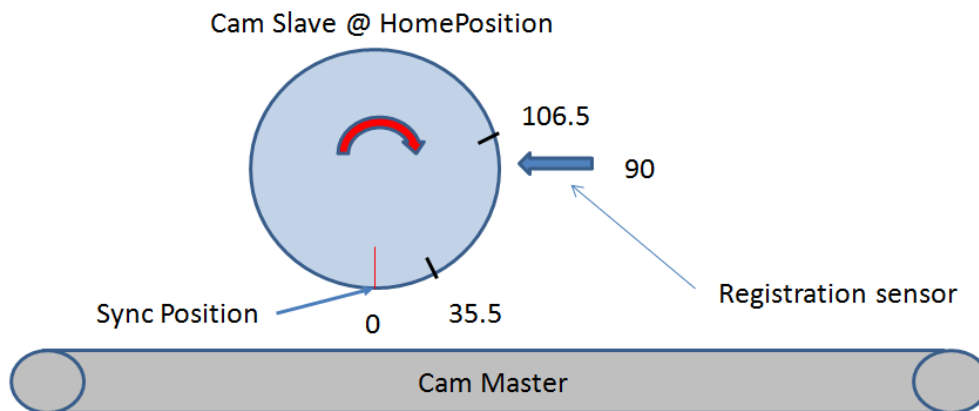
	CamSlave_FeedToLength	SlaveOffset_Control
BufferProducts (SensorDistance > 1 part length)	Not supported	Supported
Successive triggers > 2 part lengths	Reports missed latch	Makes large correction
Missed Part Indicator	Supported	Not supported
Registration within window check	Supported	Not supported
Correction limits	Supported	Not Supported

Error Description

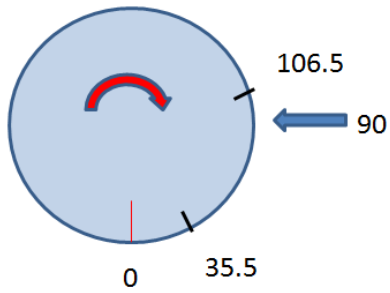
See the [Function Block ErrorID](#) list.

Application Example 1

Consider a rotary disc cammed to a master conveyor running a one way cam in the clockwise direction as shown. The home position is defined as shown below. Product length (nominal distance between parts) on the slave is 71 degrees. The nominal cam slave travel is 71 degrees. Position 0 (bottom dead center) is the position that needs to be synchronized with the master. The sensor distance is 90 units. If the first part is at 35.5 units and if the parts are exactly at the specified nominal lengths, then the second part (first part captured by the sensor) will be at 106.5. However, the actual registered position of this part may not be 106.5. In this case, the second product will not get synchronized with the master if it runs the nominal cam of 71 units. If the second product were at 105.5 units, an offset of 1 unit will have to be made for the synchronization to be effective.



The figure below shows how to configure the SlaveOffsetStruct.

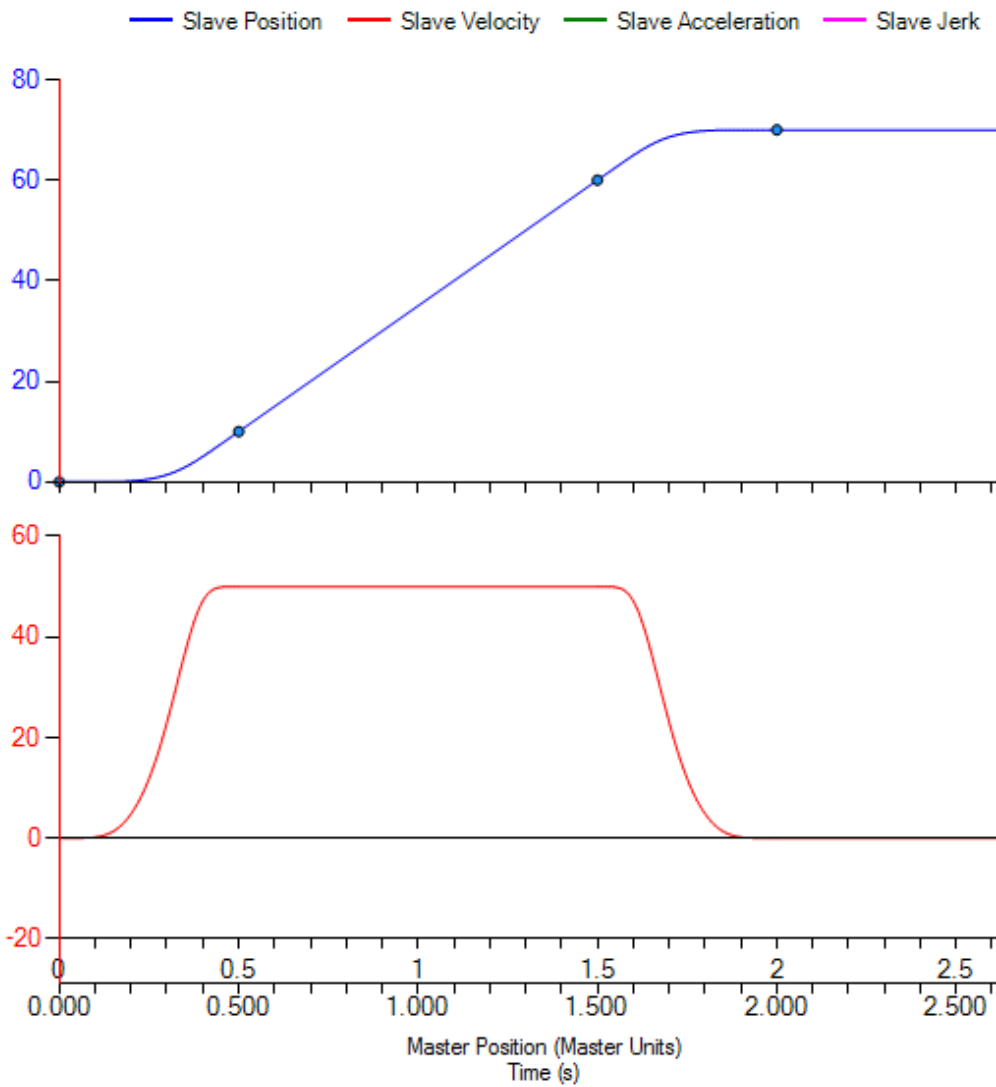


```

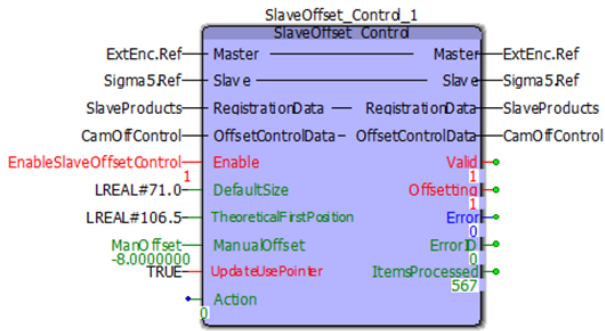
20 SlaveProducts.BufferSize := INT#20;
1 SlaveProducts.Sensor.Bit := UINT#1;
72.0000000 SlaveProducts.ProductAwayDistance := LREAL#72.0;
90.0000000 SlaveProducts.SensorDistance := LREAL#90.0;

0.3000000 CamOffControl.StartSyncPosition := LREAL#0.3;
0.1000000 CamOffControl.EndSyncPosition := LREAL#0.1;

```

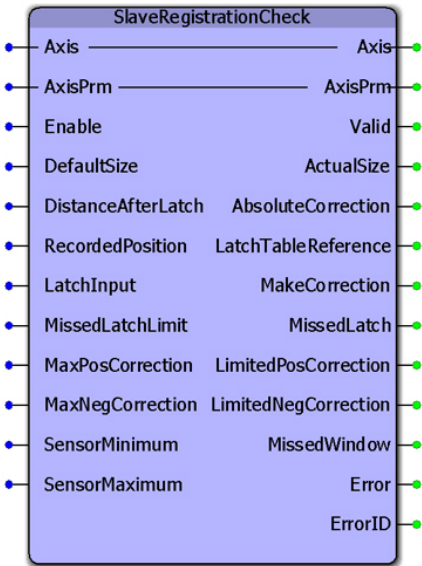


The SlaveOffset_Control block for the application described above can be set up as shown below.





SlaveRegistrationCheck



This function block was designed for use by the [CamSlave_FeedToLength](#), [CamSlave_FeedToLength2](#), and [CamSlave_PullToLength](#) function blocks. It monitors variables related to a cam slave index and fires the output "MakeCorrection" which can be connected to Y_SlaveOffset along with the AbsoluteCorrection output. The function also provides the interpolated value of the cam table output when the latch was detected.

Library

Cam Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
B	AxisPrm	AxisParameterStruct	Structure containing all parameters available for the Slave. Populate this structure using the ReadAxisParameters function block.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	DefaultSize	LREAL	Default length of the product in user units.	LREAL#0.0

V	DistanceAfterLatch	LREAL	The desired additional travel distance after the registration mark is detected	LREAL#0.0
B	RecordedPosition	LREAL	Position where trigger event occurred in user units. In accordance with PLCopen, this output is only valid when the Done output is high.	LREAL#0.0
V	LatchInput	BOOL	Typically connected to MC_TouchProbe.Done, signals the function to calculate any required correction amount.	FALSE
V	MissedLatchLimit	UINT	The number of consecutive DefaultDistances allowed to occur without seeing a registration mark in the window, and not cause an Error. Valid registration marks will reset the internal counter.	UINT#0 (interpreted as infinite)
V	MaxPosCorrection	LREAL	Limits the amount of positive correction that can be applied.	LREAL#0.0
V	MaxNegCorrection	LREAL	Limits the amount of negative correction that can be applied.	LREAL#0.0
V	SensorMinimum	LREAL	The earliest slave position where a sensor position is valid for correction.	LREAL#0.0
V	SensorMaximum	LREAL	The latest slave position where a sensor position is valid for correction.	LREAL#0.0 (function block sets SensorMaximum to ProductSize if unconnected or set to zero.)
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	ActualSize	LREAL	The actual indexed distance.	
V	AbsoluteCorrection	LREAL	The absolute value of the slave offset for use with Y_SlaveOffset.	
V	LatchTableReference	LREAL	The position of the latch corresponding to the cam table.	
V	MakeCorrection	BOOL	Used to signal that the correction calculation is valid. Typically used in conjunction with Y_SlaveOffset.Execute. Note: this output will pulse for one scan.	
V	MissedLatch	UDINT	Flag which indicates that the controller did not find a valid registration mark within the SensorMinimum and SensorMaximum positions.	
V	LimitedPosCorrection	BOOL	Indicates that the MaxPosCorrection is limiting the required correction.	
V	LimitedNegCorrection	BOOL	Indicates that the MaxNegCorrection is limiting the required correction.	
V	MissedWindow	BOOL	Indicates that a latch occurred, but was ignored because it was outside the range of SensorMinimum and SensorMaximum.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

- This function block determines where in the cam profile the latch occurred and compares it to the expected location to make a determination about the correction required.
- This function block also monitors the travel distance of the slave, and if the slave traveled 10% more than the ProductDistance and no valid latch was detected, a missed mark is counted. If the number of consecutive missed marks equals the MissedLatchLimit input variable, ErrorID UINT#10021 is output.
- Set MissedLatchLimit:=0 to disable monitoring for missed latches.
- Separate correction limits are provided for positive and negative to account for applications where it is not possible to make such corrections. For example, negative corrections typically cannot be applied to labeling applications because the material will become loose (slack).

Error Description

See the [Function Block ErrorID](#) list.



Cam Curve Characteristics

Cam Curve does not mean a shape curve which expresses a cam profile, but rather a "motion curve" of the follower moved by the cam. A motion curve is generally shown with time on the horizontal axis and displacement on the vertical axis. The purpose of a cam is to move an object smoothly in a minimum time, without vibration and with minimum power. For this purpose, various motion curves have been developed. These curves are not only used for cam mechanisms but can also be applied to various other motions. The maximum non dimensional values such as V_m , A_m , and J_m are called the characteristic values of the cam curve. From these characteristic values and from the shapes of the acceleration curves, the general properties of the cam curves can be known.

Curve Selection

The procedure for selecting a curve is as follows:

1. Velocity V and Acceleration A are to be continuous
2. Low values of V_m and Q_m are needed in low speed and heavy load applications.
3. Low values of A_m and J_m are needed in high speed and light load applications.
4. Asymmetrical curve having the longer period of deceleration than acceleration should be used for situations when positioning accuracy is critical and residual vibration must be avoided.
5. A one-dwell curve should be used when the motion has no stop at the endpoint and must return immediately.
6. Select a curve from the modified constant velocity group when constant velocity is required in the middle part of the stroke.
7. Select a curve from the modified trapezoid group when acceleration is to be minimized.
8. The modified sine curve is recommended if there are no limitations.

Cam Curve Characteristics							
Sorted by velocity							
Curve	Velocity Max	Accel Max	Accel Min	Jerk Max	Jerk Min	Inertia Torque Max	Comment
No Dwell Modified Constant Velocity	1.22	7.68	7.68	48.20	-48.20	4.69	Lowest Velocity
Modified Constant Velocity	1.28	8.01	8.01	201.40	-67.10	5.73	Highest Accel
Simple Harmonic	1.57	4.93	4.93	∞	-15.50	3.88	Lowest Inertial Torque
One Dwell Modified Sine	1.66	5.21	5.21	65.50	-21.80	4.86	
One Dwell Cycloidal (m=2/3)	1.72	6.75	-4.50	53.00	-53.00	7.53	
No Dwell Modified Trapezoid	1.72	4.20	4.20	26.40	-26.40	5.07	Lowest Jerk
One Dwell Trapecloid	1.74	4.91	4.91	61.70	-61.70	6.86	
Modified Sine	1.76	5.53	-5.53	69.50	-23.20	5.46	
One Dwell Cycloidal (m=1)	1.76	5.53	5.53	34.70	-34.70	6.32	
NC2 Curve	1.79	5.89	-4.21	∞	-111.10	8.87	
One Dwell Modified Trapezoid (m=1)	1.92	4.44	-4.44	55.80	-55.80	7.11	
One Dwell Modified Trapezoid (Ferguson)	1.92	4.68	-4.22	58.90	-58.90	7.43	
One Dwell Modified Trapezoid (m=2/3)	1.94	5.53	-3.68	69.40	-69.40	8.63	
Parabolic	2.00	4.00	-4.00	∞	∞	8.00	Lowest Accel, Highest Jerk
Cycloidal	2.00	6.28	6.28	39.50	-39.50	8.16	
Modified Trapezoid	2.00	4.89	4.89	61.40	-61.40	8.09	
Asymmetrical Cycloidal	2.00	7.85	-5.24	61.70	-61.70	10.20	
Asymmetrical Modified Trapezoid	2.00	6.11	-4.07	96.00	-96.00	10.11	
Trapecloid	2.18	6.17	6.17	77.50	-77.50	10.84	Highest Velocity, Highest Inertial Torque

Cam Curve Characteristics

Sorted by Positive Acceleration

Curve	Velocity Max	Accel Max	Accel Min	Jerk Max	Jerk Min	Inertia Torque Max	Comment
No Dwell Modified Trapezoid	1.72	4.20	4.20	26.40	-26.40	5.07	Lowest Jerk
One Dwell Cycloidal (m=1)	1.76	5.53	5.53	34.70	-34.70	6.32	
Cycloidal	2.00	6.28	6.28	39.50	-39.50	8.16	
No Dwell Modified Constant Velocity	1.22	7.68	7.68	48.20	-48.20	4.69	Lowest Velocity
One Dwell Cycloidal (m=2/3)	1.72	6.75	-4.50	53.00	-53.00	7.53	
One Dwell Modified Trapezoid (m=1)	1.92	4.44	-4.44	55.80	-55.80	7.11	
One Dwell Modified Trapezoid (Ferguson)	1.92	4.68	-4.22	58.90	-58.90	7.43	
Modified Trapezoid	2.00	4.89	4.89	61.40	-61.40	8.09	
One Dwell Trapezoid	1.74	4.91	4.91	61.70	-61.70	6.86	
Asymmetrical Cycloidal	2.00	7.85	-5.24	61.70	-61.70	10.20	
One Dwell Modified Sine	1.66	5.21	5.21	65.50	-21.80	4.86	
One Dwell Modified Trapezoid (m=2/3)	1.94	5.53	-3.68	69.40	-69.40	8.63	
Modified Sine	1.76	5.53	-5.53	69.50	-23.20	5.46	
Trapezoid	2.18	6.17	6.17	77.50	-77.50	10.84	Highest Velocity, Highest Inertial Torque
Asymmetrical Modified Trapezoid	2.00	6.11	-4.07	96.00	-96.00	10.11	
Modified Constant Velocity	1.28	8.01	8.01	201.40	-67.10	5.73	Highest Accel
Parabolic	2.00	4.00	-4.00	∞	∞	8.00	Lowest Accel, Highest Jerk
Simple Harmonic	1.57	4.93	4.93	∞	-15.50	3.88	Lowest Inertial Torque
NC2 Curve	1.79	5.89	-4.21	∞	-111.10	8.87	

Cam Curve Characteristics

Sorted by Positive Jerk

Curve	Velocity Max	Accel Max	Accel Min	Jerk Max	Jerk Min	Inertia Torque Max	Comment
No Dwell Modified Trapezoid	1.72	4.20	4.20	26.40	-26.40	5.07	Lowest Jerk
One Dwell Cycloidal (m=1)	1.76	5.53	5.53	34.70	-34.70	6.32	
Cycloidal	2.00	6.28	6.28	39.50	-39.50	8.16	
No Dwell Modified Constant Velocity	1.22	7.68	7.68	48.20	-48.20	4.69	Lowest Velocity
One Dwell Cycloidal (m=2/3)	1.72	6.75	-4.50	53.00	-53.00	7.53	
One Dwell Modified Trapezoid (m=1)	1.92	4.44	-4.44	55.80	-55.80	7.11	
One Dwell Modified Trapezoid (Ferguson)	1.92	4.68	-4.22	58.90	-58.90	7.43	
Modified Trapezoid	2.00	4.89	4.89	61.40	-61.40	8.09	
One Dwell Trapezoid	1.74	4.91	4.91	61.70	-61.70	6.86	
Asymmetrical Cycloidal	2.00	7.85	-5.24	61.70	-61.70	10.20	
One Dwell Modified Sine	1.66	5.21	5.21	65.50	-21.80	4.86	
One Dwell Modified Trapezoid (m=2/3)	1.94	5.53	-3.68	69.40	-69.40	8.63	
Modified Sine	1.76	5.53	-5.53	69.50	-23.20	5.46	
Trapezoid	2.18	6.17	6.17	77.50	-77.50	10.84	Highest Velocity, Highest Inertial Torque
Asymmetrical Modified Trapezoid	2.00	6.11	-4.07	96.00	-96.00	10.11	
Modified Constant Velocity	1.28	8.01	8.01	201.40	-67.10	5.73	Highest Accel
Parabolic	2.00	4.00	-4.00	∞	∞	8.00	Lowest Accel, Highest Jerk
Simple Harmonic	1.57	4.93	4.93	∞	-15.50	3.88	Lowest Inertial Torque
NC2 Curve	1.79	5.89	-4.21	∞	-111.10	8.87	

Cam Curve Characteristics

Sorted by Combined Score Rank

Curve	Velocity Max	Accel Max	Accel Min	Jerk Max	Jerk Min	Inertia Torque Max	Comment
No Dwell Modified Trapezoid	1.72	4.20	4.20	26.40	-26.40	5.07	Lowest Jerk
One Dwell Modified Trapezoid (m=1)	1.92	4.44	-4.44	55.80	-55.80	7.11	
One Dwell Cycloidal (m=1)	1.76	5.53	5.53	34.70	-34.70	6.32	
No Dwell Modified Constant Velocity	1.22	7.68	7.68	48.20	-48.20	4.69	Lowest Velocity
One Dwell Trapecloid	1.74	4.91	4.91	61.70	-61.70	6.86	
One Dwell Modified Trapezoid (Ferguson)	1.92	4.68	-4.22	58.90	-58.90	7.43	
One Dwell Modified Sine	1.66	5.21	5.21	65.50	-21.80	4.86	
One Dwell Cycloidal (m=2/3)	1.72	6.75	-4.50	53.00	-53.00	7.53	
Modified Trapezoid	2.00	4.89	4.89	61.40	-61.40	8.09	
Simple Harmonic	1.57	4.93	4.93	∞	-15.50	3.88	Lowest Inertial Torque
Modified Sine	1.76	5.53	-5.53	69.50	-23.20	5.46	
Parabolic	2.00	4.00	-4.00	∞	∞	8.00	Lowest Accel, Highest Jerk
Cycloidal	2.00	6.28	6.28	39.50	-39.50	8.16	
One Dwell Modified Trapezoid (m=2/3)	1.94	5.53	-3.68	69.40	-69.40	8.63	
Modified Constant Velocity	1.28	8.01	8.01	201.40	-67.10	5.73	Highest Accel
NC2 Curve	1.79	5.89	-4.21	∞	-111.10	8.87	
Asymmetrical Modified Trapezoid	2.00	6.11	-4.07	96.00	-96.00	10.11	
Asymmetrical Cycloidal	2.00	7.85	-5.24	61.70	-61.70	10.20	
Trapecloid	2.18	6.17	6.17	77.50	-77.50	10.84	Highest Velocity, Highest Inertial Torque



Cam Curve Types

- [Parabolic](#)
- [Simple Harmonic](#)
- [Cycloidal](#)
- [Modified Trapezoid](#)
- [Modified Sine](#)
- [Modified Constant Velocity](#)
- [Asymmetrical Cycloidal](#)
- [Asymmetrical Modified Trapezoid](#)
- [Trapezoid](#)
- [One Dwell Cycloidal_1](#)
- [One Dwell Cycloidal_2_3](#)
- [One Dwell Trapezoid_1](#)
- [One Dwell Trapezoid](#)
- [One Dwell Trapezoid_2_3](#)
- [One Dwell Modified Sine](#)
- [One Dwell Trapezoid](#)
- [No Dwell Simple Harmonic](#)
- [No Dwell Modified Trapezoid](#)
- [No Dwell Modified Constant Velocity](#)
- [NC2 Curve](#)
- [Tangent Matching](#)
- [Reverse Trapezoid](#)
- [Double Harmonic](#)
- [Reverse Double Harmonic](#)
- [Tangent Blending](#)
- [Arc](#)
- [Cubic Spline](#)
- [ParabolicVelocityBlend](#)



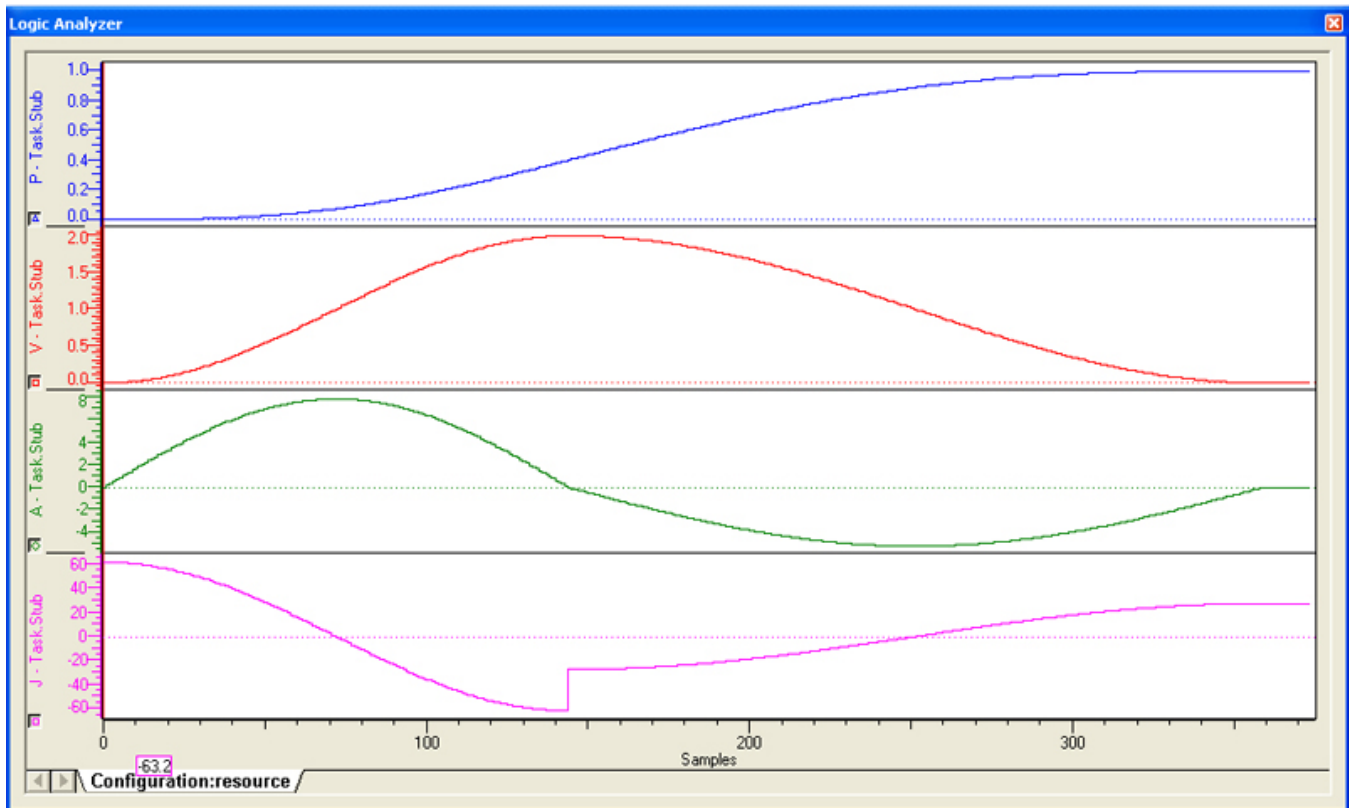
Arc

The CamSegmentStruct elements ArcRadius and ArcDirection must be declared for proper usage of this curve type.



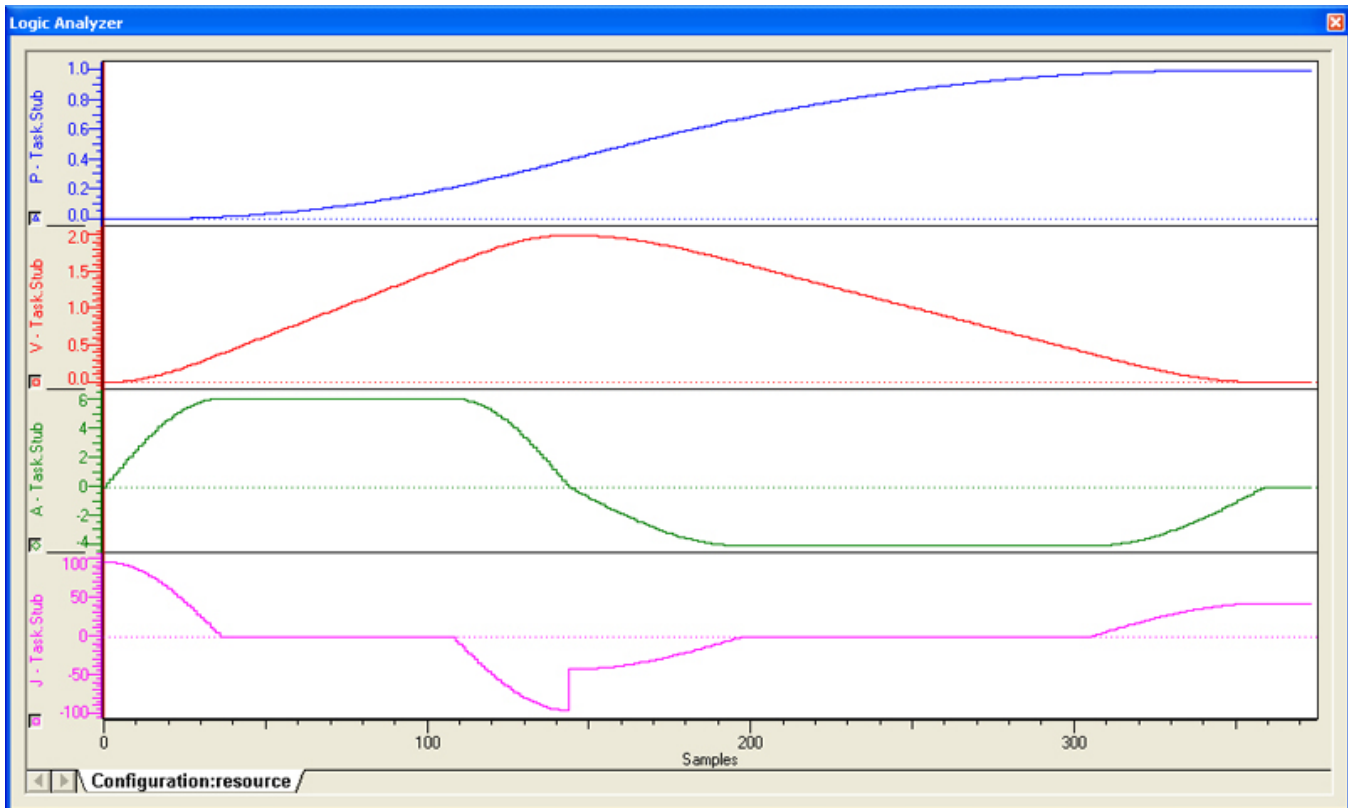


Asymmetrical Cycloidal





Asymmetrical Modified Trapezoid





Bezier

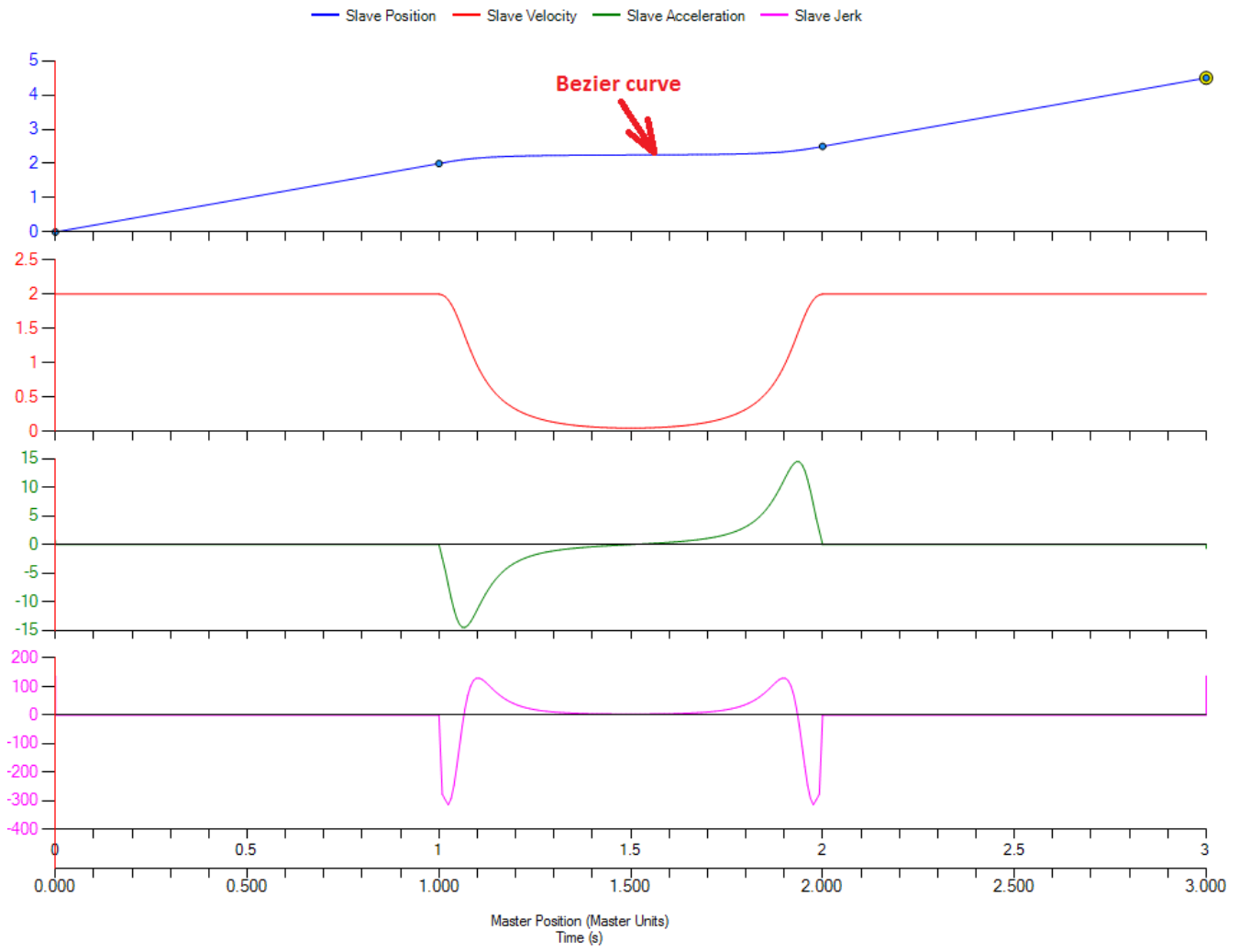
The Bezier curve type generates a non reversing bezier profile between two straight lines.

```
MyCam.SlaveStart := LREAL#0.0;
MyCam.LastSegment := INT#3;
MyCam.UseSplineSlope := FALSE;

MyCam.CamParameters[1].CurveType := TB_CurveType#StraightLine;
MyCam.CamParameters[1].MasterEnd := LREAL#1.0;
MyCam.CamParameters[1].SlaveEnd := LREAL#2.0;
MyCam.CamParameters[1].Resolution := REAL#0.0;

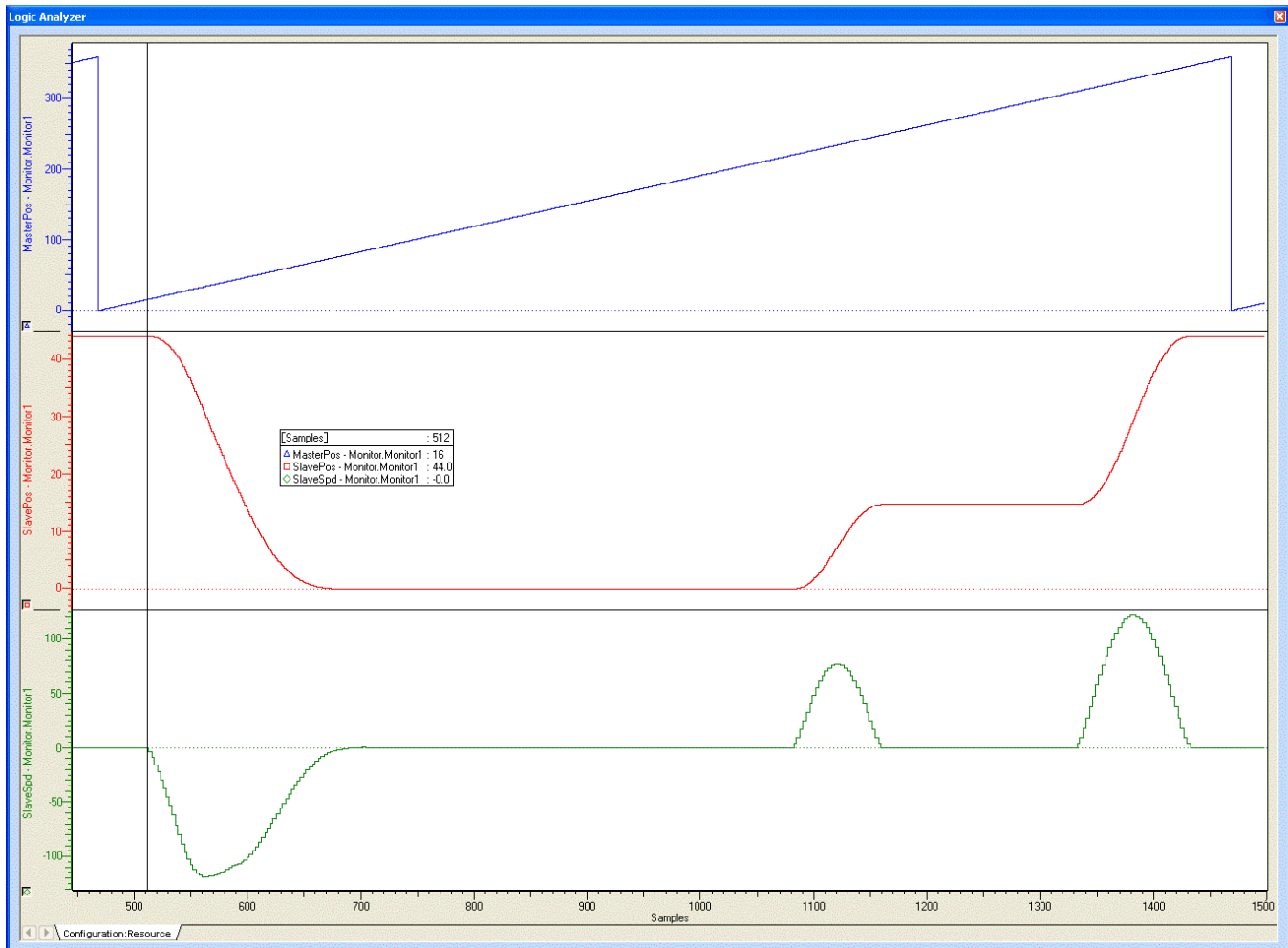
MyCam.CamParameters[2].CurveType := TB_CurveType#Bezier;
MyCam.CamParameters[2].MasterEnd := LREAL#2.0;
MyCam.CamParameters[2].SlaveEnd := LREAL#2.5;
MyCam.CamParameters[2].Resolution := REAL#0.01;

MyCam.CamParameters[3].CurveType := TB_CurveType#StraightLine;
MyCam.CamParameters[3].MasterEnd := LREAL#3.0;
MyCam.CamParameters[3].SlaveEnd := LREAL#4.5;
MyCam.CamParameters[3].Resolution := REAL#0.0;
```





Cubic Spline



In this example, the left or beginning portion of a motion profile was created using the cubic spline formula. The right or end portion of the cycle includes two modified sine motions.

The CamData values are shown below:

(* test cubic spline *)

Profile4.SlaveStart:=LREAL#44.0; (* The slaves initial and final position is not zero, it is 44.0 *)

seg:=INT#1;

Profile4.CamParameters[Seg].CurveType:=TB_CurveType#StraightLine;

Profile4.CamParameters[Seg].MasterEnd:=LREAL#15.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#44.0;

```

Profile4.CamParameters[Seg].Resolution:=REAL#0.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#17.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#43.9614;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#25.5;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#40.3036;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#34.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#30.4425;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#42.5;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#19.6003;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#43.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#19.0;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#51.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#10.0305;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#59.5;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#3.5477;

```



```
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#68.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#0.6464;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#76.5;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#0.005;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#CubicSpline;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#85.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#0.0;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#StraightLine;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#220.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#0.0;
Profile4.CamParameters[Seg].Resolution:=REAL#0.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#ModifiedSine;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#250.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#14.7;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#StraightLine;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#310.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#14.7;
Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;
Profile4.CamParameters[Seg].CurveType:=TB_CurveType#ModifiedSine;
Profile4.CamParameters[Seg].MasterEnd:=LREAL#348.0;
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#44.0;
```

Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

seg:=Seg + INT#1;

Profile4.CamParameters[Seg].CurveType:=TB_CurveType#ModifiedSine;

Profile4.CamParameters[Seg].MasterEnd:=LREAL#360.0;

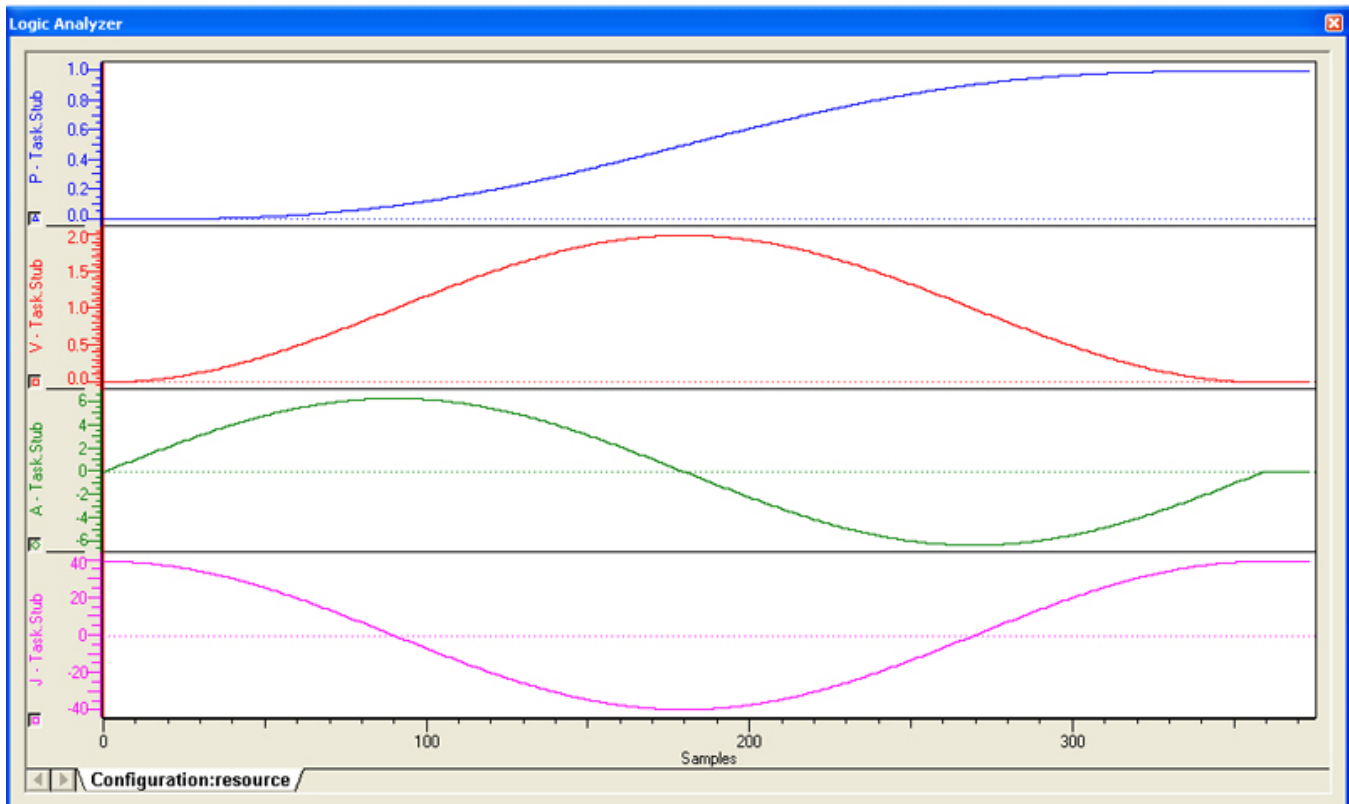
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#44.0;

Profile4.CamParameters[Seg].Resolution:=REAL#1.0;

Profile4.LastSegment:=Seg;



Cycloidal



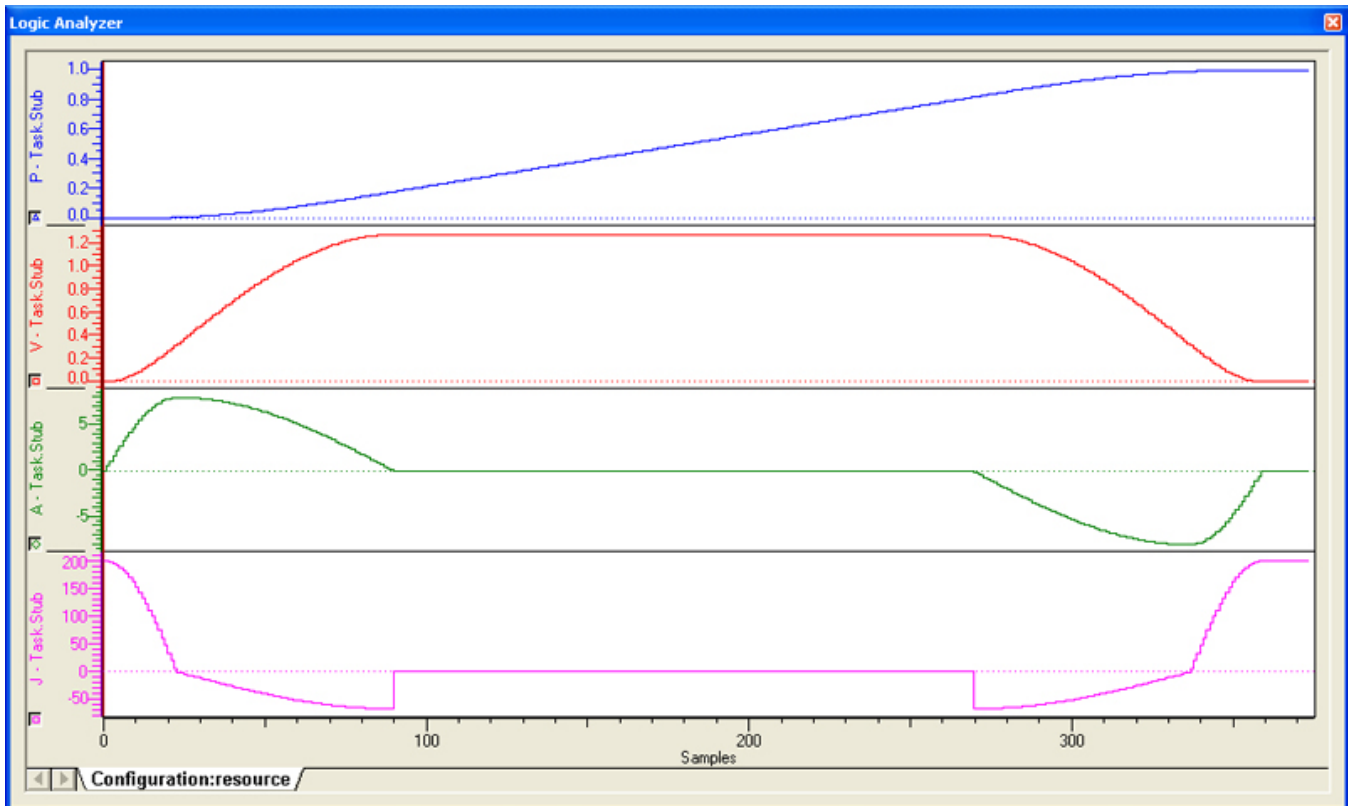
Double Harmonic



This curve type is not supported.

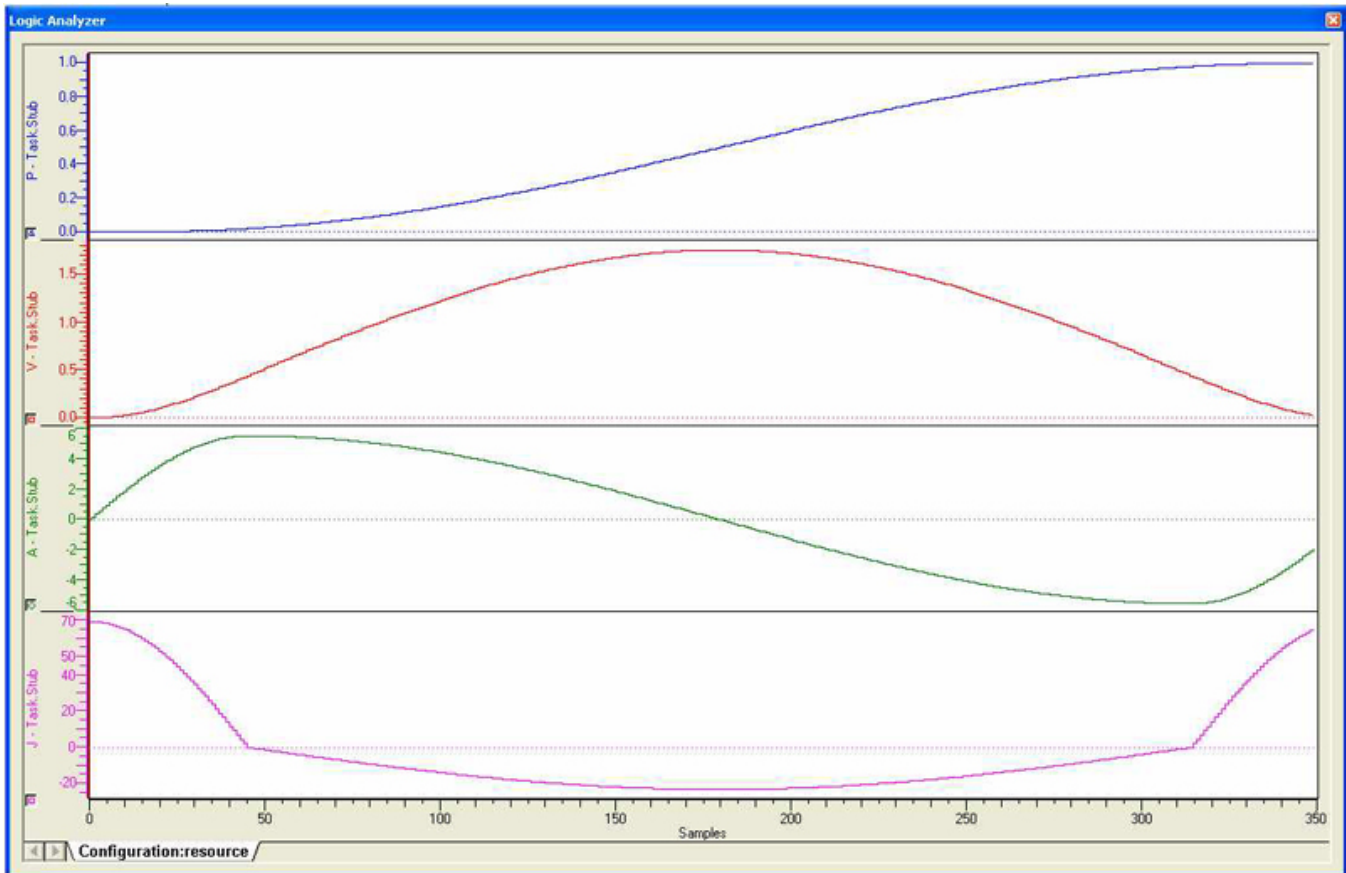


Modified Constant Velocity



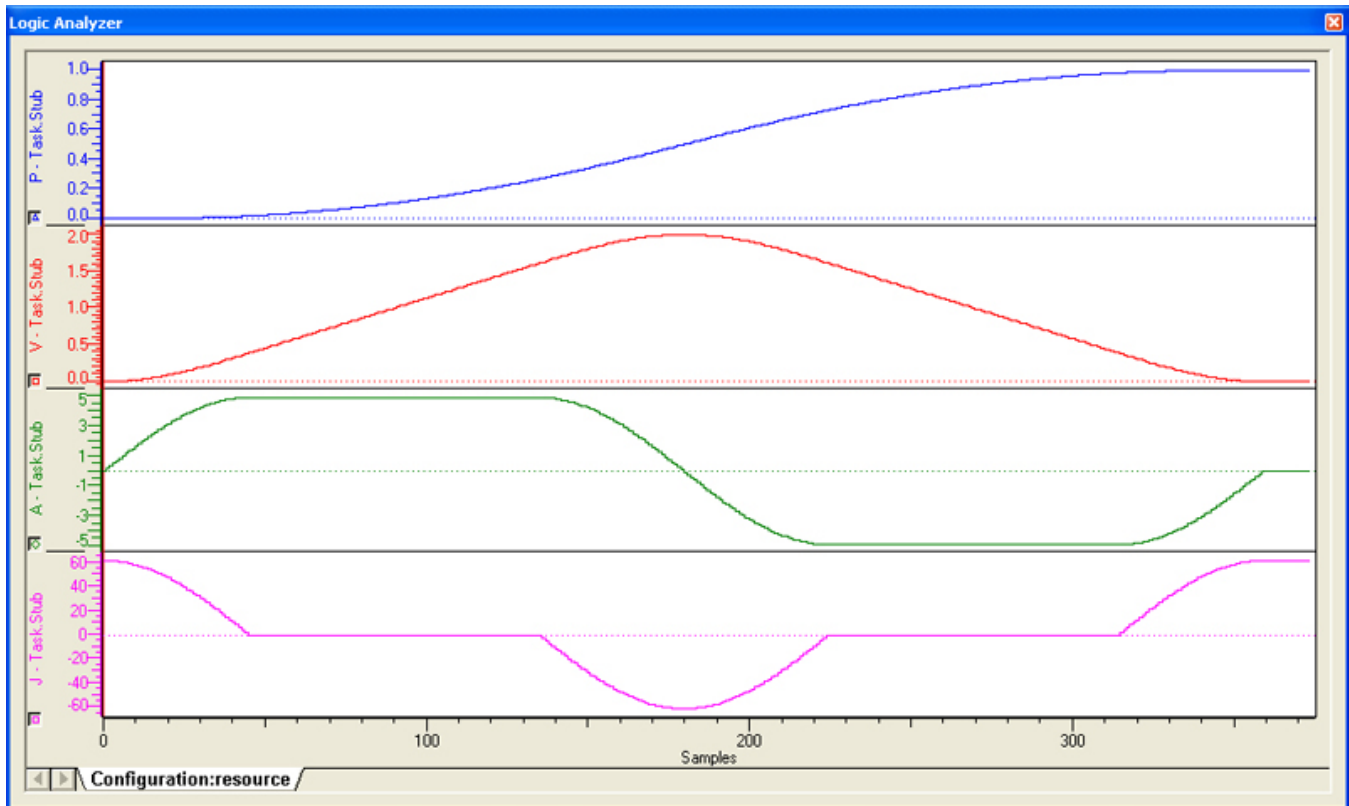


Modified Sine





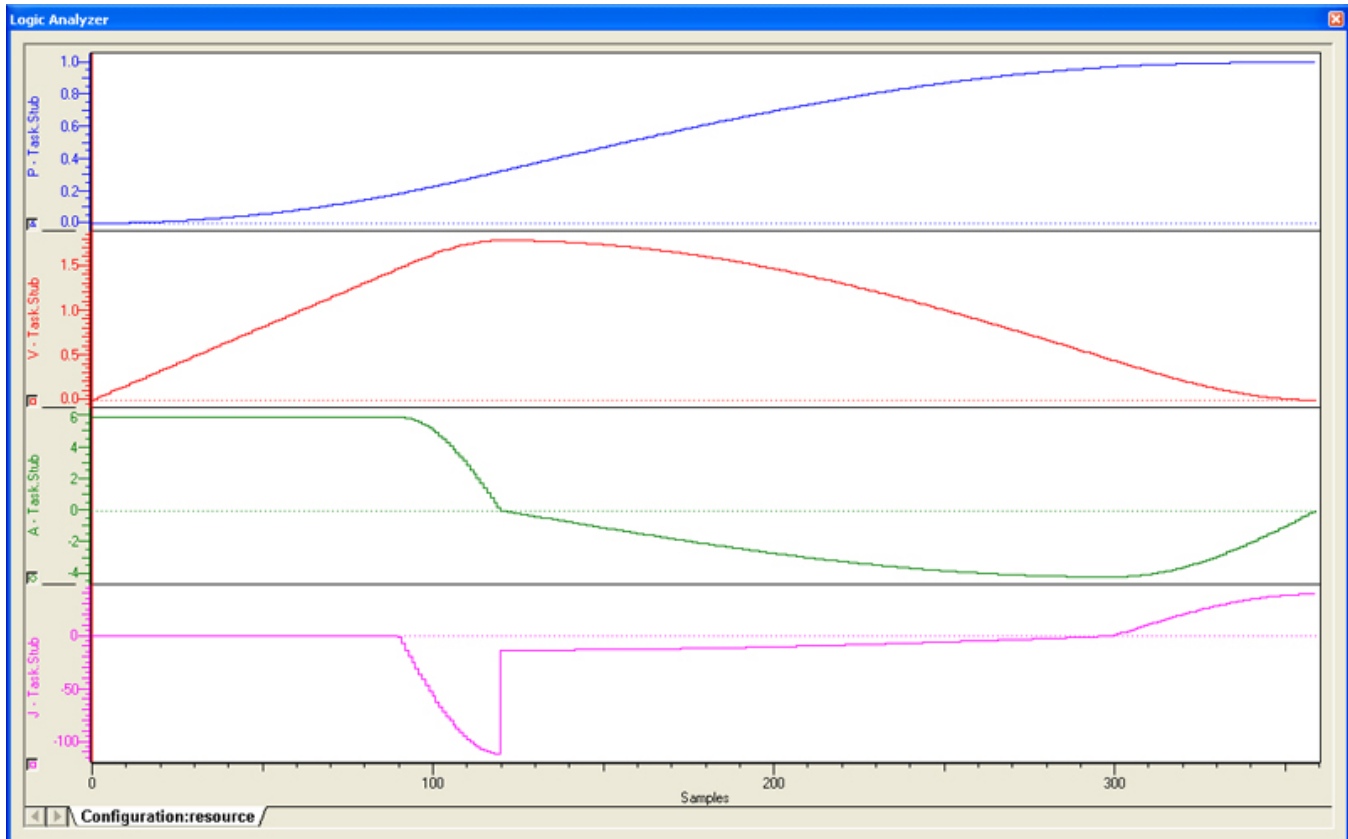
Modified Trapezoid





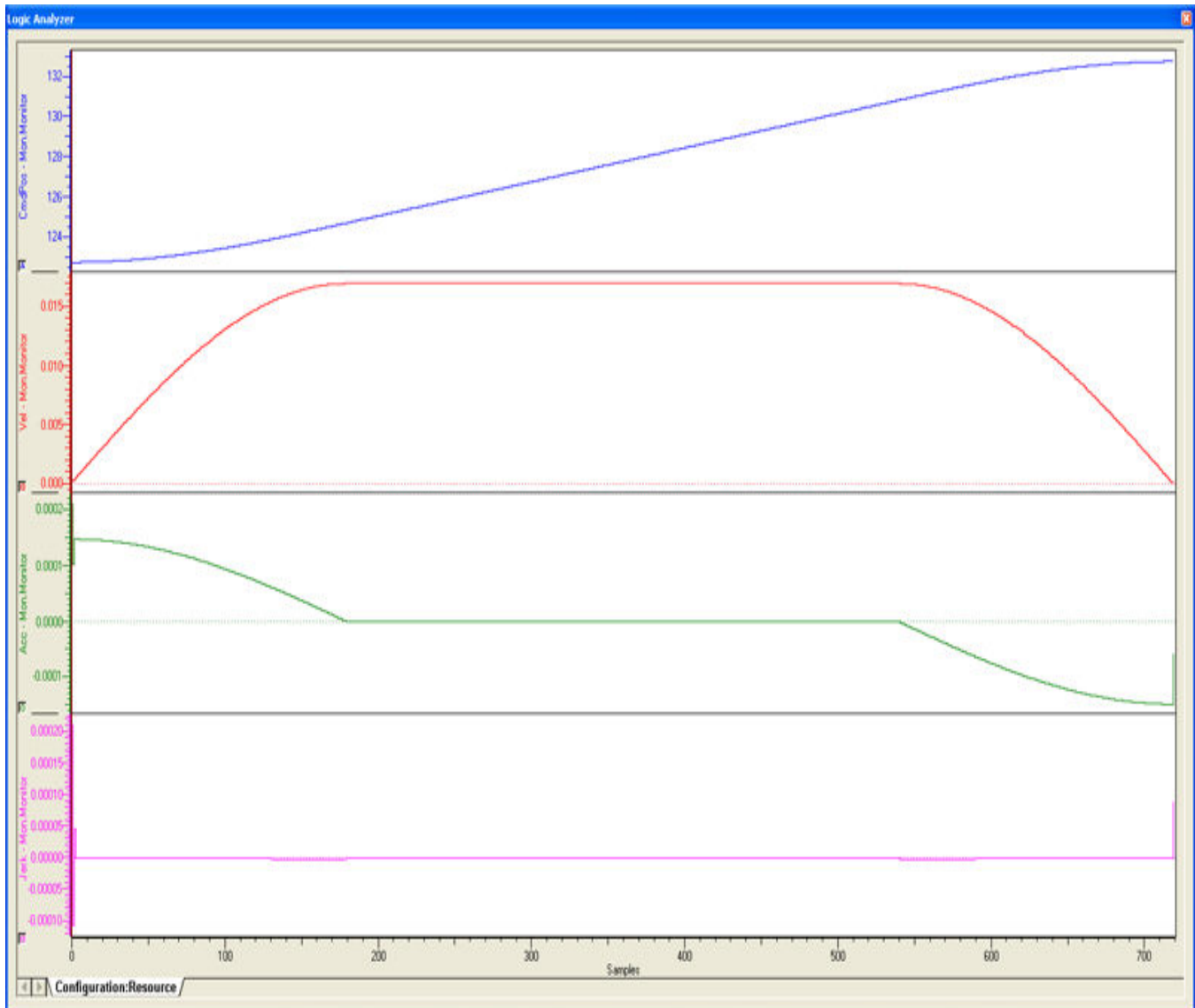
NC2 Curve

Notes: Deceleration is twice as long as acceleration, which provides the effect of restricting vibration.



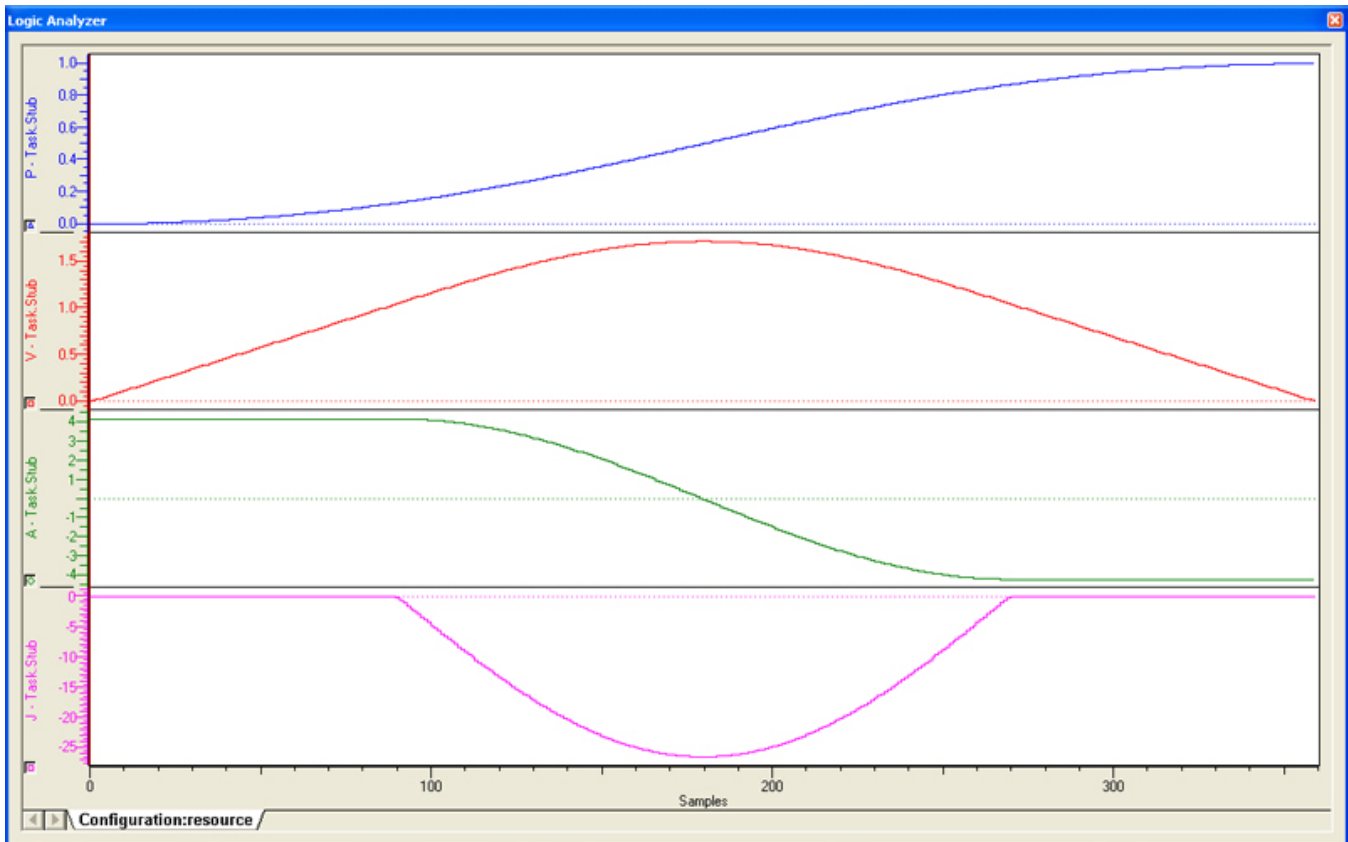


No Dwell Modified Constant Velocity



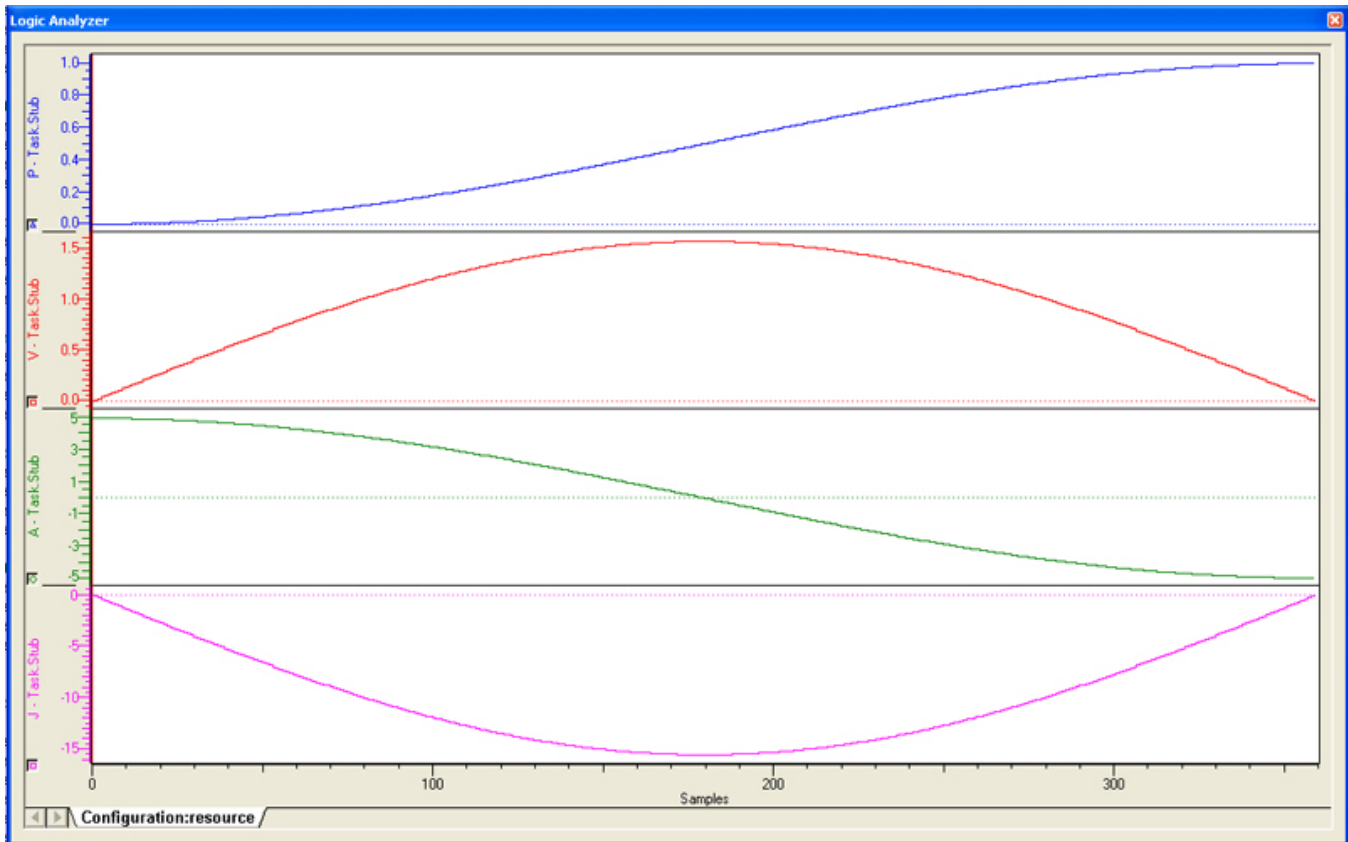


No Dwell Modified Trapezoid



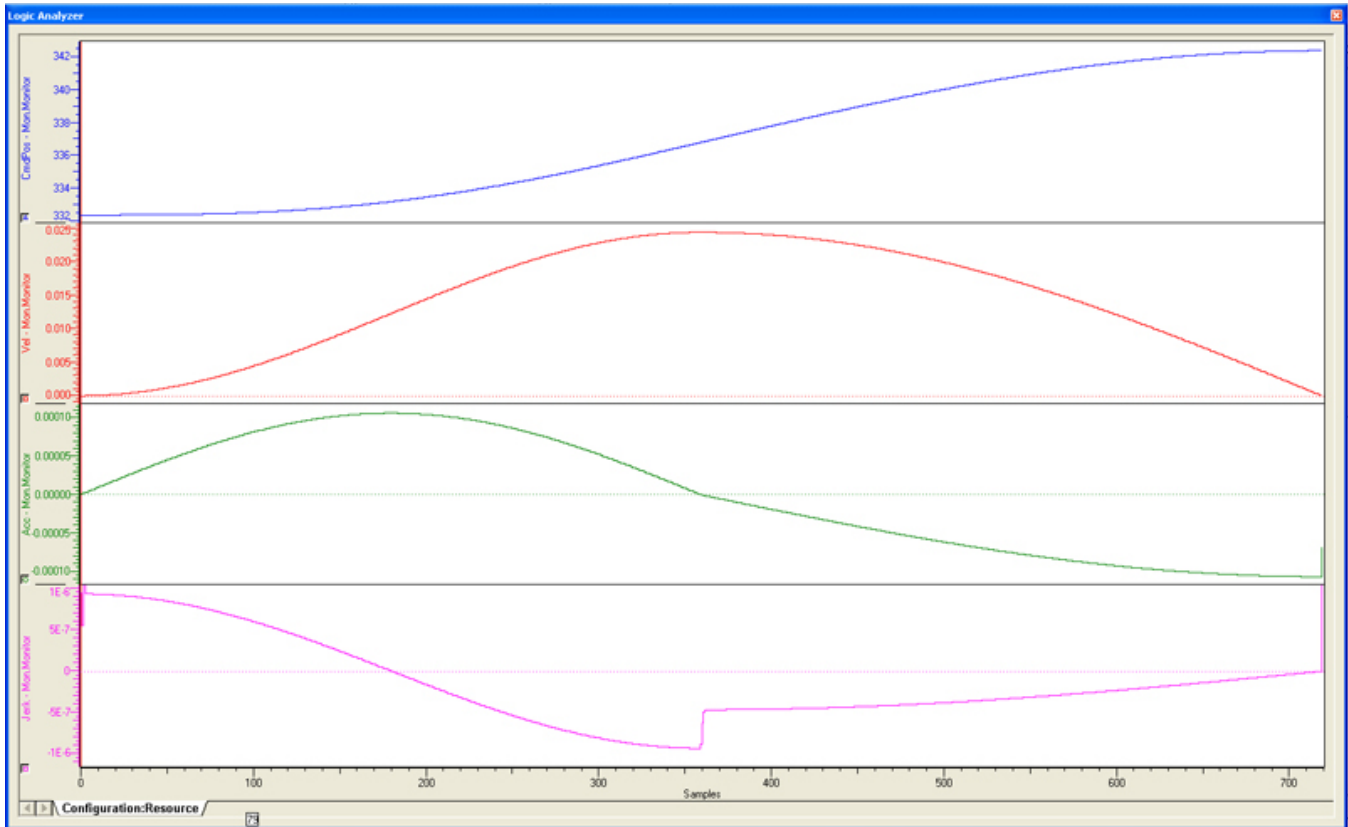


No Dwell Simple Harmonic



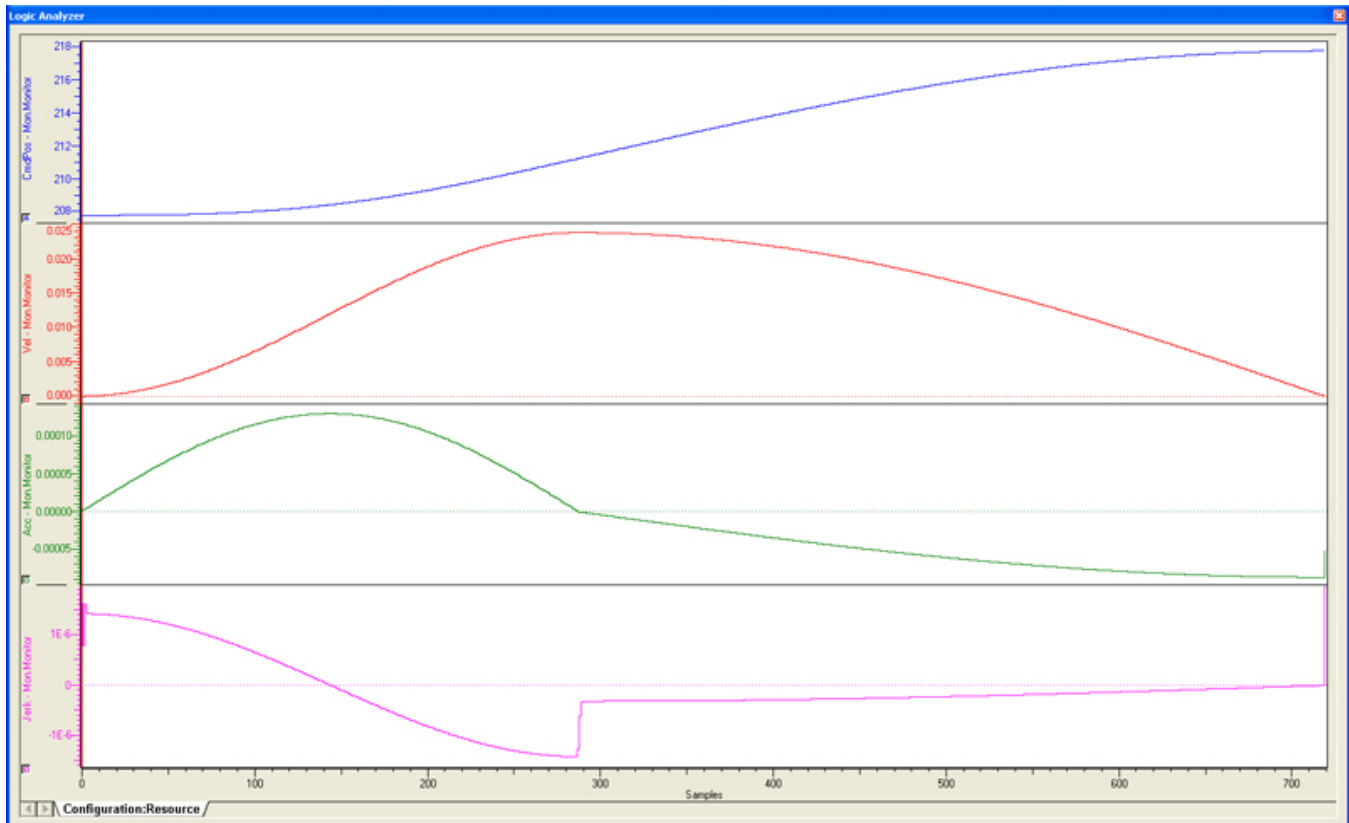


One Dwell Cycloidal_1



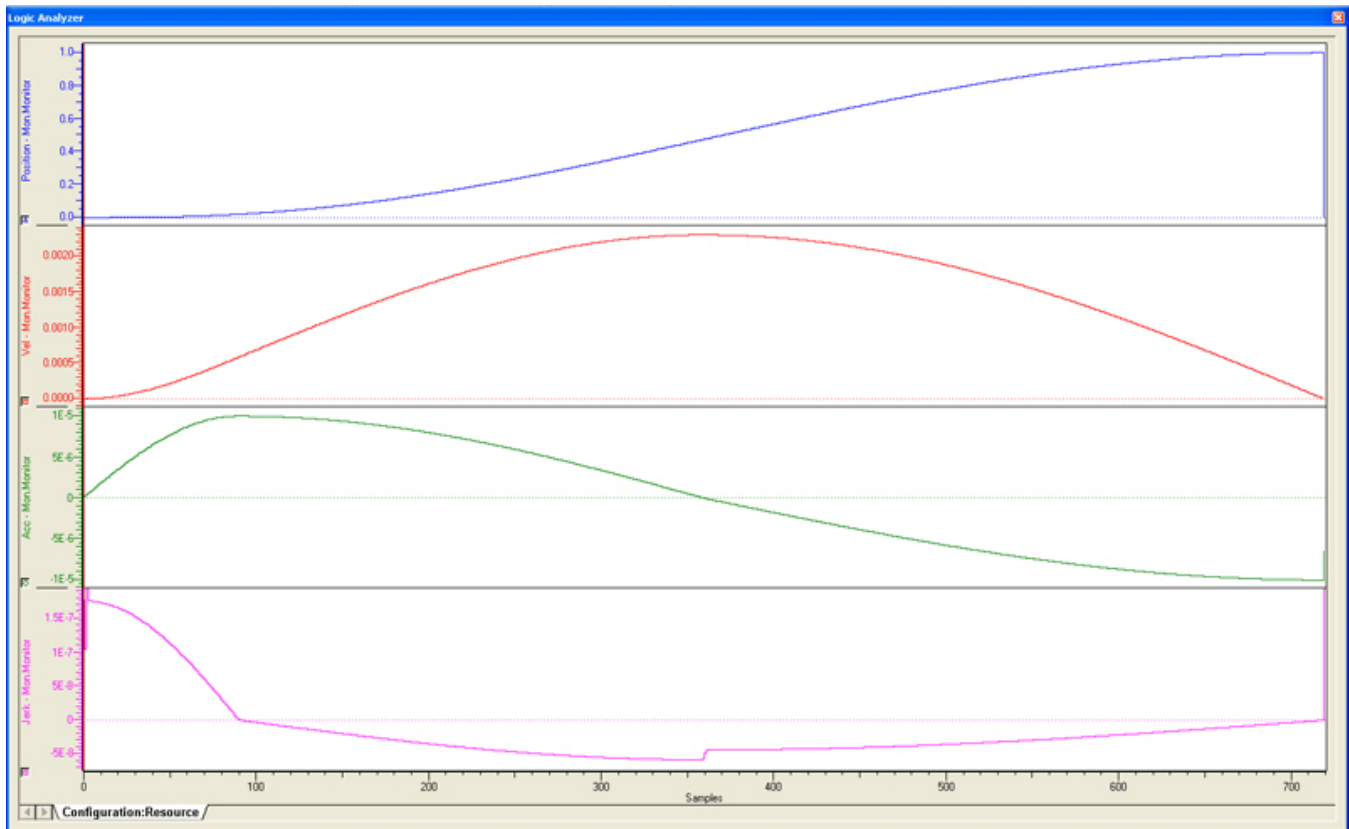


One Dwell Cycloidal_2_3



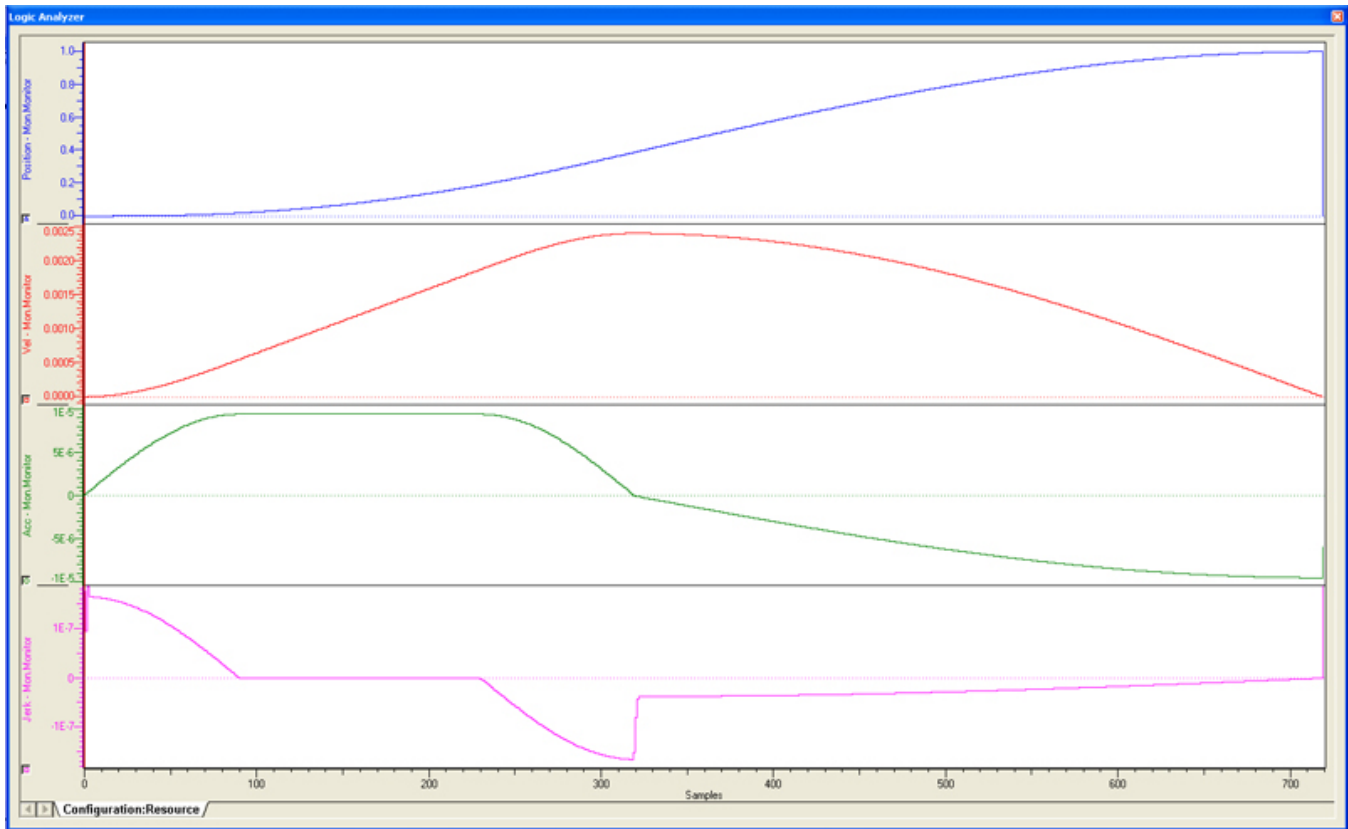


One Dwell Modified Sine



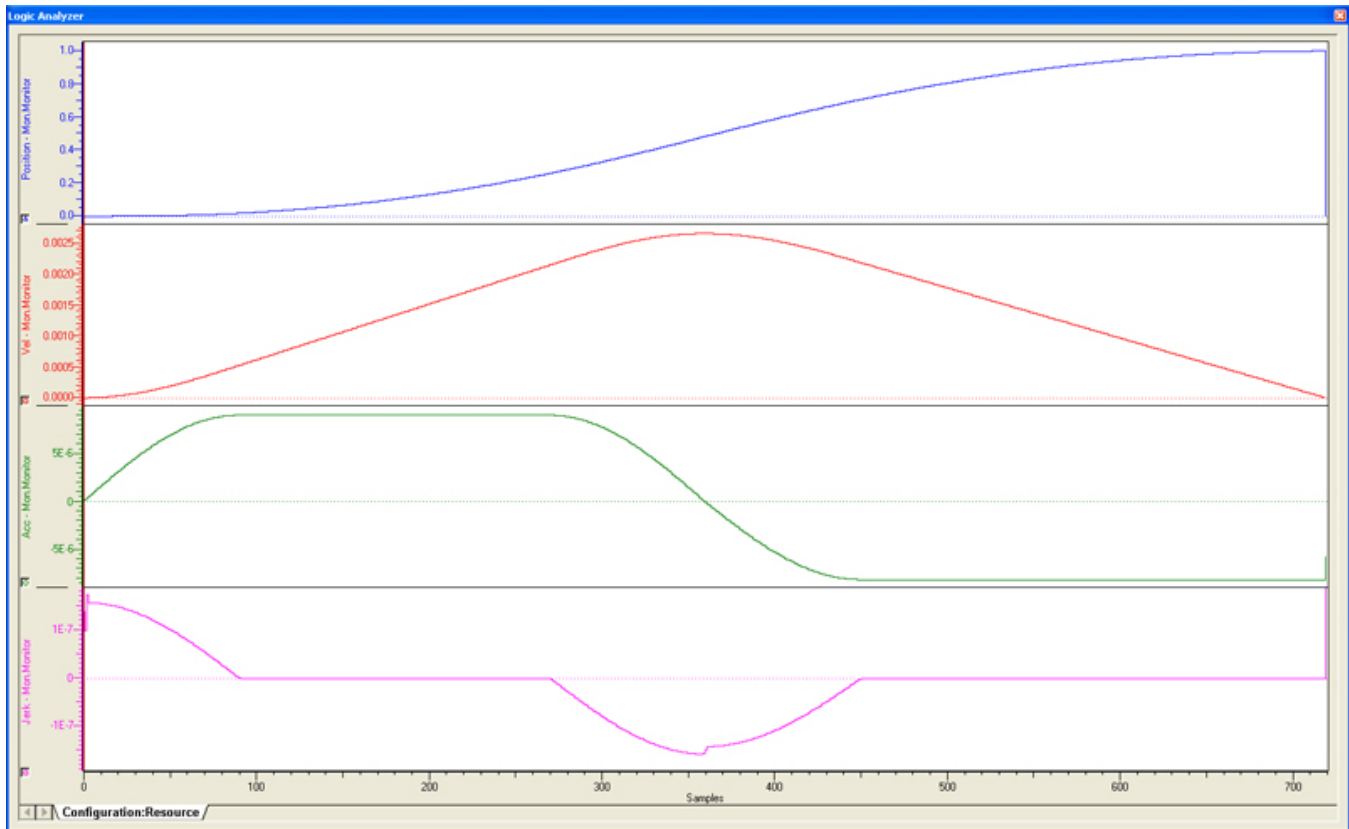


One Dwell Trapezoid



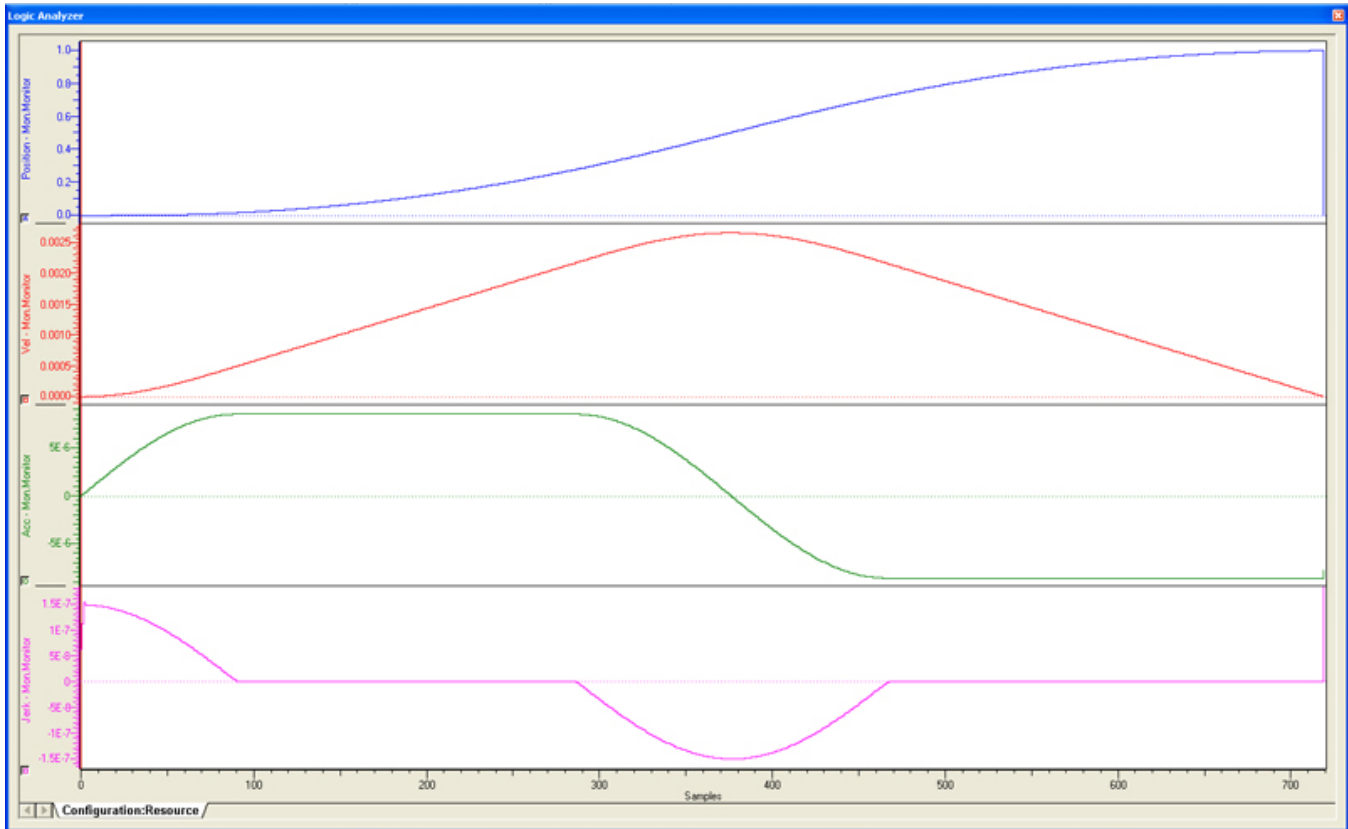


One Dwell Trapezoid



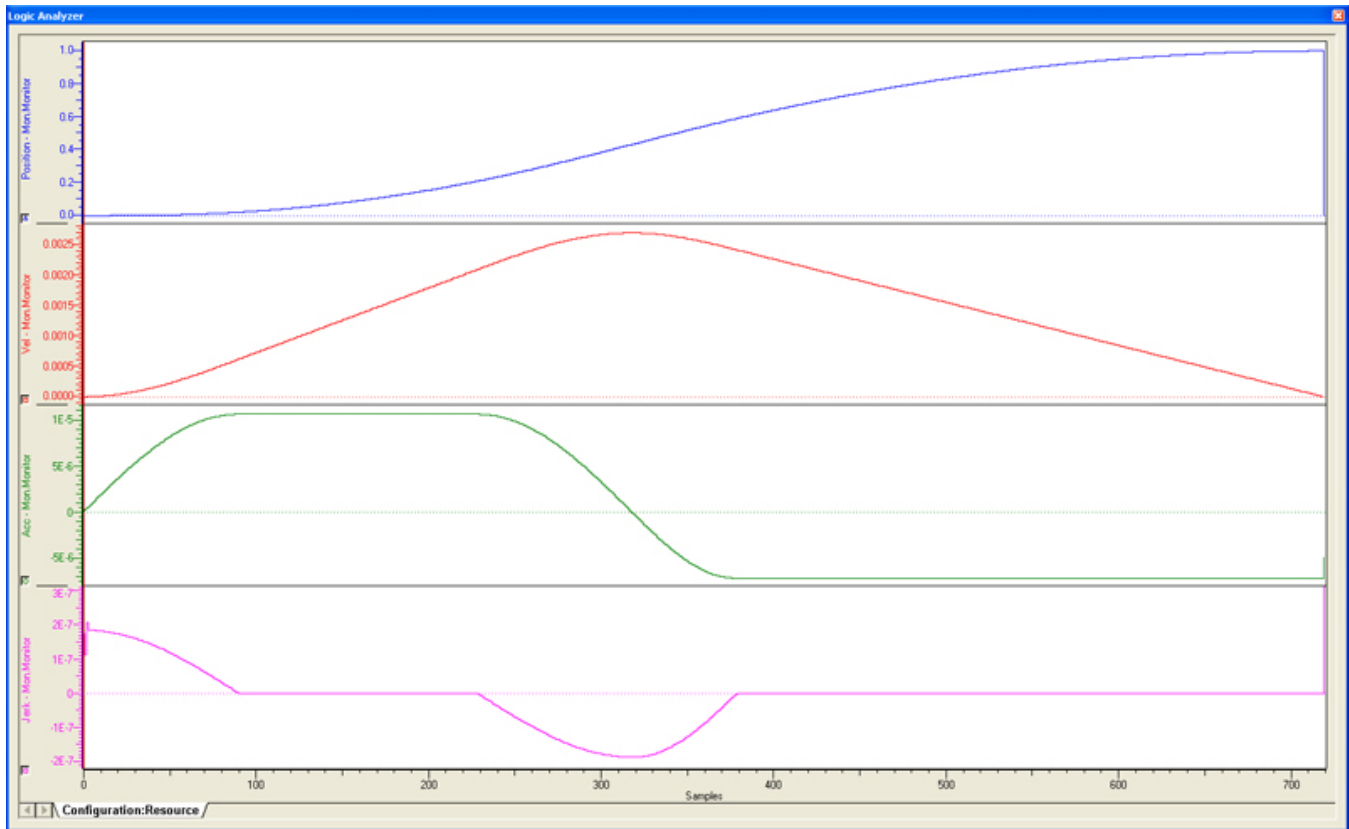


One Dwell Trapezoid_1





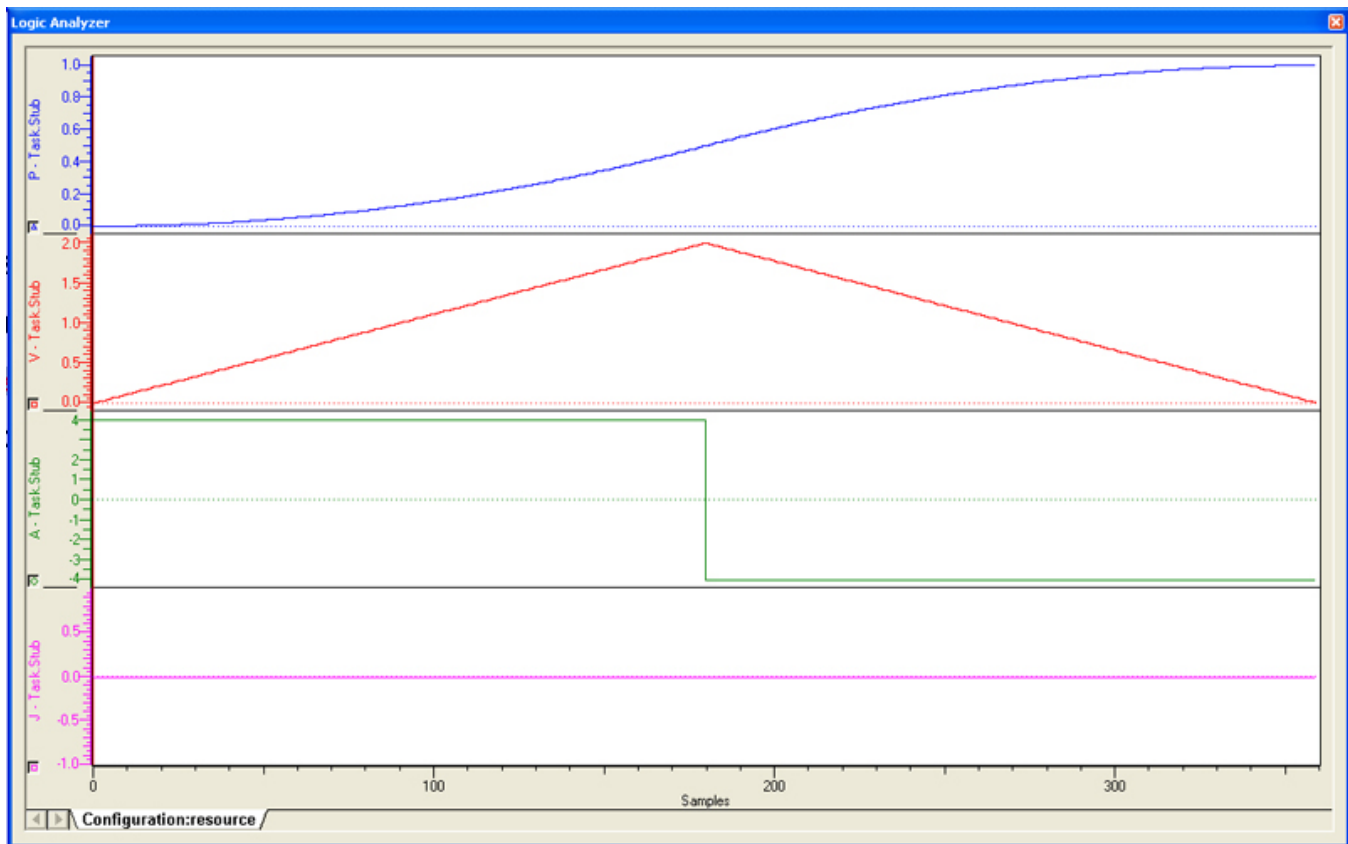
One Dwell Trapezoid_2_3





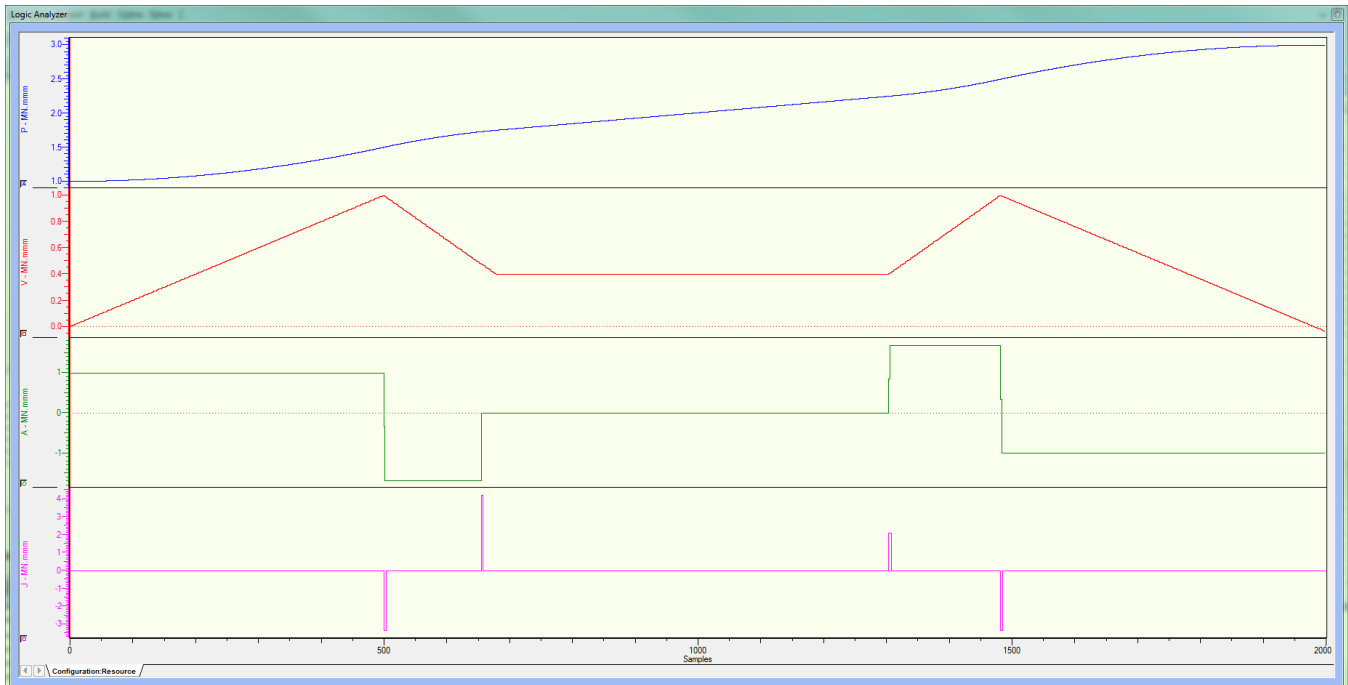
Parabolic

Designed for use as the only segment in the motion profile when a axis must be indexed. This curve has the feature that the non dimensional maximum acceleration A_m is the minimum ($A_m=4$) among all curves. Downside - Can cause vibration. Modified Trapezoid is better.





ParabolicVelocityBlend



Reverse Double Harmonic



This curve type is not supported.

Reverse Trapecloid

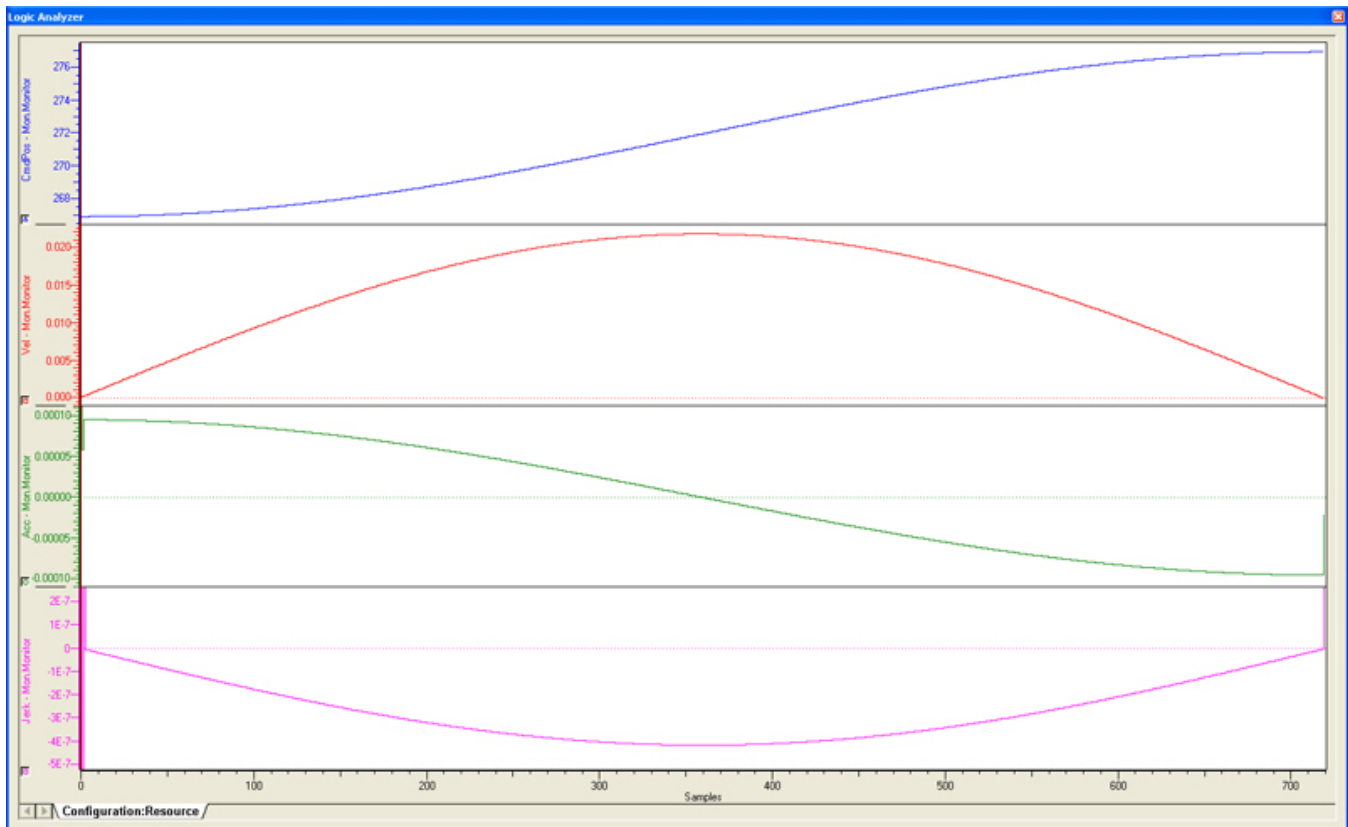


This cam curve type is not supported.



Simple Harmonic

This curve is also one of the discontinuous curves that easily causes vibration, but since it has smooth and good (low) properties, it can be used for low speed applications. When this curve is used for no-dwell applications, (out & back) the discontinuity of acceleration at the starting and end points is not a factor and then this curve is regarded as the best curve for no-dwell use. The modified sine curve is considered an improvement over the simple harmonic.





Tangent Blending

Provides the same profile as [Tangent Matching](#), but designed for use with the [CamBlend](#) function block. The difference between this and Tangent Matching is how the matching velocity is determined. For this formula type, two segments are required: a straight line and a tangent blend. Which segment comes first dictates whether a “blend in” or “blend out” profile is created.

See the [CamBlend](#) function block for application examples


```

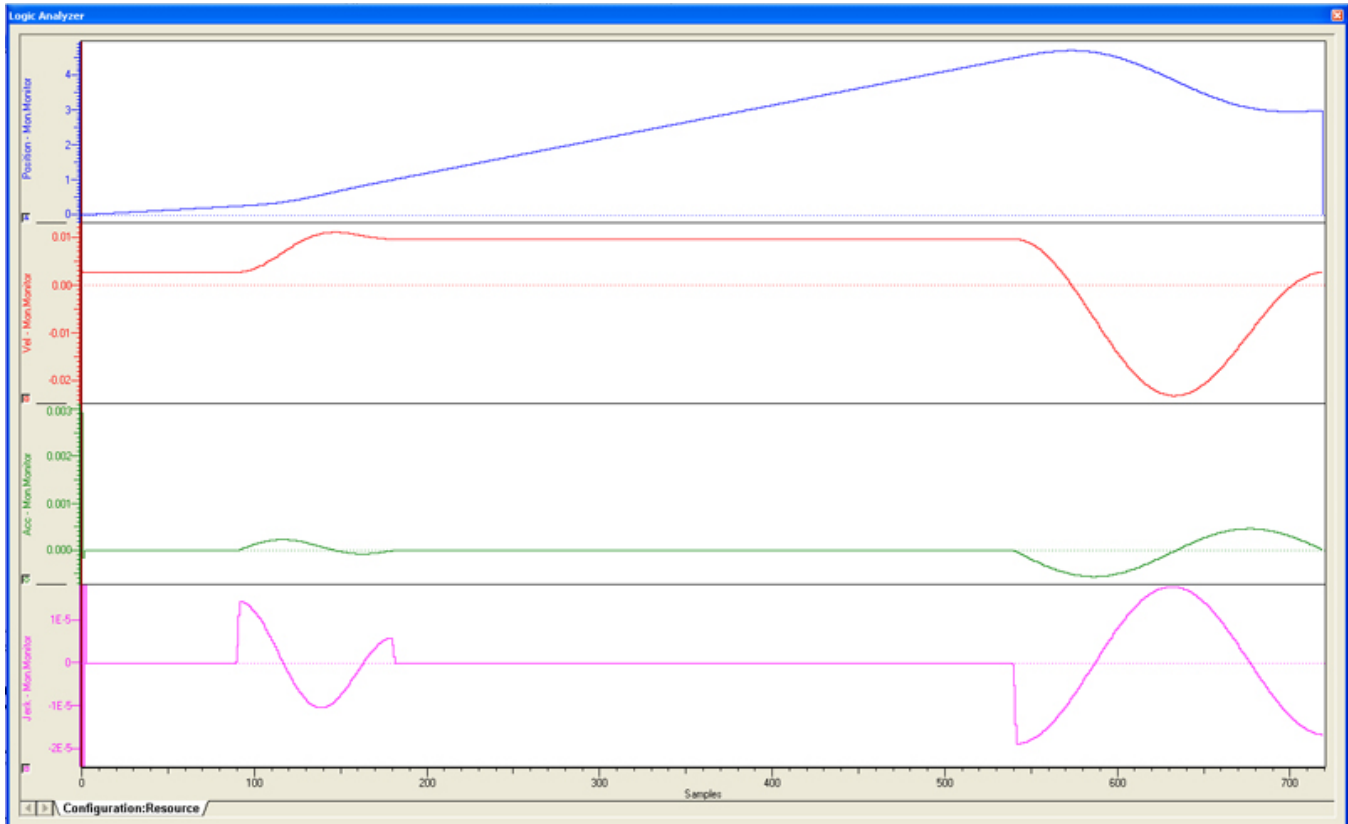
0.000 CamTool.SlaveStart:=LREAL#0.0;
4 CamTool.LastSegment:=INT#4;
1 CamTool.CamParameters[1].CurveType:=INT#1;
45.000 CamTool.CamParameters[1].MasterEnd:=LREAL#45.0;
0.250 CamTool.CamParameters[1].SlaveEnd:=LREAL#0.25;
0.500 CamTool.CamParameters[1].Resolution:=REAL#0.5;

22 CamTool.CamParameters[2].CurveType:=INT#22;
90.000 CamTool.CamParameters[2].MasterEnd:=LREAL#90.0;
1.000 CamTool.CamParameters[2].SlaveEnd:=LREAL#1.0;
0.500 CamTool.CamParameters[2].Resolution:=REAL#0.5;

1 CamTool.CamParameters[3].CurveType:=INT#1;
270.000 CamTool.CamParameters[3].MasterEnd:=LREAL#270.0;
4.500 CamTool.CamParameters[3].SlaveEnd:=LREAL#4.5;
0.500 CamTool.CamParameters[3].Resolution:=REAL#0.5;

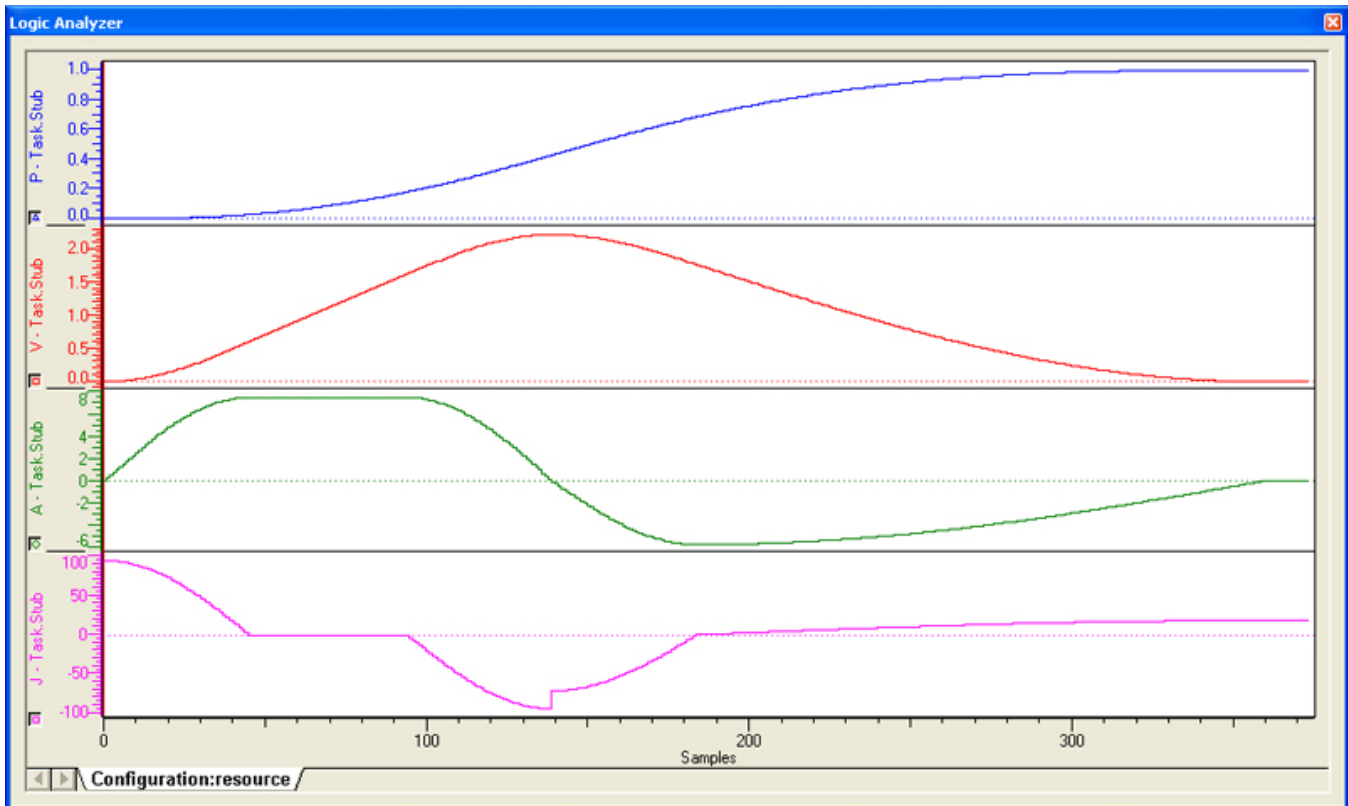
22 CamTool.CamParameters[4].CurveType:=INT#22;
360.000 CamTool.CamParameters[4].MasterEnd:=LREAL#360.0;
3.000 CamTool.CamParameters[4].SlaveEnd:=LREAL#3.0;
0.500 CamTool.CamParameters[4].Resolution:=REAL#0.5;

```





Trapezoid





Getting Started with Communications Toolbox

Requirements for v374

To use the Communications Toolbox, the project must also contain the following:

Firmware libraries:

- YDeviceComm
- PROCONOS

User libraries:

The following User Libraries must be listed above the Communications Toolbox:

- Yaskawa_Toolbox (v371 or higher)



Communications Revision History

Current Version:

2022-06-06 v374 released

- 1) CommunicationChannel FB - Added delay timer to work with Compass 1.4.0 streaming performance increase. DCR 6608.
- 2) CommunicationChannel FB - Increase size of 'InputBuffer' from 2048 to 8192 bytes. DCR 6634.
- 3) Increase CircularByteBuffer datatype to 32000. DCR 6553.

Previous Versions:

2021-12-14 v373 released

This version was only released with Compass 1.3.0 and its MotionWorks project template AN.MPIEC.34.

- 1) CircularBufferStruct: Increased size of CircularBufferStruct.Data[] from 8192 to 16384. DCR 6246. This is for increased throughput performance for Compass G-Code streaming.

2021-04-27 v372 released

No changes, identical to v371

2020-11-23 v371 released

- 1) GetCommand FB - HOTFIX! Can errantly increment the UsePointer PAST the StorePointer when CR is the last character received, AND the CircularByteBuffer has a LF as the next char from the previous circulation of chars. DCR 4403.
- 2) GetCommand FB - Improve to return CommandCreated=FALSE if Char StorePointer=UsePointer. DCR 4304.

2020-02-06 v370 released

- 1) Rename_FTP_SendData was reporting ErrorID=0 when Error TRUE. DCR 2684.
- 2) GetCommand FB was not referencing Enable VAR_INPUT. DCR 2978.

2019-02-06 v352 released

- 1) FTP_SendFile - Increased file size capability. DCR 773
- 2) FTP_SendFile - Improved for file size and writing speed. DCR 1102.
- 3) FTP_SendFile / (dns_translate_response FB) causing compile errors in MP2600iec and MP3000iec controller types. DCR 2338.
- 4) FTP_SendFile - Added Append VAR_INPUT. DCR 2342.
- 5) ReName_FTP_SendData - New function block. DCR 2389.
- 6) ReName_FTP_RecordData - New Function block. DCR 2389.

2018-08-06 v350 released

- 1) GetCommand - White spaces (extra CRLF) in G-Code files caused next line to be ignored. DCR 1525.
- 2) ReName_UserComm - New function block added to send / receive application specific structure via socket communication with a host via UserComm.DLL

2017-12-06 v340 released

- 1) CommunicationChannel - Improved operation by not causing any errors when the host application closes the socket. DCR 1206.
- 2) CommunicationChannel - Added PacketCount and BytesReceived as VAR_OUTPUTs. DCR 1298.
- 3) CommunicationChannel / InputBufferManager - Added auto setting of CircularByteBuffer.Size using UPPER_BOUND so the user is not required to initialize it. DCR 1373.

2016-12-13 v301 released

- 1) CommunicationChannel - Changed PacketSizeError logic.
- 2) GetCommand FB - DCR 701 Bug fix when buffer ends with partial command.
- 3) CommunicationChannel - DCR 1028, added explicit setting of TCP read Buffer, added 'CopyBusy' flag to throttle Y_ReadDevice DB
- 4) InputBufferManager - DCR 1029, FB was not making sure there was enough room for the entire copy from InputBuffer to CircularByteBuffer before starting to copy. Only a big problem (corrupted data) if heavy data streaming pushed the buffering capacity to its limits.
- 5) CommandProcessor - DCR 816 - Changed WHILE LOOP to IF block to eliminate the risk of watchdog if a partial command was sent, or if many many commands are sent. If necessary, performance can be increased by going back to the WHILE loop strategy, but checks must be in place to safeguard against watchdog faults.

2015-01-31 v300 released

Created from v202, but recompiled specifically for MotionWorks IEC v3.x.

- 1) CircularBufferStruct DataType: Added BufferedCount and BufferedPercent in preparation for support of a host PC which streams part data and must monitor the buffer levels.
- 2) CommunicationChannel - Changed CommConfig from VAR_INPUT to VAR_IN_OUT. This allows the RemoteIPAddress to be added to the structure, and shared with other parts of the program to open additional sockets to the remote host.

2014-05-02: v202 released. Requires firmware 2.2.0 and the YDeviceComm firmware library

- 1) Improved capability by switching from a STRING of 512 character to a BYTE array of 2048 characters for Command Streaming lower level functions. This change REQUIRES users who are upgrading from an older toolbox version to CHANGE the data-type of the CommandString variable in CommandProcessor function blocks in your main project from YTB_STRING512 to CTB_CommandStruct.
- 2) GetCommand - Changed DataType of CommandString from YTB_STRING512 to CTB_CommandStruct. This allows the functions to work on the data as a byte array instead of a string, which reduces scan time (fewer STRING operations) and allows for longer command strings to be processed. Previous limit was 512 characters, now it is 2048 per line (between delimiters).
- 3) GetParameter - Same changes as listed for GetCommand.
- 4) CircularByteBuffer.PrmDelimiter - changed data type from YTB_STRING1 to BYTE. Related to improvements listed above.
- 5) GetParameter - Added DecimalDetected output. This can be used to prevent string conversion errors when using STRING_TO_INT or similar conversions.

2013-09-02: v201 released. Requires firmware 2.2.0 and the YDeviceComm firmware library

- 1) ReName_CommandProcessor - Changed logic to call a sub function "GetCommand" to reduce the amount of code that

resides on the user project side.

2013-08-08: v200 released. Requires firmware 2.2.0 and the YDeviceComm firmware library

- 1) First release, includes Email, FTP, and Command Processing functions



Data Type: CircularBufferStruct

Data Structure used to manage a circular buffer of data used by several function blocks in the Communications Toolbox.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCircularBufferStruct	CircularBufferStruct		
C	StorePointer	INT	Pointer updated when new elements added to buffer.	MyCircularBufferStruct.StorePointer
U	UsePointer	INT	Pointer updated when elements of buffer have been read.	MyCircularBufferStruct.UsePointer
U	Size	INT	Size of circular buffer.	MyCircularBufferStruct.Size
U	CmdDelimiters	ARRAY[0..3] OF BYTE;	Specify the delimiters which separate Command Strings. If no CmdDelimiters are specified, a carriage return or carriage return line feed will be assumed.	MyCircularBufferStruct.CmdDelimiters [0]
U	PrmDelimiter	STRING	Delimiters separating parameters within a command. Default is a comma.	MyCircularBufferStruct.PrmDelimiter
U	LastDelimiter	INT	Element used by GetCommand.	MyCircularBufferStruct.LastDelimiter
U/C	Data	YTB_ByteArray8192	Array of 8192 bytes of data.	MyCircularBufferStruct.Data



Data Type: CTB_CommandStruct

For use with the [GetCommand](#) and [GetParameter](#) function block. Contains information about the communication interface used.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCommStruct	CommStruct		
C	ByteArray	YTB_ByteArray2048	Contains string data for easy parsing	MyCommandStruct.ByteArray
C	Length	INT	Length of string contained in ByteArray	MyCommandStruct.Length



Data Type: CommStruct

For use with [CommunicationChannel](#) function block. Contains information about the communication interface used.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCommStruct	CommStruct		
U	CommType	INT	Set 1 for Serial, 2 for Ethernet	MyCommStruct.CommType
U	InactivityTimeout	TIME	Use this to allow the MPic to close the socket if no communication has been received on the channel in the time specified.	MyCommStruct.InactivityTimeout
U	BufferSize	UDINT	The number of bytes to read per scan from buffer, if left at 0, the entire buffer will be transferred.	MyCommStruct.BufferSize
U	Serial	SerialConfig	Attributes required when communicating via serial interface, such as the 218-IFY1 card.	MyCommStruct.Serial.StopBits
U	Ethernet	EthernetConfig	Network attributes required when communicating via TCP/IP or UDP.	MyCommStruct.Ethernet.LocalPort



Data Type: DelimiterArray

Supporting array for [CircularBufferStruct](#)

Data Type Declaration

DelimiterArray: ARRAY[0..3] OF BYTE;



Data Type: EthernetConfig

Supporting data structure for CommStruct, contains information about Ethernet interface configuration.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyEthernetConfig	EthernetConfig		
U	LocalIPAddress	STRING	Ethernet address of controller	MyEthernetConfig.LocalIPAddress
U	LocalPort	UINT	Ethernet port number to open	MyEthernetConfig.LocalPort
C	RemoteIPAddress	STRING	Ethernet address of the device the MPiec controller is communicating with.	MyEthernetConfig.RemoteIPAddress
C	RemotePort	UINT	Port number used by the device the MPiec controller is communicating with.	MyEthernetConfig.RemotePort



Data Type: FTP_Data

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyFTP_Data	FTP_Data		
U	Username	YTB_STRING32	Username to log in to the FTP server.	MyFTP_Data.Username
U	Password	YTB_STRING32	Password to log in to the FTP server.	MyFTP_Data.Password
U	LocalIP	YTB_STRING16	Local IP of the controller.	MyFTP_Data.LocalIP
U	FTPDomain	YTB_STRING128	The domain name of the FTP server that will be resolved via DNS.	MyFTP_Data.FTPDomain
U	FTPIP	YTB_STRING16	The IP of the FTP server if a domain is not known or set.	MyFTP_Data.FTPIP
U	FTPPort	UINT	The port to connect to the FTP server through, default 21.	MyFTP_Data.FTPPort
U	DNSIP	YTB_STRING16	The DNS lookup server IP.	MyFTP_Data.DNSIP
U	DNSPort	UINT	The DNS port to connect through, the default is 53.	MyFTP_Data.DNSPort
U	Timeout	TIME	Timeout for connecting to the FTP server or data connection, default 5s.	MyFTP_Data.Timeout

Code Example

```

ftpdata.LocalIP := '192.168.1.1';
ftpdata.FTPDomain := 'ftp.example.com';
ftpdata.DNSIP := '8.8.8.8';
ftpdata.Username := 'mp2300';
ftpdata.Password := 'securepassword';
    
```



Data Type: RecipientArray

If more than 10 recipients are needed then the declaration of RecipientArray must be changed to reflect that.

Data Type Declaration

TYPE

RecipientArray : ARRAY[0..9] OF RecipientStruct;

END_TYPE



Data Type: RecipientStruct

*	Element	Data Type	Description	Usage
	MyRecipientStruct	RecipientStruct		
U	Email	YTB_STRING128		MyRecipientStruct.Email
U	Name	YTB_STRING32		MyRecipientStruct.Name



Data Type: SerialConfig

Supporting data structure for CommStruct, contains information about Serial interface configuration.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MySerialConfig	SerialConfig		
U	PortNum	UINT	For use with Y_OpenSerialPort.	MySerialConfig.PortNum
U	BaudRate	DINT	For use with Y_SetDeviceOption.	MySerialConfig.BaudRate
U	DataBits	DINT	For use with Y_SetDeviceOption.	MySerialConfig.DataBits
U	StopBits	DINT	For use with Y_SetDeviceOption.	MySerialConfig.StopBits
U	Parity	DINT	For use with Y_SetDeviceOption.	MySerialConfig.Parity
U	HandShake	DINT	For use with Y_SetDeviceOption.	MySerialConfig.HandShake
U	CalcChecksum	BOOL	For use with Y_SetDeviceOption.	MySerialConfig.CalcChecksum



Data Type: SMTP_Data

Data Type Declaration

*	Element	Data Type	Description	Usage
	MySMTP_Data	SMTP_Data		
U	DNSIP	YTB_STRING16	DNS server IP (local), used to perform lookup of mail server domain.	MySMTP_Data.DNSIP
U	DNSPort	UINT	DNS port, default is 53, leave blank unless other port is used.	MySMTP_Data.DNSPort
U	SMTPDomain	YTB_STRING128	SMTP server domain name (e.g. smtp.yourcompany.com), used for DNS lookup.	MySMTP_Data.SMTPDomain
U	SMTPIP	YTB_STRING16	The IP of the SMTP server, blank by default, provide IP to override DNS lookup.	MySMTP_Data.SMTPIP
U	SMTPPort	UINT	SMTP port, usually 25 - note: does not support SSL encrypted SMTP.	MySMTP_Data.SMTPPort
U	LocalIP	YTB_STRING16	Local IP of the controller.	MySMTP_Data.LocalIP
U	Domain	YTB_STRING128	Domain for SMTP EHLO/HELO command, example: yaskawa.com.	MySMTP_Data.Domain
U	Sender	YTB_STRING128	Sender e-mail address, example: john_smith@yaskawa.com.	MySMTP_Data.Sender
U	SenderName	YTB_STRING32	Name of sender, example: John Smith.	MySMTP_Data.SenderName
U	Subject	YTB_STRING128	Subject of the email.	MySMTP_Data.Subject
U	Recipient	RecipientArray	Array of STRING of email addresses which will receive the message.	
U	Email	YTB_STRING128	MySMTP_Data.RcptArray.[0].Email	
U	Name	YTB_STRING32	MySMTP_Data.RcptArray.[0].Name	
U	Recipients	INT	Number of emails in Recipient.	MySMTP_Data.NumRcpt
U	Timeout	TIME	Timeout for connecting to the SMTP server, defaults to 5s.	MySMTP_Data.Timeout

Code Example

```

smtpdata.LocalIP      := '192.168.1.1';
smtpdata.SMTPDomain   := 'smtp.example.com';
smtpdata.Domain       := 'example.com';
smtpdata.Sender       := 'johnsmith@example.com';
smtpdata.SenderName   := 'John Smith';
smtpdata.Subject      := 'Hello from your MP2300iec';
smtpdata.RcptArray[0].email := 'yourfriend@othercompany.com';
smtpdata.RcptArray[0].name  := 'Your Friend';
smtpdata.NumRcpt      := 1;
    
```



Enumerated Types in the Communication Toolbox

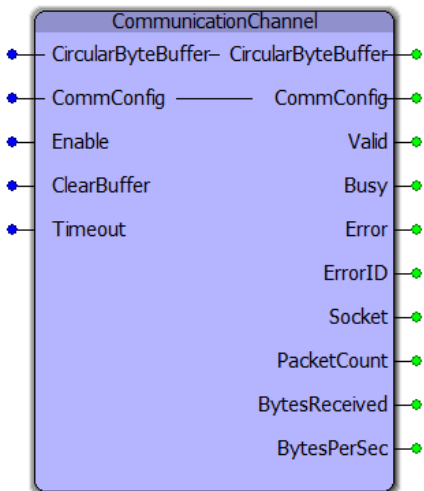
Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to zero-based integers (INT). Therefore, the first value equates to zero, the second to 1, etc. The format for enumerated types is as follows: ENUM:(0, 1, 2...) as displayed in the example below (MC_BufferMode#Aborting).

Enumerated Types Declaration

Enumerated Type	#INT Value	Enum Value	Description
COM_Type	Enumerated type to be used with CommStruct.CommType		
	0	na	
	1	Serial	
Method	2	Ethernet	
	For use with the GetParameter function. Specifies how the value is obtained.		
	0	Parameter	
	1	Character	



CommunicationChannel



The CommunicationChannel function block is designed to manage an input stream of data from either a serial or TCP socket communication interface. It collects portions of data from Y_ReadDevice each time that function’s Done output goes high, and adds it to a circular buffer for further analysis. The CommConfig structure must be initialized by the user to configure the necessary communication parameters.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	CircularByteBuffer	CircularBufferStruct	Structure containing a data buffer and other operational information required to manage the CircularByteBuffer.	
V	CommConfig	CommStruct	Structure containing information to be used in establishing socket or serial communication.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	ClearBuffer	BOOL	Resets the StorePointer and UsePointer, which logically clears the contents of the circular buffer, but any existing data is not actually cleared.	FALSE

V	Timeout	TIME	If necessary, specify a time value when after no data is received, the MPiec should close the connection. When Timeout is T#0S, the MPiec will never close the connection on its own, the host which initiated the connection can send a close command, which is the normal procedure.	T#0mS
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	Socket	DINT	File handle to be used when writing to device connected to the socket. Only valid when non-zero.	
V	PacketCount	BOOL	Total packet count since this function block was enabled.	
V	BytesReceived	UINT	Total bytes received since this function was enabled.	
V	BytesPerSec	DINT	Number of bytes received in the last second.	

Error Description

See the [Function Block ErrorID](#) list.

Setup

Follow these steps to initialize the CommConfig structure. Steps 1 & 2 show an optional easy way for the IEC application to automatically obtain its own IP Address. One of the inputs required for the Y_DeviceComm basic functions is the MPiec controllers own IP Address. This is necessary because the MPiec controller may have more than one physical Ethernet connector / MAC address, and the YDeviceComm functions need to know which interface to use. Steps 1 & 2 show a way to automatically obtain the IP address so the user will not be required to manually enter the controller's IP address for each system deployed. It is also possible to simply initialize the IPAddress as a sting such as IPAddress:='129.168.207.115';

1. Add a variable of type CONTROLLER_INFO to Global Variables as shown below. The Address must be %MD3.66560.

Name	Type	Usage	Description	Address
Controller	CONTROLLER_INFO	VAR_GLOBAL		%MD3.66560

2. Add the following code to the initialize routine to obtain controller's IP address. The variable IPAddress is a STRING. The BUF_TO_STRING function block is located in the PROCONOS firmware library. As shown below, we are using it to extract 15 bytes of the IPAddress. These bytes equate to xxx.xxx.xxx.xxx of the IP Address.

```

50 BUF_TO_STRING      (* Get the controller IP address *)
51 (
52   REQ:=TRUE,
53   BUF_FORMAT:=TRUE,
54   BUF_OFFS:=DINT#0,
55   BUF_CNT:=DINT#17,
56   BUFFER:=Controller.Network.Interface[1].IPAddress,
57   DST:=IPAddress
58 );
59 Controller.Network.Interface[1].IPAddress:=BUF_TO_STRING.BUFFER;
60 IF BUF_TO_STRING.DONE THEN
61   IPAddress:=BUF_TO_STRING.DST;
62 END_IF;

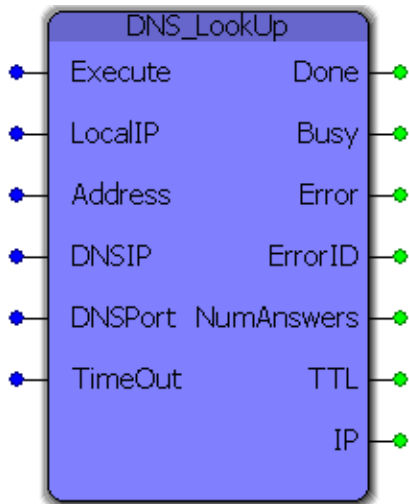
```

3. Initialize variable of data type CommStruct as shown below. Set .LocalPort to the desired connection port number used in the application. If multiple sockets will be used, ensure they each have a unique port number.

```
67 CommConfig.CommType:=COM_Type#Ethernet;  
68 CommConfig.Ethernet.LocalIPAddress:=IPAddress;  
69 CommConfig.Ethernet.LocalPort:=UINI#5000;
```




DNS_LookUp



This function block performs a DNS lookup for a provided domain name (Address) using a specified DNS IP and port and returns the number of answers, the resolved IPV4 address and the Time To Live of the returned IP.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all inputs are read and the DNS lookup is performed. To perform a lookup on a different address or perform the same lookup again, change the value and re-trigger the execute input.	FALSE
V	LocalIP	YC_STRING16	The IP address of the controller on the local network.	STRING#''
V	Address	YC_STRING128	The domain name to perform the look-up on (not an IPV4 address).	STRING#''
V	DNSIP	YC_STRING16	The IP address of the DNS server to perform the lookup through.	STRING#''
V	DNSPort	UINT	The port to connect to the DNS server through.	UINT#53
E	TimeOut	TIME	The amount of time the DNS server has to respond.	TIME#5s

VAR_OUTPUT			
B	Done	BOOL	Set high upon the completion of a successful DNS lookup.
B	Busy	BOOL	Set high upon the rising edge of 'Execute' and reset if Done or Error is true.
B	Error	BOOL	Set high if an error has occurred during the DNS lookup. Cleared upon 'Execute' being reset.
E	ErrorID	UINT	If error is true, this output provides the Error ID. Cleared upon 'Execute' being reset.
V	NumAnswers	INT	The number of answers returned by the DNS server. The answer with the longest TTL is output at 'IP'
V	TTL	UDINT	The Time To Live of the DNS response (i.e. how long the DNS server caches the answer from the authoritative nameserver instead of reissuing the query).
V	IP	YC_STRING16	The 'IP' with the longest TTL that was returned by the DNS server that resolves to the domain name provided.

Notes

- 'Address' must be a domain name (i.e. www.yaskawa.com), not an IPV4 address. Passing an IPV4 address is referred to as a "reverse DNS lookup" and is not supported by this function.
- The DNS server(s) the MPiec controller has access to depends on the network configuration. If there is no access to a local DNS server (see "Setup" below) talk to your IT professional about DNS server options.
- The main purpose of this block is for use in other Communications blocks, such as FTP and SMTP.

Setup

To perform a DNS lookup, a connection to a DNS server must be established. The easiest way to determine what DNS server to use is to open up the Windows command prompt (Windows Key + R -> "cmd" -> Enter) and type "ipconfig /all" and under "DNS Servers" in the Ethernet LAN section you will find the DNS server(s) that the computer is configured to use.


```

C:\WINDOWS\system32\cmd.exe
yaskawa.com
ybad.ad.yaskawa.com
ybad.com
yedev.com
drives.com

Ethernet adapter VMware Network Adapter VMnet8:

    Connection-specific DNS Suffix  . : 
    Description . . . . . : VMware Virtual Ethernet Adapter for
VMnet8
    Physical Address. . . . . : 00-50-56-C0-00-08
    Dhcp Enabled. . . . . : No
    IP Address. . . . . : 192.168.214.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

Ethernet adapter VMware Network Adapter VMnet1:

    Connection-specific DNS Suffix  . : 
    Description . . . . . : VMware Virtual Ethernet Adapter for
VMnet1
    Physical Address. . . . . : 00-50-56-C0-00-01
    Dhcp Enabled. . . . . : No
    IP Address. . . . . : 192.168.88.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

Ethernet adapter Wireless Network Connection:

    Media State . . . . . : Media disconnected
    Description . . . . . : Intel(R) WiFi Link 5100 AGN
    Physical Address. . . . . : 00-24-D6-77-02-00

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : ad.yaskawa.com
    Description . . . . . : Intel(R) 82567LM Gigabit Network Con
nection
    Physical Address. . . . . : 00-26-B9-97-2F-4A
    Dhcp Enabled. . . . . : Yes
    Autoconfiguration Enabled . . . . : Yes
    IP Address. . . . . : 192.168.201.36
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.201.253
    DHCP Server . . . . . : 192.168.5.10
    DNS Servers . . . . . : 192.168.5.10
    . . . . . : 192.168.5.11
    Lease Obtained. . . . . : Wednesday, October 24, 2012 8:21:53
AM
    Lease Expires . . . . . : Thursday, October 25, 2012 8:21:53 AM
F:\>

```

You can also perform DNS lookups from the command line which may help in verifying the results of the DNS lookup performed on the controller while setting this block up.

```
C:\WINDOWS\system32\cmd.exe

F:\>nslookup athena.yaskawa.com
Server: hqdc1.ad.yaskawa.com
Address: 192.168.5.10

Non-authoritative answer:
Name: athena.yaskawa.com
Address: 192.168.8.3

F:\>nslookup nothing.yaskawa.com
Server: hqdc1.ad.yaskawa.com
Address: 192.168.5.10

*** hqdc1.ad.yaskawa.com can't find nothing.yaskawa.com: Non-existent domain

F:\>nslookup google.com
Server: hqdc1.ad.yaskawa.com
Address: 192.168.5.10

Non-authoritative answer:
Name: google.com
Addresses: 74.125.225.131, 74.125.225.128, 74.125.225.130, 74.125.225.132
          74.125.225.134, 74.125.225.142, 74.125.225.135, 74.125.225.133, 74.125
          .225.136
          74.125.225.137, 74.125.225.129

F:\>_
```

The basic command structure is "nslookup [hostname] [server]" where hostname and server are both optional (if you simply type "nslookup" -> Enter it takes you in to the nslookup utility where you can then perform multiple lookups without retyping "nslookup"). For example, typing "nslookup google.com" as in the image above returns a list of IP addresses resolved for "google.com". You can also perform the lookup using a specified DNS server address which can be helpful if your block is using a different DNS server than your computer is configured to use. This is done by filling in the second optional parameter, such as "nslookup google.com 8.8.8.8" where "8.8.8.8" is a public DNS server managed by Google.

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\kevin_hull>nslookup google.com 8.8.8.8
Server: google-public-dns-a.google.com
Address: 8.8.8.8

Non-authoritative answer:
Name: google.com
Addresses: 74.125.225.136, 74.125.225.134, 74.125.225.128, 74.125.225.130
          74.125.225.135, 74.125.225.131, 74.125.225.132, 74.125.225.142, 74.125
          .225.133
          74.125.225.129, 74.125.225.137

C:\Documents and Settings\kevin_hull>_

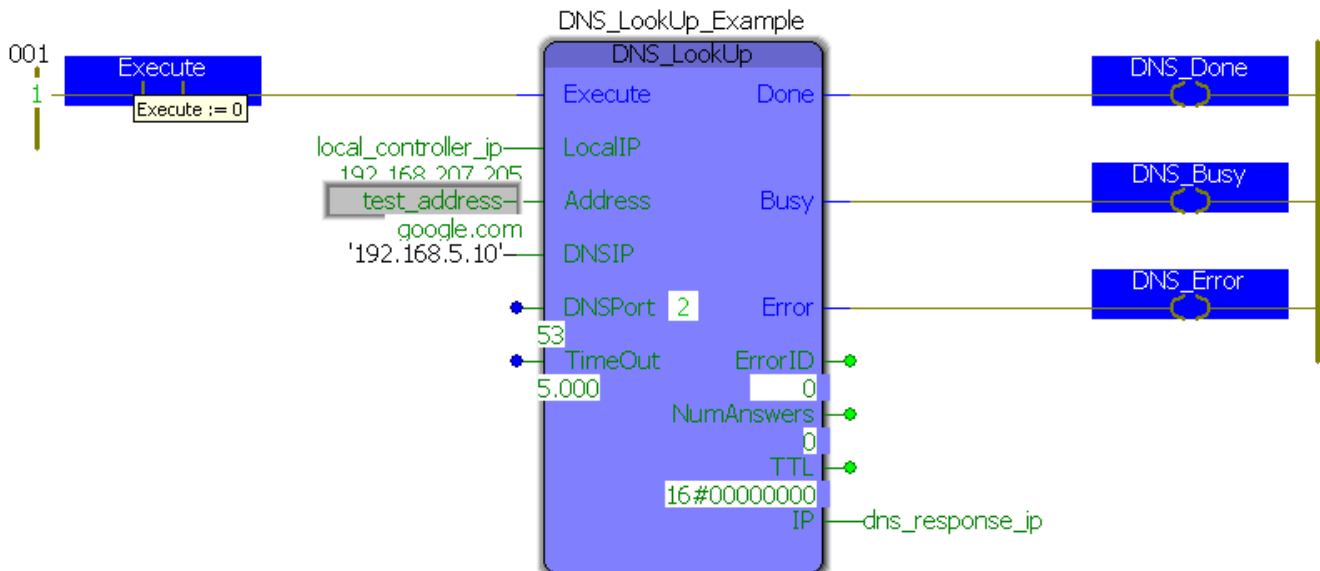
```

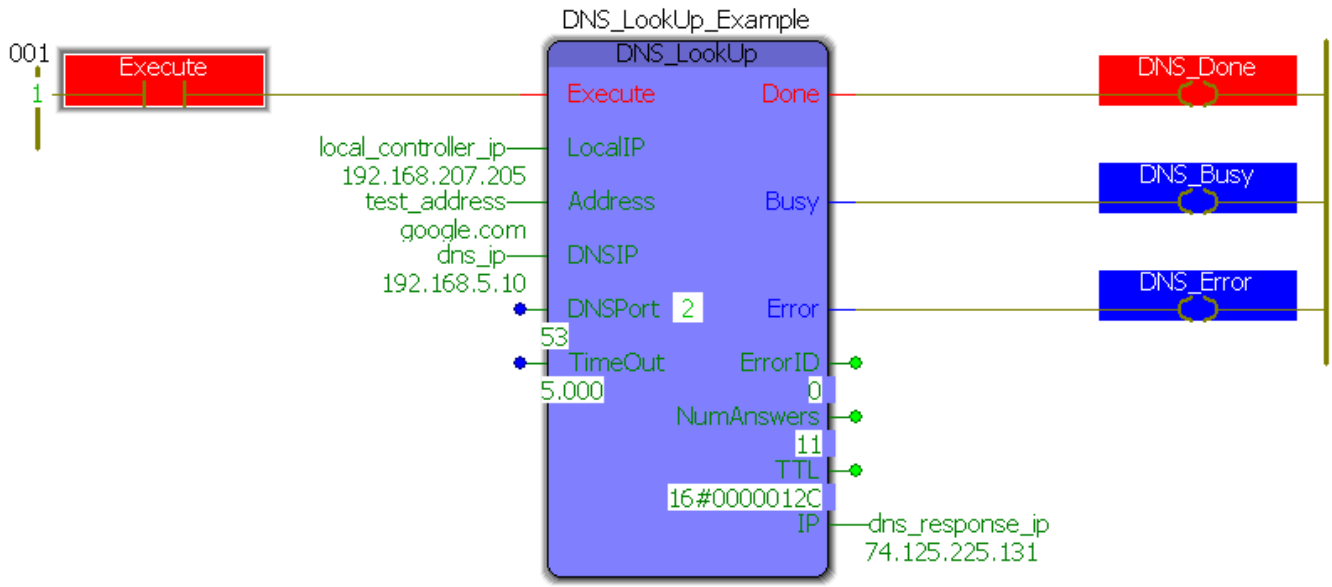
Error Description

See the [Function Block ErrorID](#) list.

Example - External Address

The following example demonstrates the blocks ability to perform a look up for an external address ("google.com") using an internal DNS server. The LocalIP, Address and DNSIP have all be configured and DNSPort and TimeOut have been left to defaults.

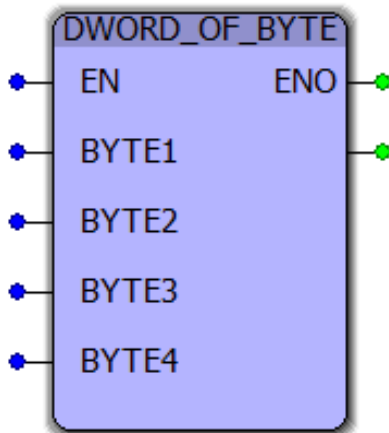




When comparing the output of the block ("74.125.225.131") to the dnslookup performed above, notice that the IP address is in the list. You can also see that NumAnswers is set to 11 which matches the number of answers returned above. Finally, the TTL is 0x0000012C which corresponds to 300 in decimal where 300s = 5 min, if you were to add the "Debug" option to nslookup ("nslookup -d google.com") then you would see that this TTL also matches.



DWORD_OF_BYTE



This function combines four bytes into a DWORD. BYTE1 is the least significant portion of the resulting DWORD.

Library

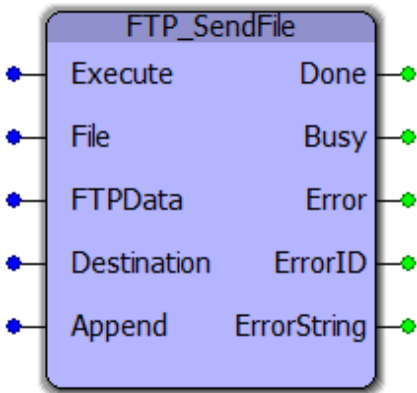
Comm Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_INPUT			
B	EN	BOOL	Enables the function.
V	BYTE1	BYTE	The least significant byte of the DWORD to be assembled.
V	BYTE2	BYTE	
V	BYTE3	BYTE	
V	BYTE4	BYTE	The most significant byte of the DWORD to be assembled.
VAR_OUTPUT			
B	ENO	BOOL	High if the function is executing normally.



FTP_SendFile



This function block uses the FTP (File Transfer Protocol) to write a file to a specified FTP server. The File must already exist on the MPiec Controller's Flash or ramdisk for this function block to read and transfer data to the Destination FTP server. The File must be created by some other method, or refer to [ReName_FTP_SendData](#) which doesn't require a pre existing file.

Library

Comm Toolbox

Parameters

* Parameter	Data Type	Description	Default
VAR_INPUT			
B Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V File	YTB_STRING128	The full file name and location on the controller, e.g. '/flash/user/data/example.csv'.	STRING#''
V FTPData	FTP_Data	The input structure that configures the FTP transfer such as FTP server address, port, etc.	All zeros in structure
V Destination	YTB_STRING64	The full file name and destination on the FTP server, e.g. 'metrics/example.csv'.	STRING#''
V Append	BOOL	To select whether an existing the file on the server should be deleted and rewritten or the new data should be appended to the existing data.	FALSE
VAR_OUTPUT			

B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	BOOL	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	ErrorString	YTB_STRING256	If 'Error' is true and it is an FTP response code related error then this output contains the response string from the FTP server.

Notes

- This block utilizes FTP, not SFTP because SSL is not supported by the MPiec firmware. As a result, all FTP traffic sent and received (e.g. username, password, file data) is sent unencrypted in plain text and is visible to anyone with access to the network. This should not be a problem if the data being sent is not of a sensitive matter and the FTP server account is CHROOT'd properly (talk to your IT professional about using FTP).
- The FTP server should either have an internal/external domain name or use a static IP address because if the address changes, it will prevent the function from transferring files. See "Setup" for more details.
- The FTP user account must have "Write" privileges to successfully write files to the server. Optionally, the account may also have "Append" privileges. If files already exists and the FTP account only has "Write" privileges, then the file will be overwritten. If the file exists and the user account has "Append" privileges, then the file contents transferred will be appended to the existing file.

Error Description

See the [Function Block ErrorID](#) list.

Basic Functionality Example - Transferring a File

This examples demonstrates how to configure the function using the data structure, create a file to send, and execute the FTP_SendFile block.

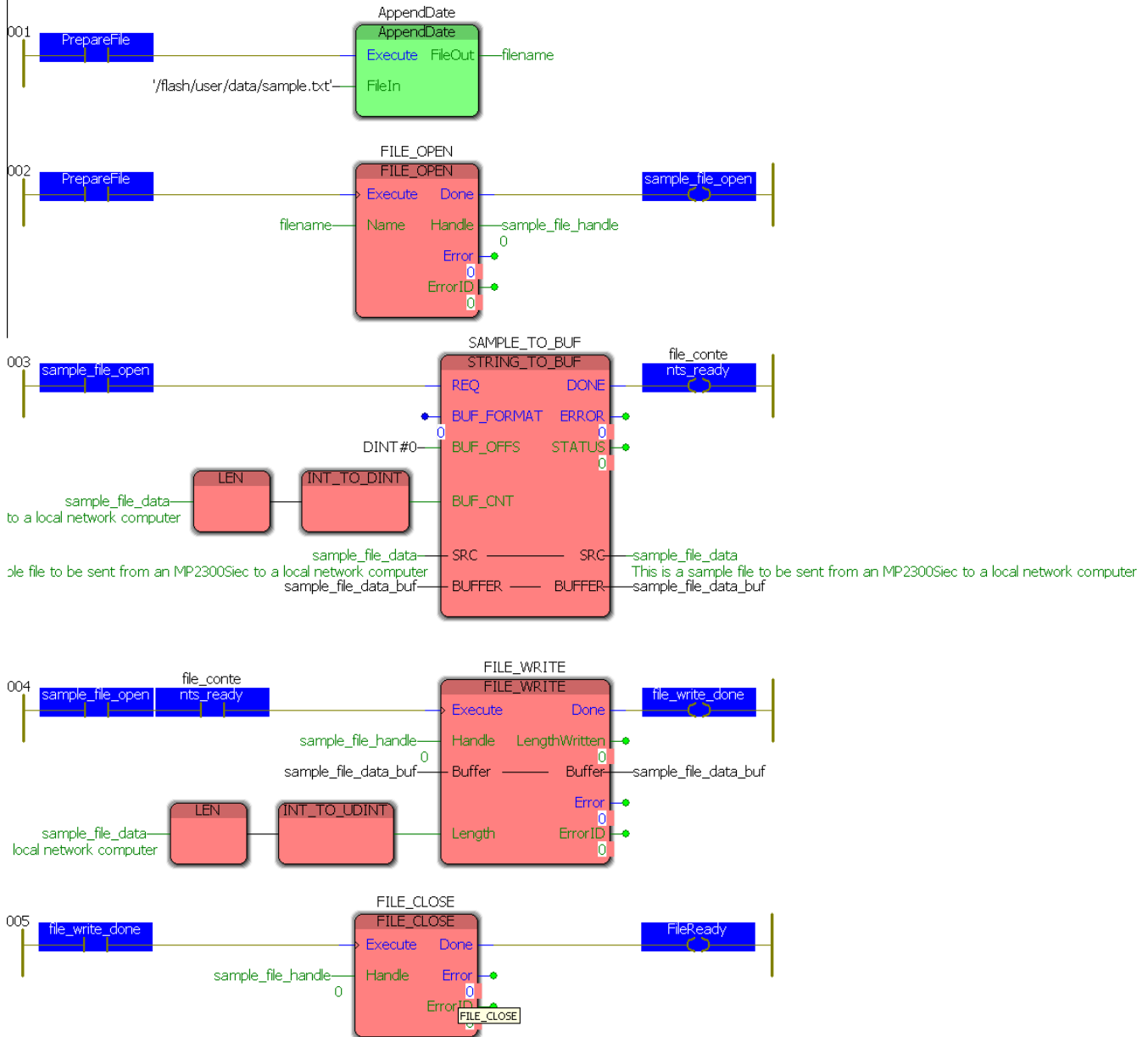
Here is the code in the "Initialize" ST program which configures the file data and the FTP structure. The FTP server is hosted on a local computer and does not have a domain name. Therefore, FTPIP was used and FTPPort was left blank as the local FTP server is configured to use the default port of 21. The LocalIP is set to the controllers IP and the username/password combination are set.

```
(* Sample file contents *)
sample_file_data := 'This is a sample file to be sent from an MP2300Siec to a local network computer via FTP';

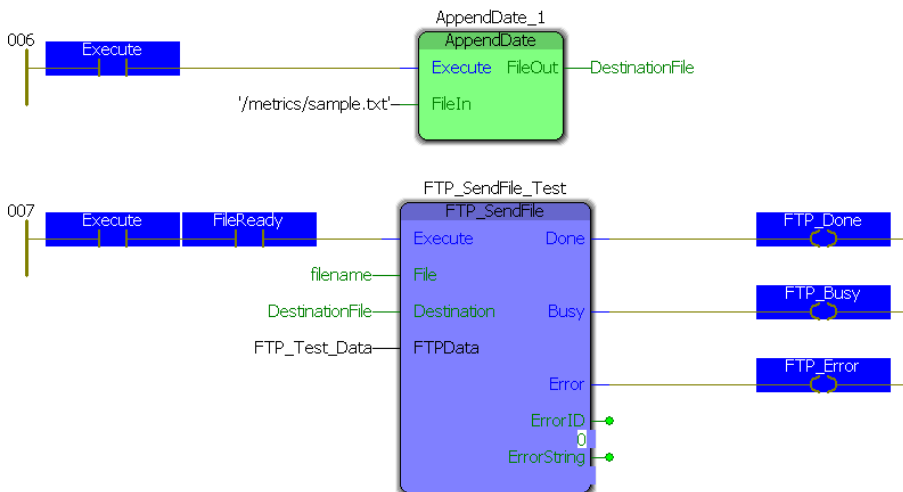
(* FTP setup structure *)
FTP_Test_Data.FTPIP := '192.168.201.36';
FTP_Test_Data.LocalIP := '192.168.207.205';
FTP_Test_Data.Password := 'anon';
FTP_Test_Data.Username := 'anon';
```

This program creates a file via the PROCONOS FILE_OPEN, STRING_TO_BUF, FILE_WRITE and FILE_CLOSE functions. The contents of the file in "sample_file_data" is converted from a YC_STRING128 to YC_BYTE128 via the "SAMPLE_TO_BUF" block. Once the file is created, the destination file name is prepared and the FTP function sends the file to the server.

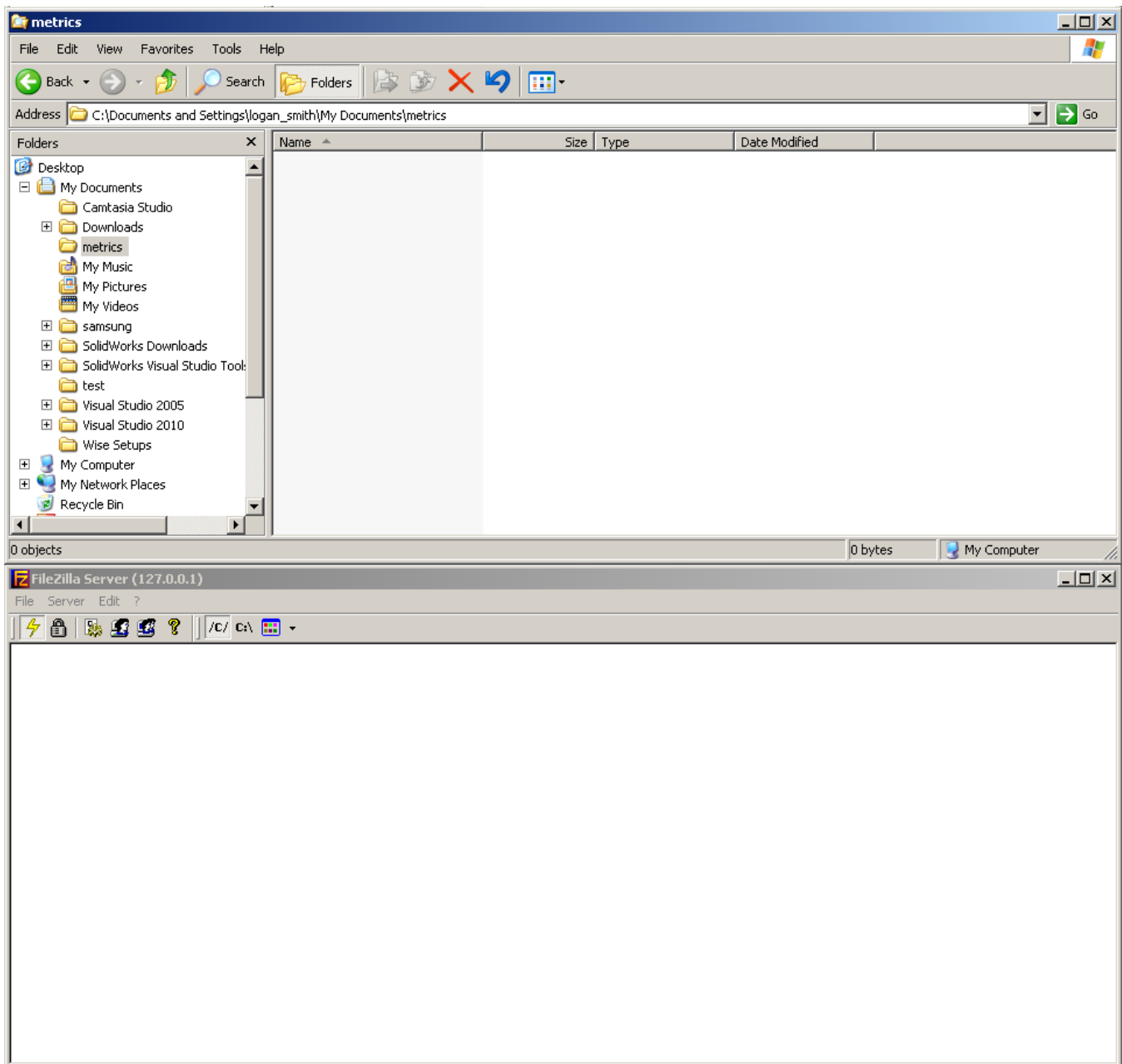
(* Prepare file to send *)



(* Send example.txt via FTP *)

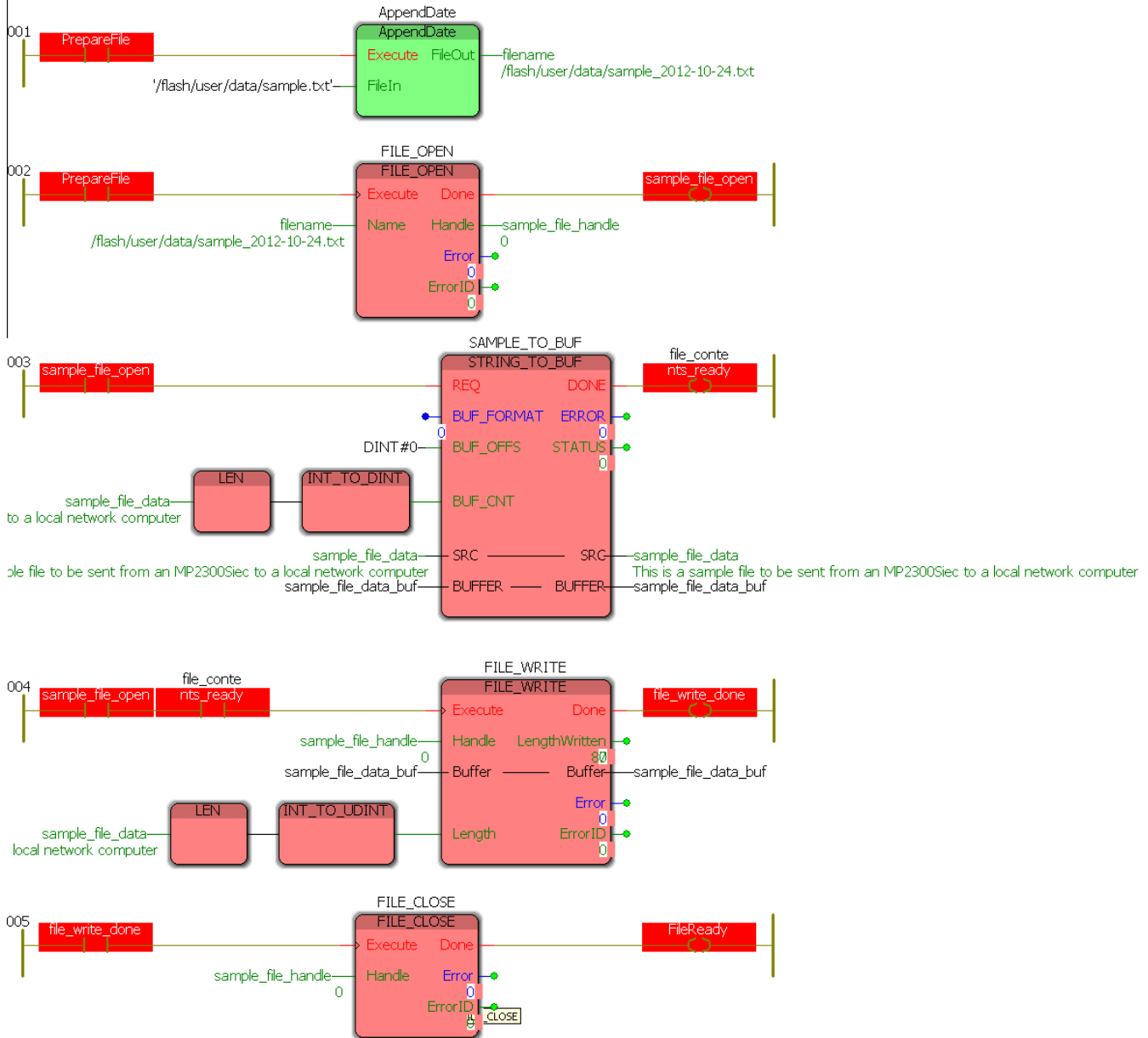


The destination folder is empty to begin with and the FTP server log has been cleared prior to connection so that the results will be obvious.

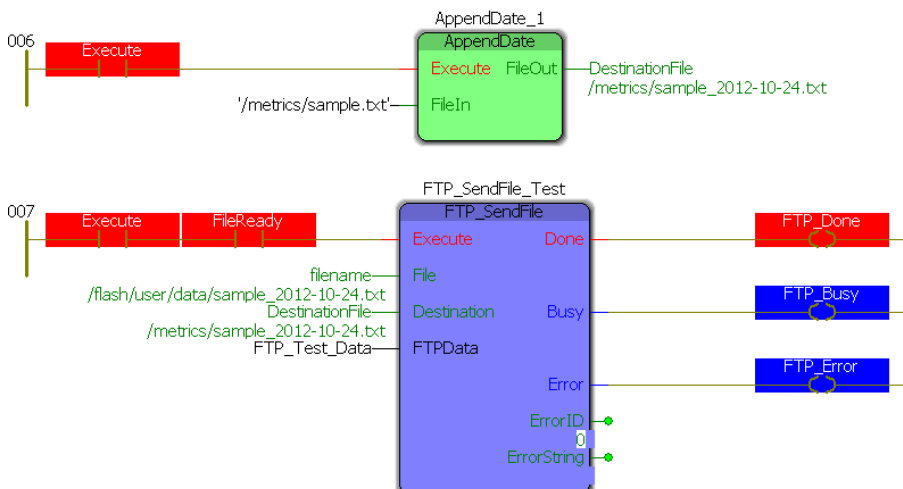


The PrepareFile contact is set true as is the Execute contact. Once both contacts are TRUE, the FTP_SendFile block sends the newly created file.

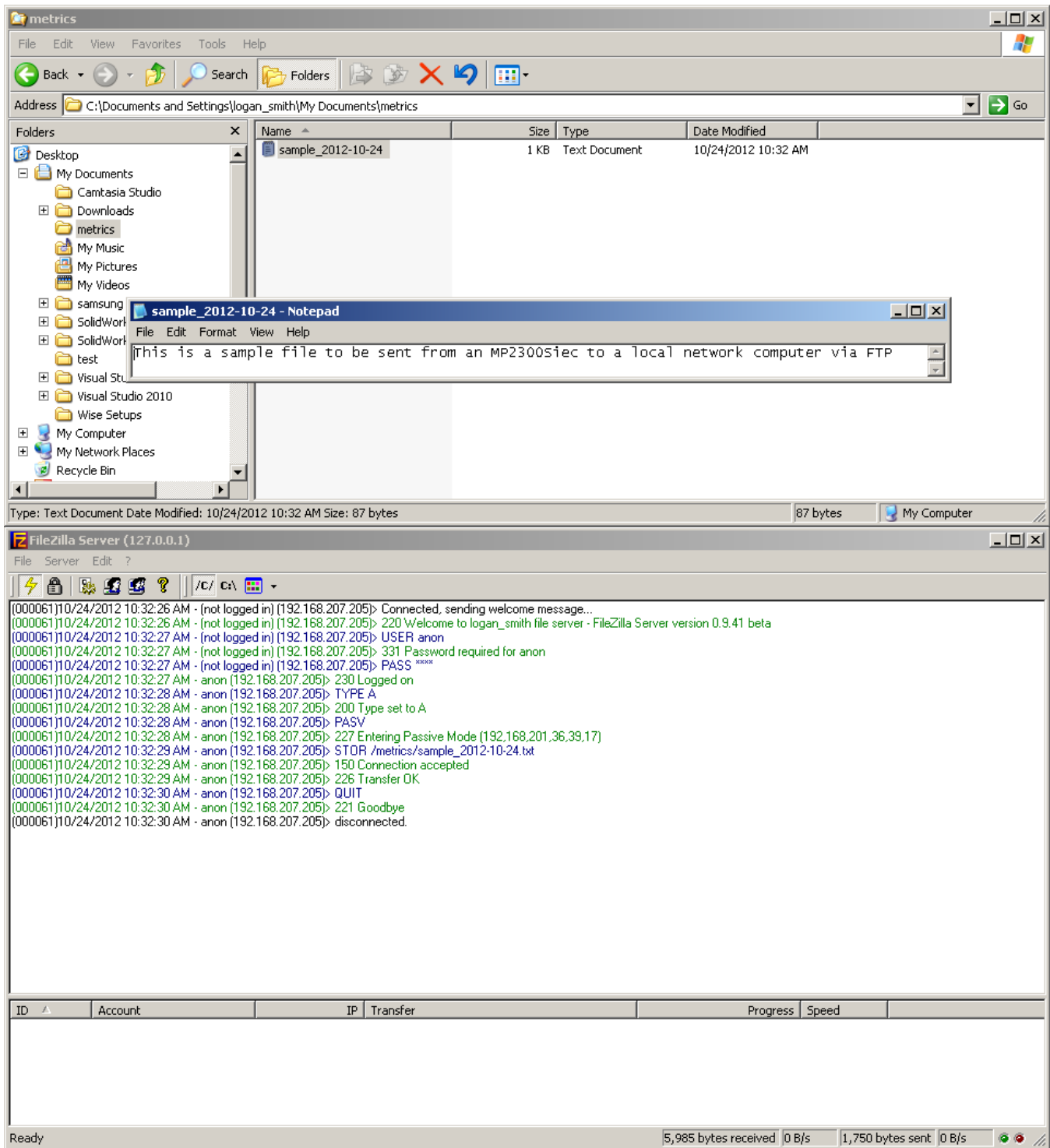
(* Prepare file to send *)



(* Send example.txt via FTP *)



The results of this block can be seen in the destination file explorer and the FTP server log:



The contents of the file match the "sample_file_data" string and the file can be seen in the explorer. In the FTP server log all of the commands sent can be viewed and it can be seen that the file was transferred properly and successfully.

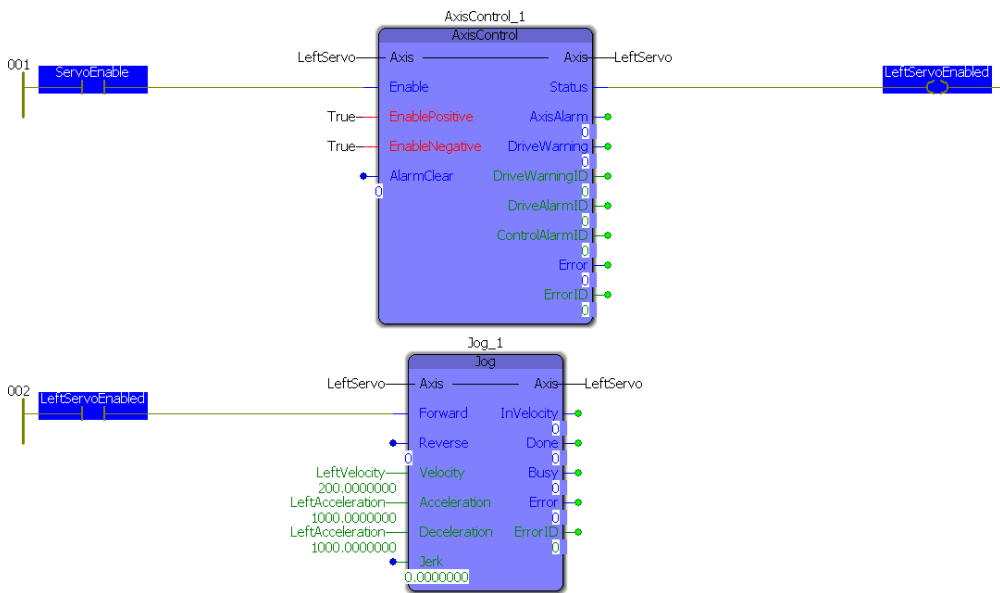
Advanced Functionality Example - Transferring a Metrics File at a Specified Rate

This examples demonstrates how to write a program to send a continuously updated metrics file (with date and time stamp) to an FTP server. This kind of functionality is extremely useful to applications requiring data acquisition as the need to connect to the controller directly is eliminated and file management is handled by the controller. For this example, the controller will continuously sample the speed and position of a servo that is jogging and the store the contents in a CSV file using the File_RW Toolbox.

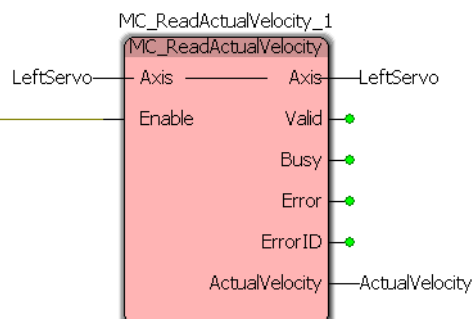
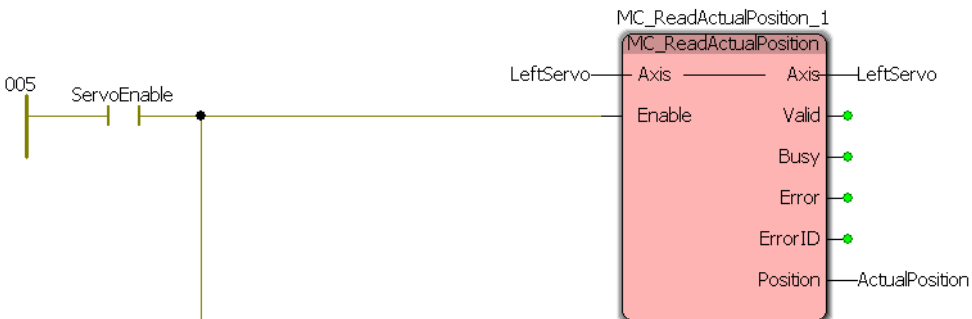
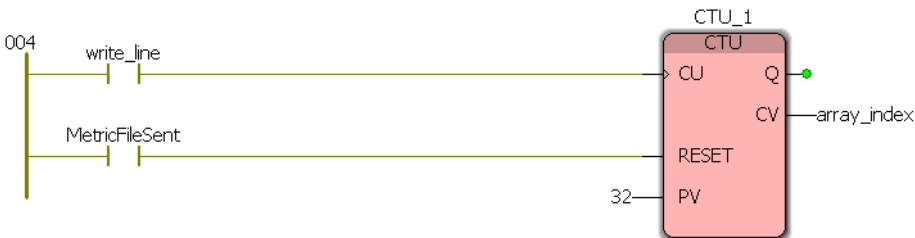
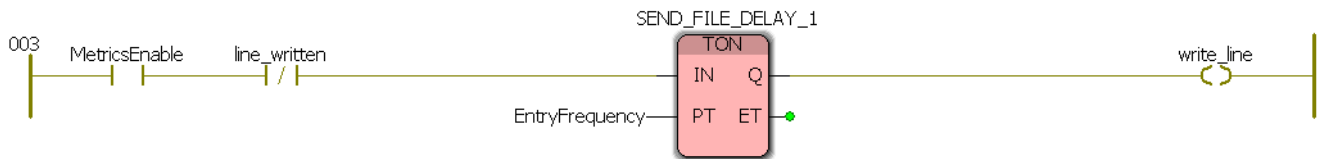
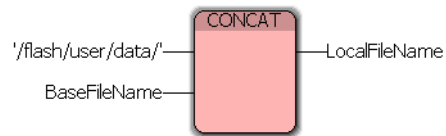
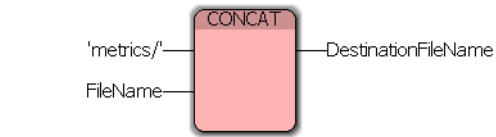
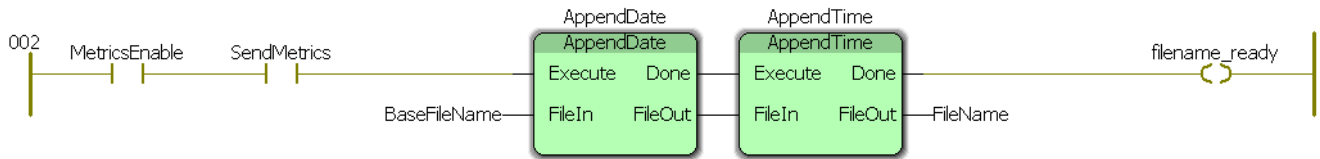
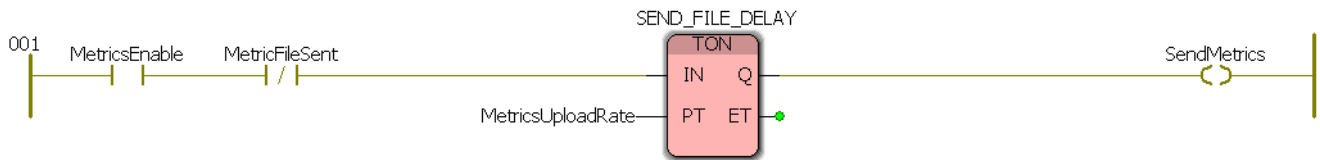
The same data configuration structure was used but there is no preset message for the file as it will be created dynamically.

```
(* FTP setup structure *)
FTP_Test_Data.FTPIP := '192.168.201.36';
FTP_Test_Data.LocalIP := '192.168.207.205';
FTP_Test_Data.Password := 'anon!';
FTP_Test_Data.Username := 'anon!';
```

In addition to the Communications Toolbox, two additional Yaskawa toolboxes are used: File_RW_Toolbox and PLCOpen_Toolbox. The File_RW_Toolbox is used to create the CSV file that is uploaded to the FTP server and the PLCOpen_Toolbox is used to control the single servo used in this example.
















Controlling this example is very simple. The servo is turned on by "ServoEnable" which then in turn starts the jog at a constant velocity. The rest of the example is controlled in the main program:



This entire program is enabled by the "MetricsEnable" contact which starts two timers: the 30 second timer which sends the CSV file and the 1 second timer which takes a sample of the current position and velocity of the servo. The filename is generated each time the file is uploaded so that the timestamp is up to date and no files are overwritten.

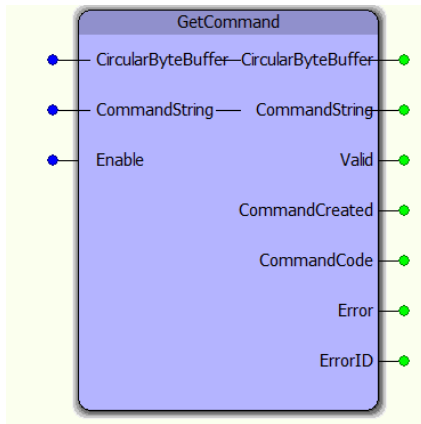
The results of this example can be monitored by exploring the target upload directory and examining the FTP server log:

```
(000074)11/12/2012 16:46:02 PM - (not logged in) (192.168.207.205)> Connected, sending welcome message...
(000074)11/12/2012 16:46:02 PM - (not logged in) (192.168.207.205)> 220>Welcome to logan_smith file server - FileZilla Server version 0.9.41 beta
(000074)11/12/2012 16:46:02 PM - (not logged in) (192.168.207.205)> USER anon
(000074)11/12/2012 16:46:02 PM - (not logged in) (192.168.207.205)> 331 Password required for anon
(000074)11/12/2012 16:46:02 PM - (not logged in) (192.168.207.205)> PASS *****
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> 230 Logged on
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> TYPE A
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> 200 Type set to A
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> PASV
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> 227 Entering Passive Mode (192,168,201,36,39,23)
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> STOR metrics/data_2012-11-12_17-44-18.csv
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> 150 Connection accepted
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> 226 Transfer OK
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> QUIT
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> 221 Goodbye
(000074)11/12/2012 16:46:02 PM - anon (192.168.207.205)> disconnected.
```

Name	Size	Type	Date Modified
 data_2012-11-12_17-35-40	1 KB	Microsoft Office Exc...	11/12/2012 4:37 PM
 data_2012-11-12_17-36-10	1 KB	Microsoft Office Exc...	11/12/2012 4:37 PM
 data_2012-11-12_17-36-41	1 KB	Microsoft Office Exc...	11/12/2012 4:38 PM
 data_2012-11-12_17-37-11	1 KB	Microsoft Office Exc...	11/12/2012 4:38 PM
 data_2012-11-12_17-37-42	1 KB	Microsoft Office Exc...	11/12/2012 4:39 PM
 data_2012-11-12_17-38-12	0 KB	Microsoft Office Exc...	11/12/2012 4:39 PM
 data_2012-11-12_17-38-43	1 KB	Microsoft Office Exc...	11/12/2012 4:40 PM
 data_2012-11-12_17-39-13	1 KB	Microsoft Office Exc...	11/12/2012 4:40 PM
 data_2012-11-12_17-39-44	1 KB	Microsoft Office Exc...	11/12/2012 4:41 PM
 data_2012-11-12_17-40-14	1 KB	Microsoft Office Exc...	11/12/2012 4:42 PM
 data_2012-11-12_17-40-45	1 KB	Microsoft Office Exc...	11/12/2012 4:42 PM
 data_2012-11-12_17-41-15	1 KB	Microsoft Office Exc...	11/12/2012 4:43 PM
 data_2012-11-12_17-41-46	0 KB	Microsoft Office Exc...	11/12/2012 4:43 PM



GetCommand



The GetCommand function block is a supporting function block for the ReName_CommandProcessor function block. It extracts a CommandString from the CircularByteBuffer as identified by the CmdDelimiter specified in the CircularByteBuffer structure.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	CircularByteBuffer	CircularBufferStruct	Structure containing a data buffer and other operational information required to manage the CircularByteBuffer.	
V	CommandString	CTB_CommandStruct	Input string containing at least two bytes of command characters and any optional parameters separated by a PrmDelimiter.	
VAR_INPUT			Default	
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	CommandCreated	BOOL	Indicates that the CommandString VAR_IN_OUT contains a new CommandString.	
V	CommandCode	INT	Integer value corresponding to the first two ASCII characters of the CommandString.	

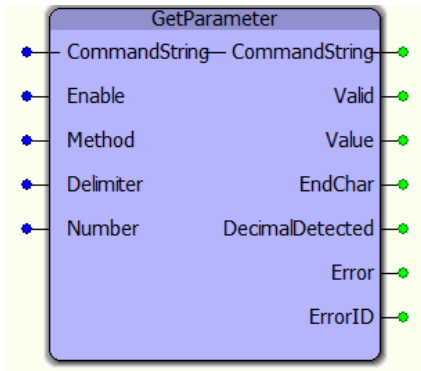
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Error Description

See the [Function Block ErrorID](#) list.



GetParameter



The GetParameter function block provides a single parameter Value extracted from the CommandString. This is supporting function block for use within the CommandProcessor function block.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	CommandString	CTB_CommandStruct	Input string containing parameters separated by delimiters. such as MV;1.0;-10.5;3.007	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Method	Method	Determines method used to retrieve variable from data buffer. Method#Parameter uses the input Number to determine the parameter number within the string to output. Method#Character uses the number input as the index in the byte array to start looking for the next valid parameter.	Method#Parameter
V	Delimiter	YTB_STRING1	String value of the character separating parameters within the CommandString.	BYTE#44 (" , ")
V	Number	INT	Depending on Method input, either the number of the parameter value to be found.	INT#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	

V	Value	STRING	Value of the parameter.
V	EndChar	INT	Last character index in byte buffer searched. When using Method#
V	DecimalDetected	BOOL	Indicates if the Value output contains a decimal. (Useful to determine conversion to a REAL or INT.)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

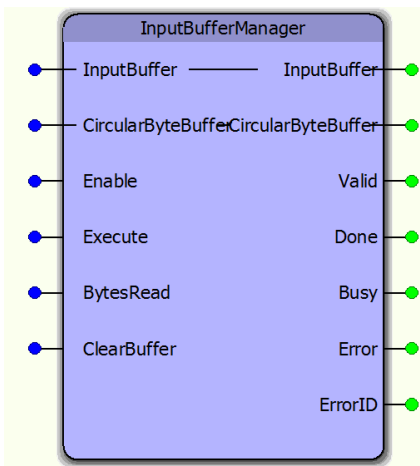
- There are two methods available with this function block; Values can be fetched via Parameter (Delimiter) count or by StartCharacter. The Parameter method always counts delimiters from the beginning of the CommandString to explicitly return the correct Value. If this Function block is executed in WHILE loop situation, it is more efficient to specify the next StartCharacter as the Number Input by feed the previous EndChar back into the function block.
- If Method = Method#Parameter, GetParameter will search through the command string to find the parameter corresponding to the Number input. This method is useful for commands with fewer parameters or when parameters are being read non-sequentially.
 - Example: CommandString = 'MV,2,4,6' Delimiter = ',' Number = 2
When Valid = TRUE, Value = 4
- If Method = Method#Character, GetParameter will search the command string for the next parameter starting at the character location equal to the Number input. The EndChar output can be used as feedback to the Number input to find the next parameter. This method is useful when parameters are being read sequentially and provides a large performance increase when parsing a CommandString with a large number of parameters.
 - Example: CommandString = 'MV,2,4,6' Delimiter = ',' Number = 5
When Valid = TRUE, Value = 4, EndChar = 7
- Further examples of both methods are provided in [ReName_CommandProcessor](#) customization section.

Error Description

See the [Function Block ErrorID](#) list.



InputBufferManager



The InputBufferManager function block manages a circular buffer of incoming data. It is a supporting function block for the CommunicationChannel function block. A user should not need to access this function directly.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	InputBuffer	YTB_ByteArray2048	Byte array containing data to be copied into the CircularByteBuffer.	
V	CircularByteBuffer	CircularBufferStruct	Structure containing a data buffer and other operational information required to manage the CircularByteBuffer.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	INT # 0
V	BytesRead	UDINT	Number of bytes to be copied from InputBuffer to CircularByteBuffer.	UDINT # 0

V	ClearBuffer	BOOL	Resets the StorePointer and UsePointer, which logically clears the contents of the circular buffer, but any existing data is not actually cleared.	INT#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

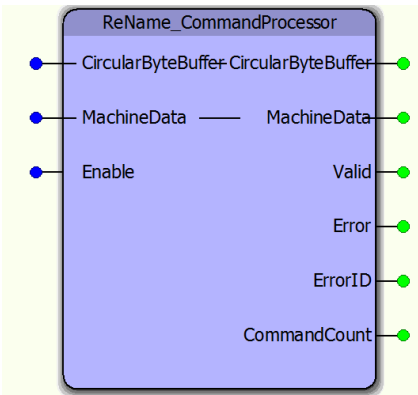
This is a hybrid function block that incorporates both PLCopen specified behaviors: Enable and Execute. This was mainly done to separate two types of initialization: one that occurs when the Enable goes high, and another that occurs only when the Execute goes high.

Error Description

See the [Function Block ErrorID](#) list.



ReName_CommandProcessor



The ReName_CommandProcessor function block is a user customizable function block that parses data from a circular buffer and copies it into a user defined structure of data to operate the machine. To use this function, copy and paste it into the main project, rename it, and customize it and the MachineData structure. It's designed to identify variable length commands in the CircularByteBuffer and process them on a case by case basis.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	CircularByteBuffer	CircularBufferStruct	A string of characters such as MV;1.0;-10.5;3.007	
V	MachineData	User Defined	Create a user defined type which suits the needs of the application.	
VAR_INPUT			Default	
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	CommandCount	UDINT	Reports the number of commands processed since Enable was set high.	

Notes

- This function block is a template for designing a unique command line interpreter and requires customization. See the customization steps below.
- The command streaming tools provided in the Comm Toolbox are designed to interpret commands starting with a two character (two byte) command code followed by either delimiter separated parameters or no parameters. The reason for this is because two ASCII bytes can easily be converted to an INT, which is used with the CASE statement in this function block. Example commands are located in the customization steps below.

Error Description

See the [Function Block ErrorID](#) list.

Customization Steps

1. Copy this Function block from the Comm Toolbox, paste it into your project, and rename with a different (but similar) name.
2. Data type MyMachineStruct (VAR_IN_OUT 'MachineData') is only an example structure. A custom structure must be designed to uniquely match the needs of the application. An example is shown below.

```
223     PositionArray  : ARRAY[1..50] OF LREAL;
224
225     CommandStruct: STRUCT
226         Enable:BOOL;
227         HomeReg:BOOL;
228         StartMoveRelative:BOOL;
229         MoveRelativeSpeed:LREAL;
230         MoveRelativeAccel:LREAL;
231         MoveRelativeDist :LREAL;
232     END_STRUCT;
233
234     Monitor: STRUCT
235         Position: LREAL;
236         Velocity: LREAL;
237         Torque: LREAL;
238     END_STRUCT;
239
240     MotorDataStruct: STRUCT
241         Num:AXIS_REF;
242         Command: CommandStruct;
243         Monitor: Monitor;
244         LoadPosition: PositionArray;
245     END_STRUCT;
246
247     MotorDataArray : ARRAY[1..5] OF MotorDataStruct;
248
249     MachineInfo: STRUCT
250         Estop           :BOOL;
251         ClearAlarms    :BOOL;
252         RunMode        :INT;           (* machine running state *)
253         Conveyer       : MotorDataStruct;
254         Arm             : MotorDataArray;
255     END_STRUCT;
```

3. Change the 'MachineData' DataType in the CommandProcessor function block to match your structure name.

MachineData	MachineInfo	VAR_IN_OUT
-------------	-------------	------------

- Initialize the configuration elements in CircularByteBuffer.

```

67 CBBuffer.CmdDelimiters[0] := BYTE#13;
68 CBBuffer.Size := INT#8192;
69 CBBuffer.PrmDelimiter := ',';

```

- CmdDelimiters are used to mark the end of a complete command. Up to four characters can be specified. Typically, <cr>, which is BYTE#13 or <cr><lf>, which is BYTE#13 BYTE#10 are used. If CmdDelimiters not specified, will default functionality will automatically accept Carriage Return or Carriage Return & Line Feed.
 - PrmDelimiter specifies the character that separates individual parameters within a command. If PrmDelimiter is not specified, the function will automatically default to a comma, (BYTE#44).
 - Size must represent the defined size of the DataType definition for the CircularBufferStruct's "Data" Element. If Size not specified, it will default to zero and the InputBufferManager function block will cause an error. Normally, this value is 8192 as the structure definition is in the Comm Toolbox itself. If this must be increased for any reason, modify the Comm Toolbox DataType definition and set the Size input accordingly.
- Locate the comments "Customize the code below" and "Customize the code above"
- Remove example commands to avoid potential errors in operation.

```

131  (*****
132  (*****                                     Customize the code below
133  (*****
134
135  CASE CommandCode OF
136
137      (* insert new commands here *)
138
139  ELSE
140      Error_UnsupportedCommand:=TRUE;
141  END_CASE; (* CommandCode *)
142
143  (*****
144  (*****                                     Customize the code above
145  (*****

```

- Add commands. Two examples are shown below:

- Move Relative command:

- MR,<axisnumber>,<distance>,<speed>,<accel/decel>
- Calculate the CommandCode which corresponds to the ASCII characters 'MR'. The equation is: $CHAR_TO_INT('M') * 256 + CHAR_TO_INT('R') = 19794$.
- Add the CommandCode to the case statement.
- Use the GetParameter function block to separate command parameters. The example below uses GetParameter with "Method#Parameter?"

```

19794 : (* MR - Move Relative *);
FOR ParameterIndex := 1 TO 4 DO
  GetParameter.CommandString:=CommandString;
  GetParameter(Number:=ParameterIndex, Method := Method#Parameter);
  CommandString:=GetParameter.CommandString;
  IF ( GetParameter.Valid := TRUE ) THEN
    CASE ParameterIndex OF
      1: AxisNum := STRING_TO_INT(GetParameter.Value);
      2: MachineData.Arm[AxisNum].Command.MoveRelativeDist := STRING_TO_LREAL(GetParameter.Value);
      3: MachineData.Arm[AxisNum].Command.MoveRelativeSpeed:= STRING_TO_LREAL(GetParameter.Value);
      4: MachineData.Arm[AxisNum].Command.MoveRelativeAccel:= STRING_TO_LREAL(GetParameter.Value);
        MachineData.Arm[AxisNum].Command.StartMoveRelative:= TRUE;
    END_CASE;
  END_IF;
END_FOR;

```

- Load Positions command:

- LP,<Position1>,<Position2>,...,<Position50>
- Calculate the CommandCode which corresponds to the ASCII characters 'LP'. The equation is: $CHAR_TO_INT('L') * 256 + CHAR_TO_INT('P') = 19536$
- Add the CommandCode to the case statement.

4. Use the GetParameter function block to separate command parameters. The example below uses GetParameter with "Method#Character"

```
19536 : (* LP - Load Positions *)
CharacterIndex := 0;
FOR PositionCount := 1 TO 50 DO
  GetParameter.CommandString:=CommandString;
  GetParameter(Number:=CharacterIndex, Method := Method#Character);
  CommandString:=GetParameter.CommandString;
  CharacterIndex:= GetParameter.EndChar;
  IF ( GetParameter.Valid := TRUE ) THEN
    MachineData.Conveyor.LoadPosition[PositionCount] := STRING_TO_LREAL(GetParameter.Value);
  END_IF;
END_FOR;
```

Optional Customization Steps

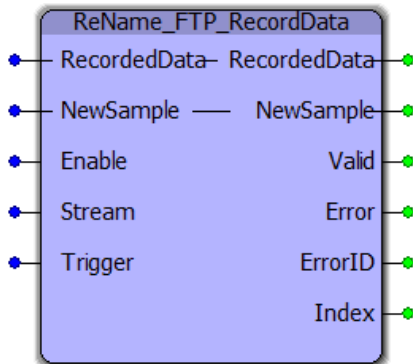
The CommandProcessor can process one or many commands per scan. This is a performance tuning issue. If the host device must send several setting at once, the MPiec controller may seem slow to process all the commands based on the Task interval. If the Task Interval and priority are set such that the CommandProcessor will have time to continue scanning the CircularByteBuffer in one scan until ALL bytes have been processed, performance will be improved by changing the following CommandProcessor code:

1. Remove AND NOT(CommandCreated) from main WHILE loop as shown

```
40
41   WHILE (CircularByteBuffer.StorePointer <> CircularByteBuffer.UsePointer) (*AND NOT (CommandCreated)*) DO
42     CommandCreated:=FALSE;
```




Rename_FTP_RecordData



This function works in combination with [ReName_FTP_SendData](#). Both functions must be copied and pasted into the main project and slightly customized for use. This function is designed to continuously record data using two databanks. Together, ReName_FTP_RecordData and ReName_FTP_SendData handshake and synchronize operations such that Recording takes place in one databank while the FTP transfer takes place using the other databank. Files of nearly unlimited size can be created on an FTP Server. Customization consists of creating an application specific structure similar to "ReName_RecordedUserData" in the main project which contains the specific data required by the application. This technique writes binary data to the file and therefore requires a program to read and interpret the same data structure. The data is non ASCII, unless converted to ASCII as part of the MotionWorks IEC customization using LREAL_TO_STRING and similar functions. Converting the data to ASCII on the MPlc is not recommended, especially if high speed recording is desired. ReName_FTP_RecordData typically executes in a faster task, even as fast as the Mecahtrolink update rate if necessary, recording up to 1000 samples of data in each "DataBank." There are two Databanks; one which is being written to by FTP_RecordData, and the other which contains data being uploaded to the FTP server. Rename_FTPSendData can typically execute in a slower task or the DEFAULT task.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	RecordedData	ReName_RecordedUserData	User customizable structure containing the data to be recorded. See examples below.
V	NewSample	ReName_RecordedSample	A single element of the many samples in one of the databanks. All the data in one sample is collected in one time slice.
VAR_INPUT			Default

B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Stream	BOOL	If Stream is True, a Sample will be added to the UserData structure every task interval.	FALSE
V	Trigger	BOOL	If Stream is False, Samples can be added to the UserData structure only upon the rising edge of Trigger.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	ErrorString	YC_STRING256	If 'Error' is true and it is an FTP response code related error then this output contains the response string from the FTP server.	

Notes

- This block utilizes FTP, not SFTP because SSL is not supported by the MPiec firmware. As a result, all FTP traffic sent and received (e.g. username, password, file data) is sent unencrypted in plain text and is visible to anyone with access to the network. This should not be a problem if the data being sent is not of a sensitive matter and the FTP server account is CHROOT'd properly (talk to your IT professional about using FTP).
- The FTP server should either have an internal/external domain name or use a static IP address because if the address changes, it will prevent the function from transferring files. See "Setup" for more details.
- The FTP user account must have "Write" privileges to successfully write files to the server. Optionally, the account may also have "Append" privileges. If files already exists and the FTP account only has "Write" privileges, then the file will be overwritten. If the file exists and the user account has "Append" privileges, then the file contents transferred will be appended to the existing file.

Error Description

See the [Function Block ErrorID](#) list.

Example

(It is assumed that an FTP server is configured and access is possible.)

1. Open your project.
2. Open the Comm Toolbox in a second copy of MotionWorks IEC.
3. In the second copy (Comm Toolbox) Project Tree Window, select ReName_FTP_SendFile and ReName_FTP_RecordData and **Copy**.
4. In the main project, Project Tree Window, right click on Logical POU's and select **Paste**.
5. For both of the new function blocks in your project, right click on them to access the properties page and rename the function blocks.

DataTypes

Create a datatype which describes the information to be recorded. The four datatypes shown below make up "MyRecordedUserData" which must be connected to the function blocks. Customize the detailed information to be recorded as shown on lines 5 through 11. Add or subtract as many variables are required. All code shown here was copied and pasted from the Comm Toolbox, then all instances of "ReName_" were changed to "My." Change to anything meaningful and appropriate. The only datatype which does not require modification and can remain in the Comm Toolbox is RecordStatusData.

```

1  TYPE
2
3      MyDataStruct:                (* Customized Sample of data to be recorded at frequent intervals *)
4      STRUCT
5          TimeStamp:RTC_Struct;
6          X:LREAL;
7          Y:LREAL;
8          Z:LREAL;
9          XTorque:LREAL;
10         YTorque:LREAL;
11         ZTorque:LREAL;
12     END_STRUCT;
13     MyDataArray: ARRAY[0..999] OF MyDataStruct;    (* Number of Samples to be recorded / written before toggling databanks *)
14     MyDataSet:ARRAY[0..1] OF MyDataArray;          (* Two databanks, no need to modify *)
15
16
17     MyRecordedUserData:            (* This is the datatype to be referenced by the FTP function blocks *)
18     STRUCT
19         Data: MyDataSet;
20         Status: RecordStatusData;    (* Using RecordStatusData from Comm Toolbox, everything else customized. *)
21     END_STRUCT;
22
23
24 END_TYPE

```

Initialization

```

1  Config.LocalIP:='192.168.207.151';    (* MPiec Controller *)
2  Config.DNSIP:='192.168.5.10';        (* DNS Server only necessary if using the FTP domain name rather than a hard coded IP address *)
3  Config.FTPIP:='192.168.201.29';      (* FTP Server *)
4  Config.Username:='User';
5  Config.Password:='test';

```

Editing the variables worksheets

Variables worksheet for ReName_FTP_RecordData:

Rename the datatypes.

Name	Type	Usage	Description
RecordedData	MyRecordedUserData	VAR_IN_OUT	
NewSample	MyDataStruct	VAR_IN_OUT	
Enable	BOOL	VAR_INPUT	
Stream	BOOL	VAR_INPUT	
Trigger	BOOL	VAR_INPUT	
Valid	BOOL	VAR_OUTPUT	
Error	BOOL	VAR_OUTPUT	
ErrorID	UINT	VAR_OUTPUT	
R_TRIG_Trigger	R_TRIG	VAR	
R_TRIG_Enable	R_TRIG	VAR	
Index	INT	VAR_OUTPUT	
RecordSize	INT	VAR	
RecordBank	INT	VAR	
OverrunError	BOOL	VAR	

Variables worksheet for ReName_FTP_SendData:

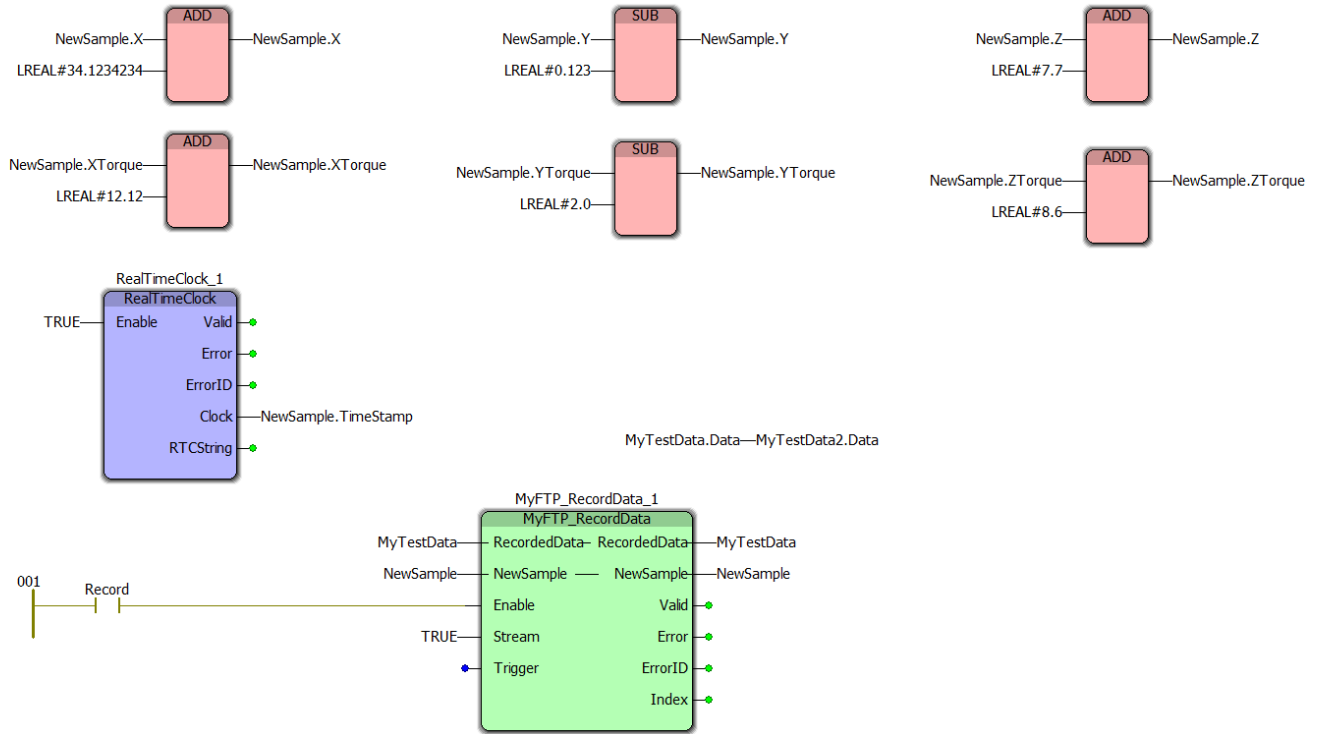
Rename the datatype.

Name	Type	Usage	Description	Address
RecordedData	MyRecordedUserData	VAR_IN_OUT		
Enable	BOOL	VAR_INPUT		
FTPData	FTP_Data	VAR_INPUT		
Destination	YTB_STRING64	VAR_INPUT		
Append	BOOL	VAR_INPUT		
Valid	BOOL	VAR_OUTPUT		
Busy	BOOL	VAR_OUTPUT		
Error	BOOL	VAR_OUTPUT		
ErrorID	UINT	VAR_OUTPUT		
ErrorString	YTB_STRING256	VAR_OUTPUT		

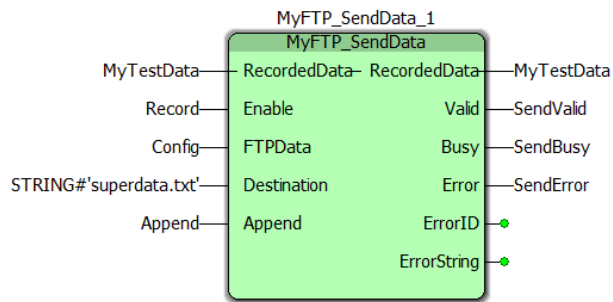
Function Blocks added to the Project:

MyFTP_RecordData_1 is in a "fast" task:

(*Just Test Data*)

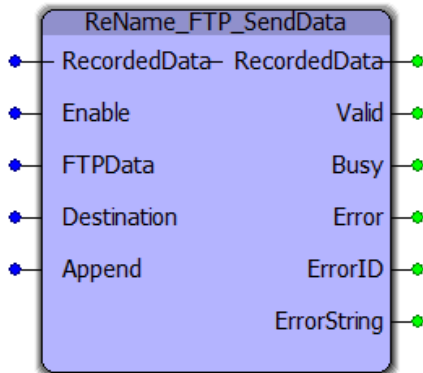


MyFTP_SendData is in the DEFAULT task:





ReName_FTP_SendData



This function works in combination with [ReName_FTP_RecordData](#). Both functions must be copied and pasted into the main project and slightly customized for use. This function is designed to continuously record data using two databanks. Together, ReName_FTP_RecordData and ReName_FTP_SendData handshake and synchronize operations such that Recording takes place in one databank while the FTP transfer takes place using the other databank. Files of nearly unlimited size can be created on the FTP Server. Customization consists of creating an application specific structure similar to "ReName_RecordedUserData" in the main project which contains the specific data required by the application. This technique writes binary data to the file and therefore requires a program to read and interpret the same data structure. The data is non ASCII, unless converted to ASCII as part of the MotionWorks IEC customization using LREAL_TO_STRING and similar functions. Converting the data to ASCII on the MPiec is not recommended, especially if high speed recording is desired. ReName_FTP_RecordData typically executes in a faster task, even as fast as the Mecahtrolink update rate if necessary, recording up to 1000 samples of data in each "DataBank." There are two Databanks; one which is being written to by FTP_RecordData, and the other which contains data being uploaded to the FTP server. Rename_FTPEndData can typically execute in a slower task or the DEFAULT task.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	RecordedData	ReName_ RecordedData	Rename and customzie this structure based on the needs of the application.
VAR_INPUT			Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
			FALSE

V	File	YC_STRING128	The full file name and location on the controller, e.g. '/flash/user-/data/example.csv'.	STRING#"
V	FTPData	FTP_Data	The input structure that configures the FTP transfer such as FTP server address, port, etc.	
V	Destination	YC_STRING64	The full file name and destination on the FTP server, e.g. 'metrics/example.csv'.	STRING#"
V	Append	BOOL	To select whether an existing the file on the server should be deleted and the new data should be appended to the existing data.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error).	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	ErrorString	YC_STRING256	If 'Error' is true and it is an FTP response code related error then this output contains the response string from the FTP server.	

Notes

- This block utilizes FTP, not SFTP because SSL is not supported by the MPiec firmware. As a result, all FTP traffic sent and received (e.g. username, password, file data) is sent unencrypted in plain text and is visible to anyone with access to the network. This should not be a problem if the data being sent is not of a sensitive matter and the FTP server account is CHROOT'd properly (talk to your IT professional about using FTP).
- The FTP server should either have an internal/external domain name or use a static IP address because if the address changes, it will prevent the function from transferring files. See "Setup" for more details.
- The FTP user account must have "Write" privileges to successfully write files to the server. Optionally, the account may also have "Append" privileges. If files already exists and the FTP account only has "Write" privileges, then the file will be overwritten. If the file exists and the user account has "Append" privileges, then the file contents transferred will be appended to the existing file.
- There is a handshake system in place between ReName_FTP_RecordData and ReName_FTP_SendData so that if there is a timing problem and ReName_FTP_RecordData must switch to the other DataBank but ReName_FTP_SendData is still busy, and Error will occur. Timing issues could be based on the tasks in which both function blocks are executing, and if further customization has been performed, such as by modifying the number of samples in each DataBank.

Error Description

See the [Function Block ErrorID](#) list.

Example Customization

(It is assumed that an FTP server is configured and access is possible.)

1. Open your project.
2. Open the Comm Toolbox in a second copy of MotionWorks IEC.
3. In the second copy (Comm Toolbox) Project Tree Window, select ReName_FTP_SendFile and ReName_FTP_RecordData and **Copy**.
4. In the main project, Project Tree Window, right click on Logical POUs and select **Paste**.
5. For both of the new function blocks in your project, right click on them to access the properties page and rename the function blocks.

DataTypes

Create a datatype which describes the information to be recorded. The four datatypes shown below make up "MyRecordedUserData" which must be connected to the function blocks. Customize the detailed information to be recorded as shown on lines 5 through 11. Add or subtract as many variables are required. All code shown here was copied and pasted from the Comm Toolbox, then all instances of "ReName_" were changed to "My." Change to anything meaningful and appropriate. The only datatype which does not require modification and can remain in the Comm Toolbox is RecordStatusData.

```

1  TYPE
2
3      MyDataStruct:                (* Customized Sample of data to be recorded at frequent intervals *)
4      STRUCT
5          TimeStamp:RTC_Struct;
6          X:LREAL;
7          Y:LREAL;
8          Z:LREAL;
9          XTorque:LREAL;
10         YTorque:LREAL;
11         ZTorque:LREAL;
12     END_STRUCT;
13     MyDataArray: ARRAY[0..999] OF MyDataStruct;    (* Number of Samples to be recorded / written before toggling databanks *)
14     MyDataSet:ARRAY[0..1] OF MyDataArray;          (* Two databanks, no need to modify *)
15
16
17     MyRecordedUserData:            (* This is the datatype to be referenced by the FTP function blocks *)
18     STRUCT
19         Data: MyDataSet;
20         Status: RecordStatusData;    (* Using RecordStatusData from Comm Toolbox, everything else customized. *)
21     END_STRUCT;
22
23
24 END_TYPE

```

Initialization

```


1  Config.LocalIP:='192.168.207.151';    (* MPiec Controller *)
2  Config.DNSIP:='192.168.5.10';        (* DNS Server only necessary if using the FTP domain name rather than a hard coded IP address *)
3  Config.FTPIP:='192.168.201.29';      (* FTP Server *)
4  Config.Username:='User';
5  Config.Password:='test';

```

Editing the variables worksheets

Variables worksheet for ReName_FTP_RecordData:

Rename the datatypes.



Name	Type	Usage	Description
☐ Default			
RecordedData	MyRecordedUserData	VAR_IN_OUT	
NewSample	MyDataStruct	VAR_IN_OUT	
Enable	BOOL	VAR_INPUT	
Stream	BOOL	VAR_INPUT	
Trigger	BOOL	VAR_INPUT	
Valid	BOOL	VAR_OUTPUT	
Error	BOOL	VAR_OUTPUT	
ErrorID	UINT	VAR_OUTPUT	
R_TRIG_Trigger	R_TRIG	VAR	
R_TRIG_Enable	R_TRIG	VAR	
Index	INT	VAR_OUTPUT	
RecordSize	INT	VAR	
RecordBank	INT	VAR	
OverrunError	BOOL	VAR	

Variables worksheet for ReName_FTP_SendData:

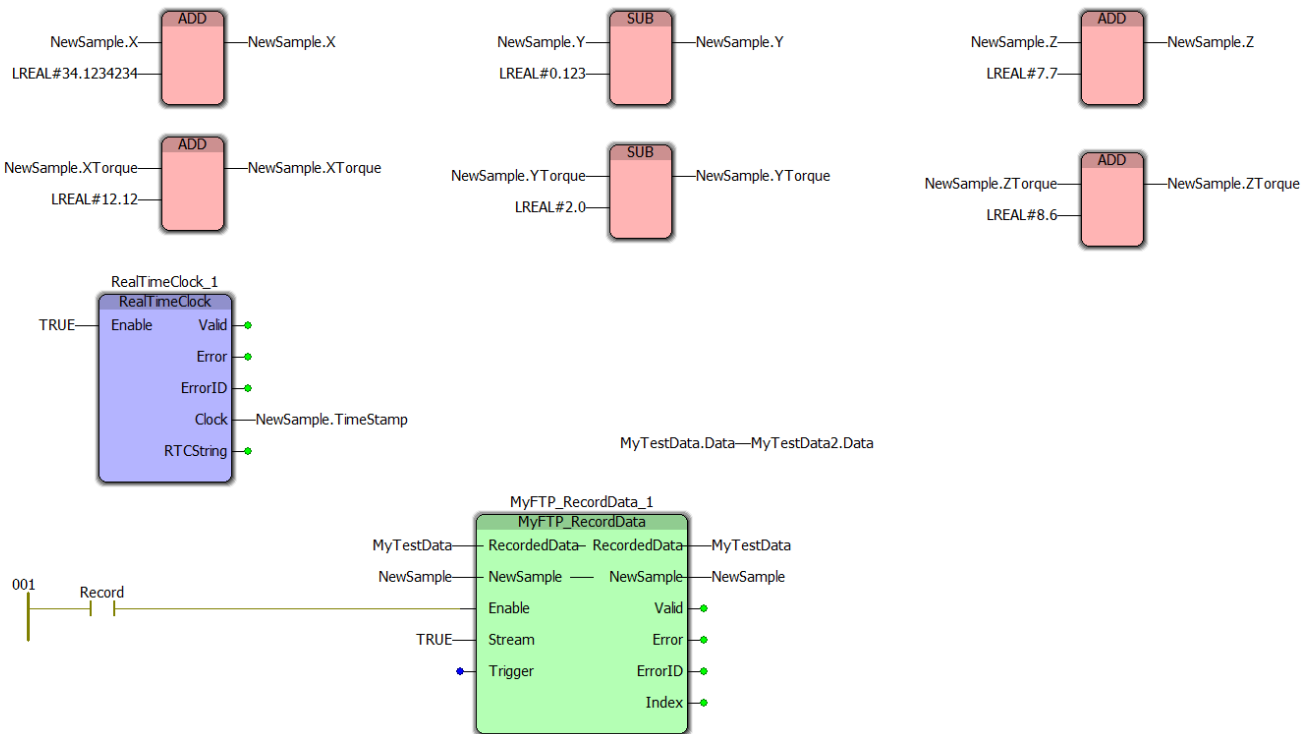
Rename the datatype.

Name	Type	Usage	Description	Address
Private				
RecordedData	MyRecordedUserData	VAR_IN_OUT		
Enable	BOOL	VAR_INPUT		
FTPData	FTP_Data	VAR_INPUT		
Destination	YTB_STRING64	VAR_INPUT		
Append	BOOL	VAR_INPUT		
Valid	BOOL	VAR_OUTPUT		
Busy	BOOL	VAR_OUTPUT		
Error	BOOL	VAR_OUTPUT		
ErrorID	UINT	VAR_OUTPUT		
ErrorString	YTB_STRING256	VAR_OUTPUT		

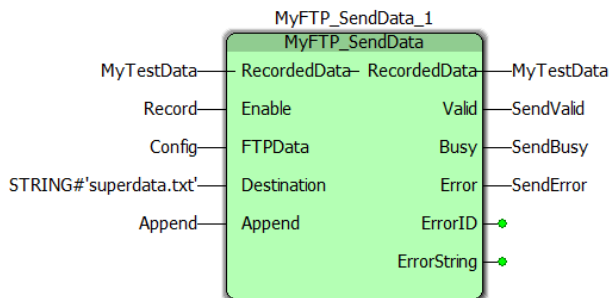
Function Blocks added to the Project:

MyFTP_RecordData_1 is in a "fast" task:

(*Just Test Data*)

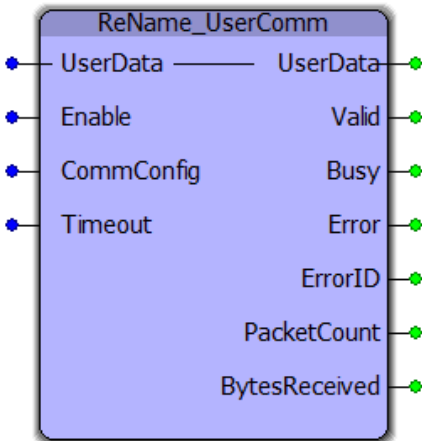


MyFTP_SendData is in the DEFAULT task:





Rename_UserComm



The ReName_UserComm function block is designed to facilitate simple data exchange with another device by opening a socket and sending /receiving raw binary data. The function must be copied and pasted into the main project and slightly customized for use. The overall design includes a C# DLL which must also be customized.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	UserData	ReName_UserStruct	During customization, define and create any structure as required for the application.	
VAR_INPUT				
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	CommConfig	BOOL	Resets the StorePointer and UsePointer, which logically clears the contents of the circular buffer, but any existing data is not actually cleared.	FALSE
V	TimeOut	TIME	TIME	FALSE
VAR_OUTPUT				

B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	PacketCount	UDINT	Total packet count since this function block was enabled.
V	BytesReceived	UDINT	Total bytes received since this function was enabled.

Error Description

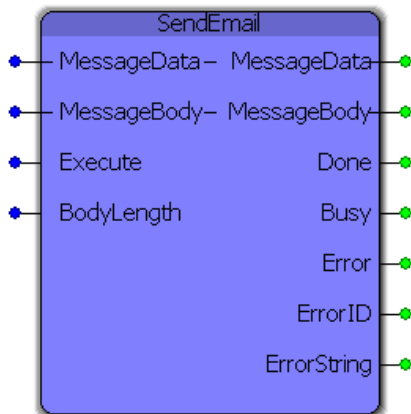
See the [Function Block ErrorID](#) list.

Example

1. Open your project.
2. Open the Comm Toolbox in a second copy of MotionWorks IEC.
3. In the second copy (Comm Toolbox) Project Tree Window, select ReName_FTP_SendFile and ReName_FTP_RecordData and **Copy**.
4. In the main project, Project Tree Window, right click on Logical POU's and select **Paste**.
5. For both of the new function blocks in your project, right click on them to access the properties page and rename the function blocks.



SendEmail



This function block sends an e-mail via SMTP commands (Simple Mail Transfer Protocol) through a specified SMTP server. The output is highly configurable including multiple recipients, any message body structure, specified sender e-mail and name and other features listed below.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	MessageData	SMTP_Data	A user customized data structure for configuring the e-mail block.	
V	MessageBody	YTB_ BYTE4096	The e-mail body as a 4096 element byte array. If a larger body is required, this declaration can be changed and the library recompiled.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	BodyLength	UDINT	The length (number of bytes) of the e-mail body that will be sent. While not necessary it is highly suggested, see notes below.	UDINT #0
VAR_OUTPUT				
B	Done	BOOL	Set high upon successfully sending an e-mail.	
B	Busy	BOOL	Set high upon the start of communications with the SMTP server and low when 'Done' or 'Error' go high.	

B	Error	BOOL	Set high when an error occurs during e-mail configuration and sending. Set low upon Execute being reset.
E	ErrorID	UINT	If Error is true, this output provides the ErrorID. Cleared upon 'Execute' being reset.
V	ErrorString	YTB_STRING256	If 'Error' is true and it is an SMTP response code related error then this output contains the response string from the SMTP server.

Notes

- The MPiec series controllers do not support SSL SMTP servers and therefore will most likely only work with local network SMTP servers. Talk with your IT professional about connecting to a local SMTP server from an MPiec Series Controller (see "Setup" below for more details about the required configuration).
- The "BodyLength" input is optional but highly suggested to reduce the packet size and the potential for large amounts of padding ("0") bytes on the recipients side. All examples include this Input and demonstrate how to get the correct length.

Error Description

See the [Function Block ErrorID](#) list.

Example

As this is a complicated function, additional examples are provided in separate help files listed under "Additional Examples" and prefixed with "SMTP_". The example shown here sets up the block, creates a message body and sends an e-mail to external Gmail account.

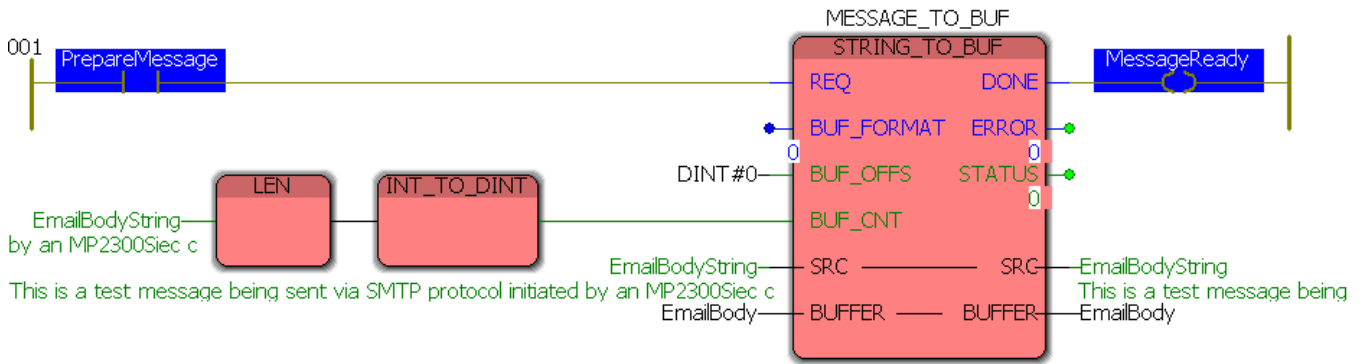
The variable EmailBodyString is of type YC_STRING256. Below is the configuration of the SMTP_Data structure:

```
(* E-mail Setup *)
EmailBodyString := 'This is a test message being sent via SMTP protocol initiated by an MP2300Siec controller.';

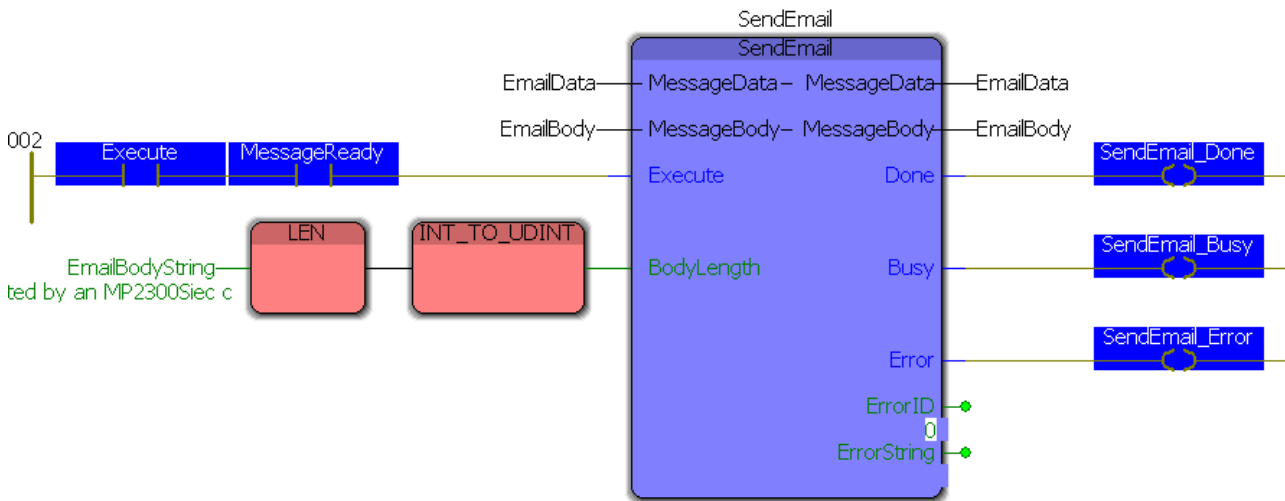
EmailData.DNSIP := '192.168.5.10';
EmailData.Domain := 'YASKAWA';
EmailData.LocalIP := '192.168.207.205';
EmailData.NumRcpt := INT#1;
EmailData.RcptArray[0].name := 'Logan Smith';
EmailData.RcptArray[0].email := '██████████';
EmailData.Sender := 'logan_smith@yaskawa.com';
EmailData.SenderName := 'MP2300Siec';
EmailData.SMTPDomain := 'athena.yaskawa.com';
EmailData.Subject := 'Test message from your MP2300Siec';
```

The most basic form of sending an e-mail is simply converting a string to a byte array via the STRING_TO_BUF function block provided in the PROCONOS firmware library. With the data structure shown above and this STRING_TO_BUF block, the email is configured and ready for use.

(* Pass the message into a buffer *)

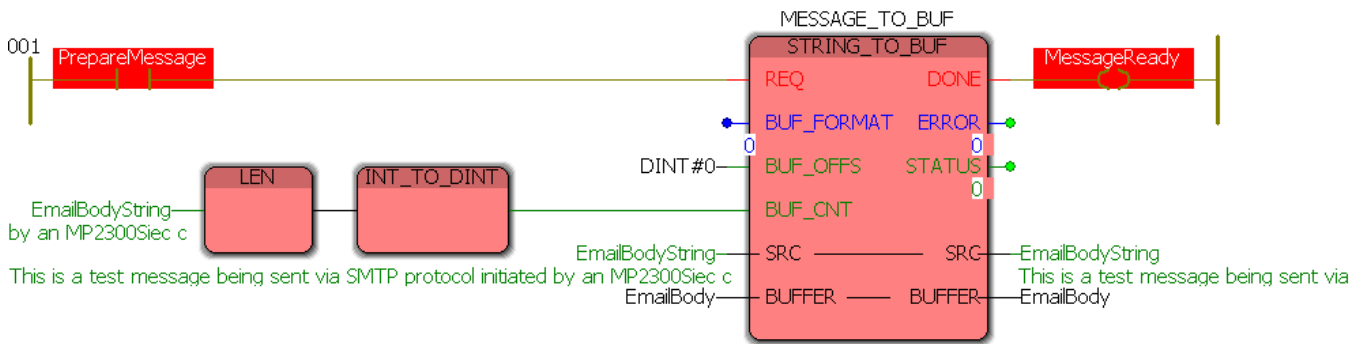


(* Send the message *)

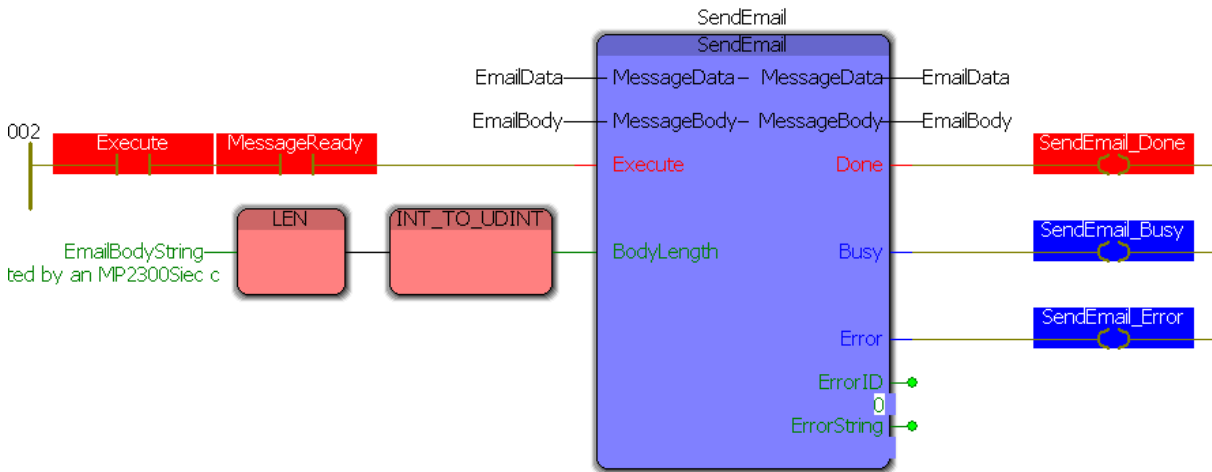


After toggling PrepareMessage, here is the result.

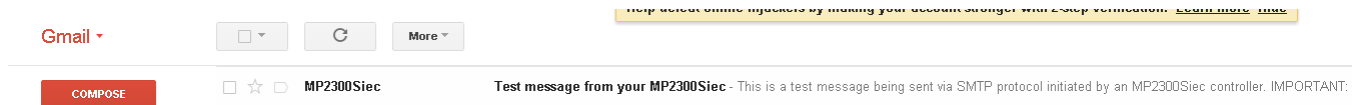
(* Pass the message into a buffer *)



(* Send the message *)



And to demonstrate the end result, here is the e-mail in the inbox of the Gmail account used. The sender and subject are both listed correctly and a portion of the send message can be seen.





Getting Started with File Read / Write Toolbox

Requirements

To use the File Read / Write Toolbox, the project must also contain the following:

Firmware libraries:

- PROCONOS

User libraries:

The following User Libraries must be listed above the File Read Write Toolbox:

- Yaskawa_Toolbox

The File Read / Write Toolbox contains some functions must be customized for use in every application.

The two customizable function blocks in this library are:

- [Write_CSV_File](#)
- [Read_CSV_File](#)

To use these functions, they must be copied and pasted into your main project as a function block with a different (but similar) name. To do this, copy and paste the structured text and the variable definitions grid from the toolbox version. These four main functions refer to other sub functions in the File Read Write toolbox, which do not require customization and can remain in the File Read Write Toolbox. There is no need to move the following function blocks:

- [Read_Buffer](#)
- [Read_Line](#)
- [Read_Value](#)

More detailed customization information and examples are provided for the help for each of the functions blocks mentioned above.

See Yaskawa's YouTube Webinar - [CSV File Transfer with the File_RW Template](#).



File RW Toolbox Revision History

Current Version:

2023-09-27 v375 released

- 1) HC_WriteParameter FB - Cant now write prm 1833 (MachineCycle) if not already set as rotary (get ErrorID 10165). DCR 7004.
- 2) HC_WriteParameter FB - Fix bug that may change parameter data for an axis beyond the specified AXIS_REF. Only experienced when trying to write prm 1833. DCR 7178.
- 3) HC_WriteParameter FB - Add support for changing from linear to rotary and vice versa. DCR 7179.
- 4) HC_WriteParameter FB - Fix ability to set prm 1813. DCR 7180.
- 5) ReadValue FB - (G Code Emulation mode 2 issue) F60Z is not accepted, no error. Did not see the single instruction 'Z' as the last on the line. DCR 7118.

Previous Versions:

2022-06-11 v374 released

- 1) HC_WriteParameter - Added Support for External encoder prm 1016 ~ 1018 to invert external encoder signals. DCR 3324.
- 2) HC_WriteParameter - HC online values didn't not show the updated value that was last written by HC_WriteParameter. DCR 6268.
- 3) DataLogCompare - Improve to allow slightly less recorded samples than the reference without Error. DCR 6696

2021-07-07 v372 released

- 1) ReadValue FB - G-Code Override commands with trailing spaces caused GCode_Processor to hangup. DCR 5199.
- 2) ReadValue FB - No longer outputs the comma delimiter with desired value (in Non GCode mode.) DCR 5416.
- 3) HC_WriteParameter FB - Error logic was not correct, could cause the function to hang busy in some cases. DCR 5520.
- 4) Read_XML_File FB - Was missing Done output in RETURN statement. DCR 5519.
- 5) HC_WriteParameter / Set_XML_Data FB - Was not detecting servo or external axistype to determine need to write legacy parameters. DCR 5518.
- 6) DataCompare function blocks DataLogRecord, DataLogWrite and DataLogCompare reintroduced.

2020-11-30 v371 released.

- 1) ReadValue FB - added support to ignore comments added to G-Code % preamble. DCR 4697.
- 2) HC_WriteParameter - Fixed bug preventing the Hardware Configuration from detecting that parameters had been changed via this function block. DCR 4808.

2020-02-06 v370 released.

- 1) Homing via Group Toolbox GroupToHome FB did not execute more than once when using the v352 Toolbox Installer version. DCR 2985.
- 2) G-Code: [support comments with square brackets]. DCR 3173.

3) G-Code: support lower case in G-Code data. DCR 3329.

2018-12-28 v352 released.

- 1) ReadValue - Failure when CRLF falls exactly on a ReadBuffer boundary. (CR and end of buffer, LF at next beginning. DCR 2110.
- 2) ReadValue, Read_CSV_File - Increase size of STRING DataTypes from 80 to 128 chars. DCR 2111.
- 3) G-Code - lines containing both variables and comments allowed Comments to be passed through to G-Code parser. DCR 2228.
- 4) ReadValue - supports G-Code block skip mode. DCR 2191.
- 5) ReadValue - G-Code lines containing IF statements containing no spaces allowed conditional statements be copied into command buffer with extra character. DCR 2254.
- 6) ReadValue G-Code lines (with comments like this) were leaking a CR character into the command buffer. DCR 2349.

2018-06-26 v350 released.

- 1) Read_CSV_File template FB - Fixed initialization Error. DCR 1439.
- 2) Data logging FBs - Improvement to programatically check array sizes (non hard coded) DCR 1722.
- 3) CreateFilePath - New function block to concatenate path filename and extension. DCR 1773.

2017-12-06 v340 released. Created using 3.4.0 firmware

- 1) GetTagName FB - New in v340. For example, <HeightZ>164</HeightZ> will return HeightZ. DCR 1066.
- 2) ReadBuffer FB - Fixed DataBuffer Initialization bug. DCR 925.
- 3) Read_CSV_File template - Fixed DataBuffer Initialization bug. DCR 1109.
- 4) HC_WriteParameter - Writing Prm 1833 did not update Axis.XML to change MachineCycle. DCR 1194.
- 5) ByteBufferStruct - Changed datatype for FilePosition from DINT TO UDINT. DCR 1225.
- 6) ReadValue - Now detects special statements in G-Code data for expressions and formulas. DCR 1241.
- 7) ReadValue - Was filtering out space characters, now includes in the output. DCR 1294.

2016-09-15 v301 released. Created using 3.2.0 firmware, but not necessary

- 1) ReadLine & ReadValue functions - Removed VAR_INPUT StartChar and VAR_OUTPUT EndChar, the function blocks now manage this data in the databuffer directly.
- 2) HC_WriteParameter - new function block added. Writes configuration parameters which are not available with MC_WriteParameter.
- 3) ReadBuffer - Improved ReadBuffer to report when no data was read from file.
- 4) Read_CSV_File - The performance of this template POU was improved by using a two scan WHILE loop technique.
- 5) ReadValue - Improved automatic handling for files with CRLF, CR, or just LF.

2015-01-31 v300 released

- 1) Identical to v205, but recompiled specifically for MotionWorks IEC v3.x.

2015-01-15 v205 released. Created using 2.5.0 firmware

Merged new data logging from FileRW_Toolbox_v204HSv2.ZWT

- 1) Added Function Blocks: DataLogGenerate, DataLogGenerate_2Points, DataLogCompare, & DataLogCompareLive. *)
- 2) DataLogCompareLive - Known issue: Setting Periodic to TRUE on the function block will only work when if log file has more than 1000 entries in the log file otherwise the function block will error.

3) ReadValue - Added VAR_INPUT GCodeMode to support detection of numeric to alpha transitions as delimiters in G-Code.

2014-06-13 v204 released. Created using 2.5.0 firmware

- 1) ReadValue - Fixed bug introduced in v203 to auto detect CRLF, CR, or LF as line delimiters. A file with CRLF was being populated into every other array index because the first character in the read after a new line was BYTE#10.
- 2) FileExists - Added new function block.
- 3) ReadValue - Added VAR_INPUT GCodeMode to support detection of numeric to alpha transitions as delimiters in G-Code.
- 4) Known Issue - When reading CSV files, the last line must contain the same line delimiter, CR or CRLF as all other lines in the file, or the last line will not be read into the structure.

2014-04-11 v203 released. Created using 2.5.0 firmware

- 1) Added support for files that only use line feed (BYTE#10) as a line delimiter instead of carriage return (BYTE#13).
Two locations in ReadBuffer FB, One location in ReadValue FB. *)

2013-09-02 v202 released. Created using 2.4.0 firmware

- 1) ReadValue - Added "OR (x = DataBuffer.Length)" to cause EOF flag even if the <CR> is not the last byte in a line
- 2) Read_CSV_File & Write_CSV_File - Added PreStringError 10017 to detect if the controller already has a String Conversion alarm
posted before the function blocks execute.

2012-08-23 v201 released. Created using 2.2.0 firmware

- 1) Improved reading of very last portion of buffer. (partial buffer read when EOF occurred)
- 2) Tested Write_CSV_File Append mode. (Now supported.)
- 3) Added Write_CSV_File ErrorID 10119 to stop execution if the Row or Column data in the User Structure are zero.
- 4) Improved handling of ErrorID=22 by forcing a FILE_CLOSE if there was an error opening a file.

2012-02-08 v200 released. created using 2.1.0 firmware

- 1) First official release version with documentation provided for Read_CSV_File. (See Toolbox manual.)
- 2) Read_Binary_File and Write_Binary_File still under development for speed improvements.

2011-12-19 v004beta created using 2.1.0 firmware

- 1) Added ReadBuffer function block for use by Read_CSV_File to increase the efficiency of the file read operation.
Previously, logic was searching for one line of bytes at a time, and it took 6 scans to process each line. With the ReadBuffer FB, Read_CSV_File can process a given buffersize in 6 scans. Default is 2048 bytes, settable by user.



Data Type: ByteBufferStruct

For use with the ReadBuffer, ReadLine, and ReadValue function blocks. These block are designed to share the same data. DataBuffer is the largest piece of data, read directly from a file using the FILE_READ function block from the ProConOS firmware library, ReadLine returns a line of ASCII data from the DataBuffer, and ReadValue returns a single value found between two delimiters.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyDataBuffer	ByteBufferStruct		
C	BufferSize	UINT	Not Used.	MyDataBuffer.BufferSize
C	FilePosition	DINT	The location in the file which was last read into the DataBuffer. The location in the file which is the beginning of the next read or write.	MyDataBuffer.FilePosition
C	Char	YTB_ByteArray16384	Array of Bytes read from the file. This datatype is declared in the Yaskawa Toolbox.	MyDataBuffer.Char[4]
U	CharPointer	UINT	Points to the characters in DataBuffer.Char[] which one of the functions is currently evaluating.	MyDataBuffer.CharPointer
U	CharsRead	UINT	Reports the actual number of characters read from the file, which may be less than the full size of DataBuffer.Char[]. If this is the case, the file was either smaller than the size of DataBuffer.Char[] or the end of the file has been reached.	MyDataBuffer.CharsRead
U	CharBuffer	UINT	CharBuffer may be less than CharsRead because certain chars are filtered out, like TAB, BYTE#9	MyDataBuffer.CharBuffer



Data Type: ChannelArray

Supporting array for [ChannelStruct](#). For use with the [DataLogCompare](#) function block.

Data Type Declaration

TYPE

ChannelArray: ARRAY[1..6] OF [ChannelStruct](#);

END_TYPE



Data Type: ChannelStruct

Used with the [DataLogCompare](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyChannelStruct	ChannelStruct		
C	RefValue	LREAL	Value of the reference data point.	MyChannelStruct.RefValue
C	CompValue	LREAL	Value of the comparison data point..	MyChannelStruct.CompValue



Data Type: CompareStruct

Used with the [DataLogCompare](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCompareResults	CompareStruct		
C	LoadingReferenceFile	BOOL	Denotes whether or not the Reference file is being loaded into a Reference structure.	MyCompareResults.LoadingReferenceFile
C	LoadingComparisonFile	BOOL	Denotes whether or not the Comparison file is being loaded into a Comparison structure.	MyCompareResults.LoadingComparisonFile
C	RowsCompared	DINT	The number of rows in each file that have been compared.	MyCompareResults.RowsCompared
C	FileCompared	BOOL	Denotes whether or not the two files have been compared.	MyCompareResults.FileCompared
C	NumOfDifferences	INT	Reports the number of differences found.	MyCompareResults.NumOfDifferences
C	Differences	DiffValueArray	An array of each difference found.	MyCompareResults.Differences



Data Type: DataLoggingStruct

Structure containing parameters for recording and writing general purpose data. This structure is used with the [DataLoggingRecord](#) and [DataLoggingWrite](#) function blocks.

Data Type Declaration

	*	Element	Data Type	Description	Usage
		MyRecordingPrms	DataLoggingStruct		
C		Version	DINT	The version of this structure [yyyymmdd].	MyRecordingPrms.Version
C		ChannelsUsed	WORD	Denotes the channels that are used to record data.	MyRecordingPrms.ChannelsUsed
U		MaxRecordingTime	DINT	The maximum amount of time in seconds to spend recording.	MyRecordingPrms.MaxRecordingTime
C		TimeLimitReached	BOOL	Denotes whether or not the time limit has been reached.	MyRecordingPrms.TimeLimitReached
C		SampleRate	REAL	The sample rate of the logged data [ms].	MyRecordingPrms.SampleRate
C		Status	StatusFlagsSet	Status of recording and writing data for each buffer.	MyRecordingPrms.Status
C		Size	INT	Denotes how many rows of data can be stored.	MyRecordingPrms.Size
C		Max	INT	The LAST array index where data can be stored.	MyRecordingPrms.Max
C		Data	LoggedValuesSet	The data that is recorded.	MyRecordingPrms.Data



Data Type: DiffStruct

Used with the [DataLogCompare](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyDiffStruct	DiffStruct		
C	RefRow	DINT	The row in the Reference file where this difference is located.	MyDiffStruct.RefRow
C	CompRow	DINT	The row in the Comparison file where this difference is located.	MyDiffStruct.CompRow
C	Channel	ChannelArray	An array of channels.	MyDiffStruct.Channel



Data Type: DiffValueArray

Supporting structure for [CompareStruct](#). For use with the [DataLogCompare](#) function block.

Data Type Declaration

TYPE

DiffValueArray: ARRAY[0..1000] OF [DiffStruct](#);

END_TYPE



Data Type: LoggedValuesArray

Supporting array for [LoggedValuesStruct](#). For use with the [DataLogRecord](#) and [DataLogWrite](#) function blocks.

Data Type Declaration

TYPE

LoggedValuesArray: ARRAY[0..4999] OF [LoggedValuesStruct](#);

END_TYPE



Data Type: LoggedValuesSet

Supporting structure for [DataLoggingStruct](#). For use with the [DataLogRecord](#) and [DataLogWrite](#) function blocks.

Data Type Declaration

TYPE

LoggedValuesSet: ARRAY[0..1] OF [LoggedValuesArray](#);

END_TYPE



Data Type: LoggedValuesStruct

Supporting structure for [LoggedValuesArray](#). Used with the [DataLogRecord](#) and [DataLogWrite](#) function blocks.

Data Type Declaration

	*	Element	Data Type	Description	Usage
		MyLoggedValues	LoggedValuesStruct		
C		SampleNum	DINT	The sample number of this data set.	MyLoggedValues.SampleNum
C		Channel1	LREAL	Value of the first input.	MyLoggedValues.Channel1
C		Channel2	LREAL	Value of the second input.	MyLoggedValues.Channel2
C		Channel3	LREAL	Value of the third input.	MyLoggedValues.Channel3
C		Channel4	LREAL	Value of the fourth input.	MyLoggedValues.Channel4
C		Channel5	LREAL	Value of the fifth input.	MyLoggedValues.Channel5
C		Channel6	LREAL	Value of the sixth input.	MyLoggedValues.Channel6



Data Type: StatusFlagsSet

Supporting structure for [DataLoggingStruct](#). For use with the [DataLogRecord](#) and [DataLogWrite](#) function blocks.

Data Type Declaration

TYPE

StatusFlagsSet: ARRAY[0..1] OF [StatusFlagsStruct](#);

END_TYPE



Data Type: StatusFlagsStruct

Used with the [DataLogRecord](#) and [DataLogWrite](#) function blocks.

Data Type Declaration

	*	Element	Data Type	Description	Usage
		MyStatusFlags	StatusFlagsStruct		
C		Recording	BOOL	Denotes whether or not the data is being recorded.	MyStatusFlags.Recording
C		Writing	BOOL	Denotes whether or not the data is being written to a file.	MyStatusFlags.Writing
C		StorePointer	INT	The array index where the last data point is stored.	MyStatusFlags.StorePointer
C		Overwritten	BOOL	Flag which indicates if the buffer has circulated over the beginning of the data; to prevent reuse.	MyStatusFlags.Overwritten



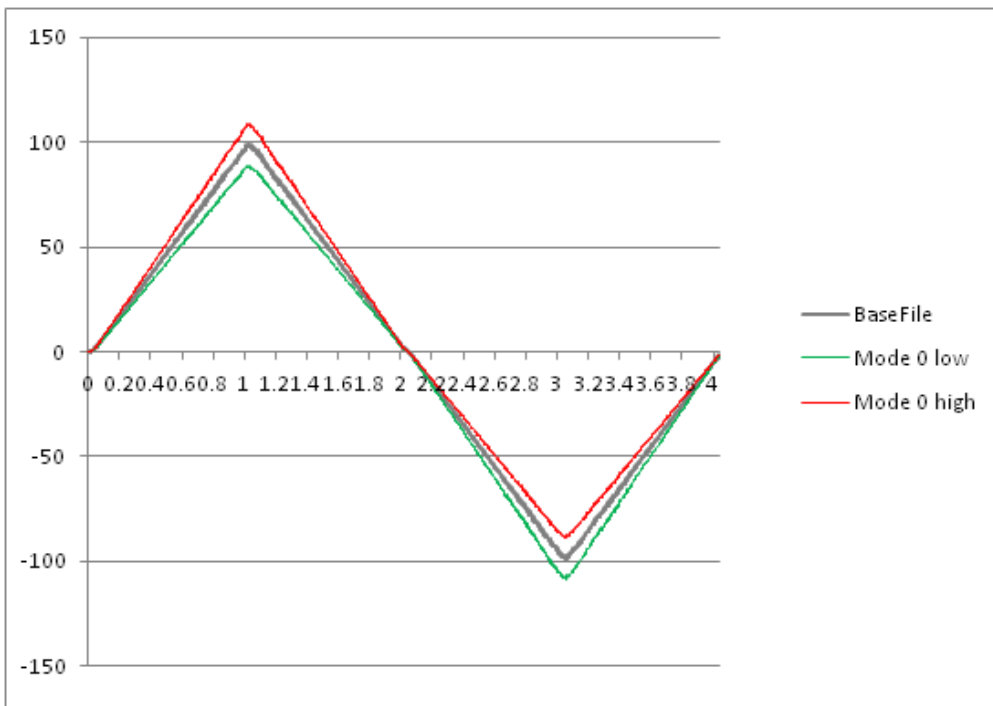
Enumerated Types in the File R/W Toolbox

Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to zero-based integers (INT). Therefore, the first value equates to 0, the second to 1, etc.

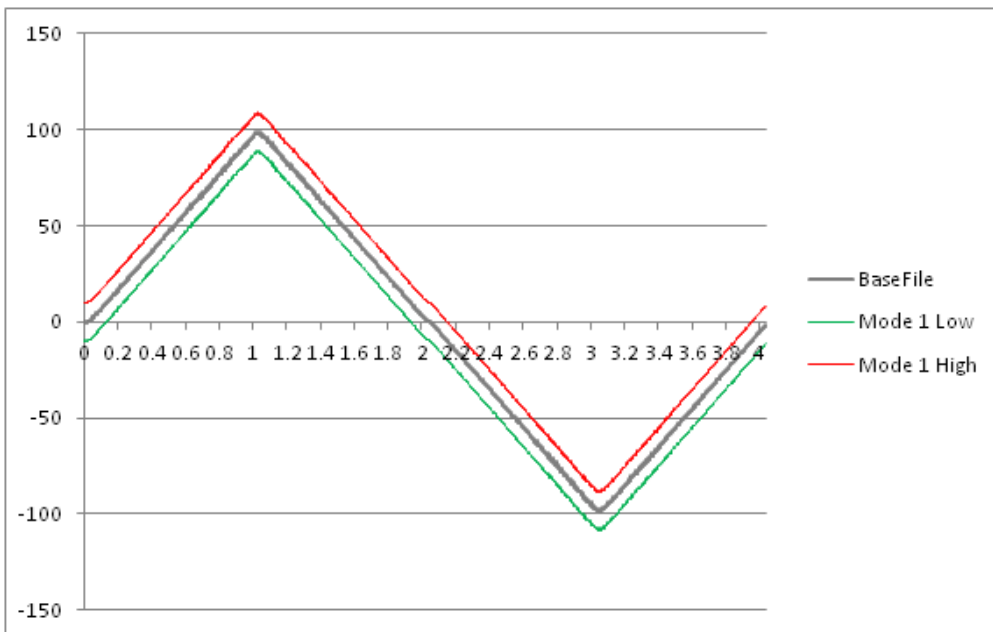
Enumerated Types Declaration

Enumerated Type	#INT Value	Enum Value	Description
ComparisonMode	Enumerated type to be used with CommStruct.CommType		
	0	Percentage	Uses a percentage of the current value in the BaseFile to set the window.
	1	PercentOfMax	Uses a percentage of the maximum value found in the BaseFile to set the window.
	2	PercentOfAverage	Uses a percentage of the average absolute value found in the BaseFile to set the window.
	3	RawWindow	Uses a raw value to specify a window around the BaseFile.
FTB_FileLocation	Enumerated type to specify where a file is stored on the controller. Used with the CompareFiles and WriteFile function blocks.		
	0	Ram	The file is in ramdisk.
	1	Flash	The file is in flash.

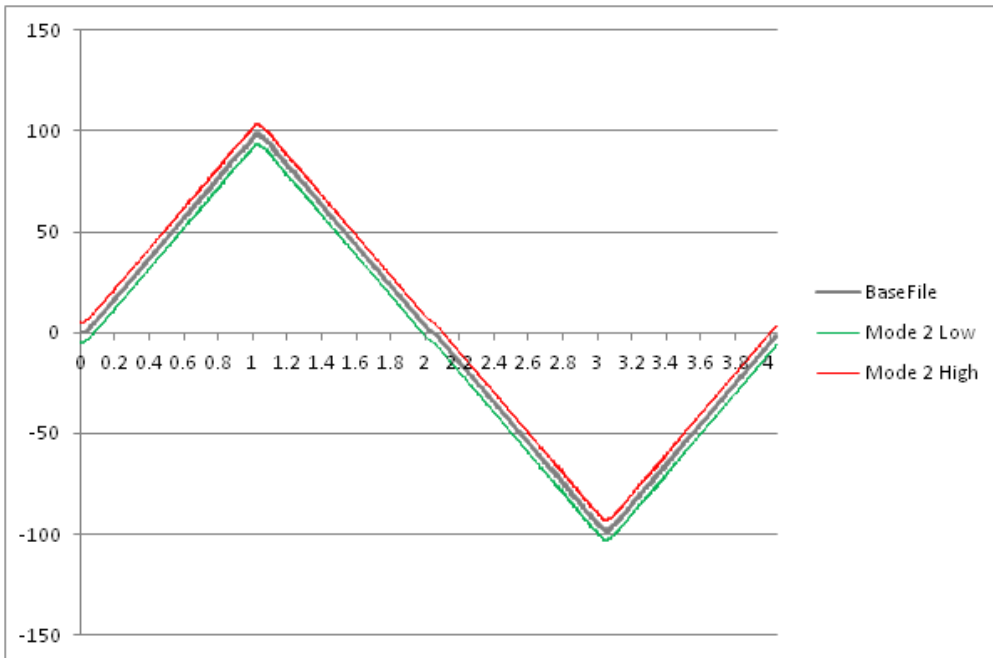
Percentage



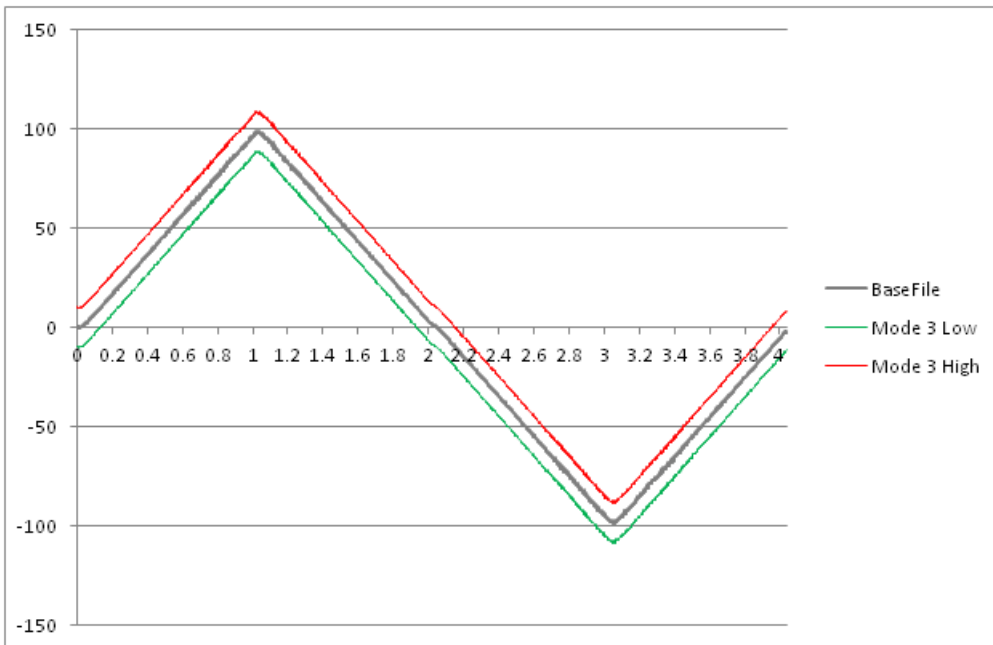
PercentOfMax



PercentOfAverage

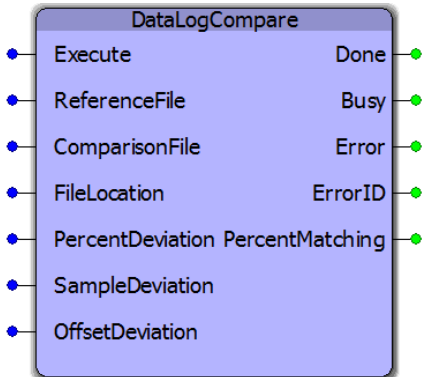


RawWindow





DataLogCompare



This function block compares two files and reports the PercentMatching within the user specified allowable PercentDeviation, SampleDeviation and OffsetDeviation.

Library

FileRW Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	ReferenceFile	STRING	The name of the file on the controller that is presumed to have good data. Do not include an extension, .txt will be automatically appended.	STRING#"
V	ComparisonFile	STRING	The name of the file on the controller that is presumed to have good data. Do not include an extension, .txt will be automatically appended.	STRING#"
V	FileLocation	FTB_FileLocation	Specify where on the controller the Comparison file is stored as either: FTB_FileLocation#Ram or FTB_FileLocation#Flash	FTB_FileLocation#Ram
V	PercentDeviation	YTB_LREALArray8	Percentage by which a given sample in the ComparisonFile can differ from the ReferenceFile and still be considered a match.	LREAL#0.0

V	SampleDeviation	FTP_INTArray8	The number of neighboring samples to search for a matching value if a match cannot be attained in the equivalent sample using the PercentDeviation and OffsetDeviation criteria. Specify 0 to 20 samples.	INT#0
V	OffsetDeviation	YTB_LREALArray8	Maximum offset by which a given sample in the ComparisonFile can differ from the ReferenceFile and still be considered a match.	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low. Set high when two files have been compared.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error) Busy will be high while comparing two files to each other.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	PercentMatching	LREAL	A summary reflecting the similarity of the two files.	

Notes

- Three function blocks work together in this solution: [DataLogRecord](#), [DataLogWrite](#), and DataLogCompare.
- Start and Stop the DataLogRecord function in conjunction with an event signifying when the data should be recorded. This is intended to be a single event, not a series of intermittent events. At some time after DataLogRecord is Valid, Execute the DataLogWrite function, typically in a slower task than DataLogRecord. Typical usage includes DataLogRecord in a Cyclic task as fast as the Mechatrolink rate if necessary, DataLogWrite in a 32 to 64 mSec task, and DataLogCompare in a 32 to 64 mSec task. DataLogWrite and DataLogCompare work most most efficiently and quickly in the Default task.
- These functions use a double buffer technique, swapping from one to other. DataLogRecord accesses one buffer while DataLogWrite accesses the other. The two activities must never interfere with each other, or an error will be generated if there is a timing conflict when trying to switch buffers. If this occurs, it is possible that DataLogWrite must be executed at a faster interval.
- The comparison algorithm first checks for a match within the offset specified. If the data does not match, then it will check within the allowed percentage. If still a match cannot be determined, the process repeats by iterating through the number of samples before and after the equivalent time sample. If a SampleDeviation of 20 is provided, the process will check up to 20 samples before and 20 after the sample in question.

Error Description

See the [Function Block ErrorID](#) list.

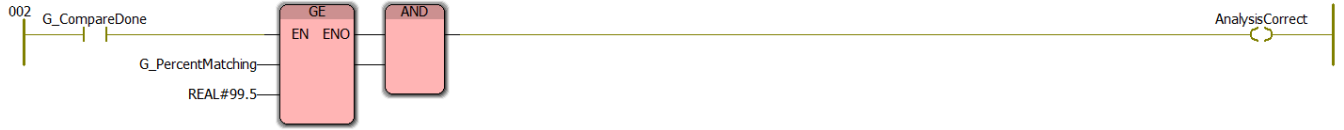
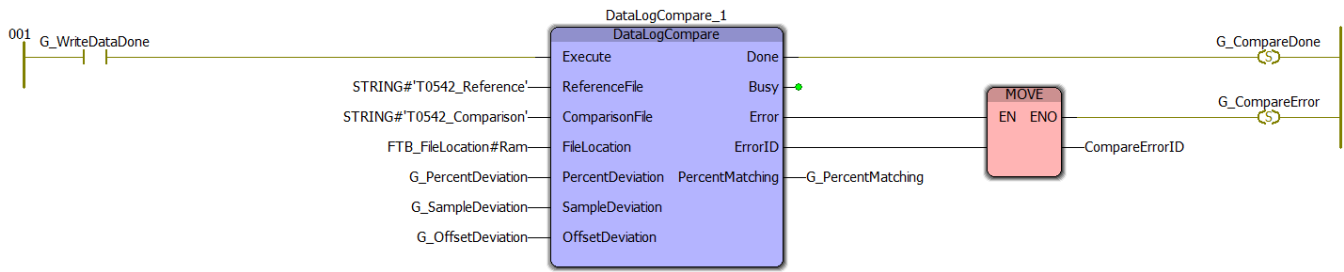
Example:

Initialize these structures for each item to be recorded by DataLogRecord. Each data set may require a different set of matching criteria.

```

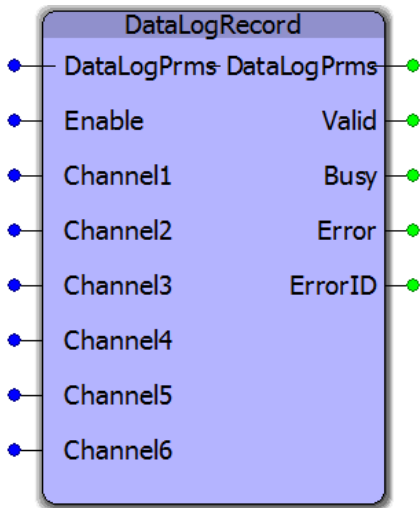
36 G_PercentDeviation[1]:=LREAL#5.0;          (* Channel 1 matching criteria *)
37 G_SampleDeviation[1]:=INT#10;
38 G_OffsetDeviation[1]:=LREAL#0.5;

```





DataLogRecord



This function block records data on up to 6 channels every scan and stores the data in DataLogPrms.Data[0] and DataLogPrms.Data[1] structures. The data could be downloaded and analyzed in Excel or other software, but the intention is to re-record at some future time under the same conditions, and use the [DataLogCompare](#) function block to analyze the two files for significant differences.

Library

FileRW Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	DataLogPrms	DataLoggingStruct	Structure containing the parameters for recording the data.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Channel1	LREAL	The value of Channel 1 data to be recorded every scan.	LREAL#0.0
V	Channel2	LREAL	The value of Channel 2 data to be recorded every scan.	LREAL#0.0
V	Channel3	LREAL	The value of Channel 3 data to be recorded every scan.	LREAL#0.0
V	Channel4	LREAL	The value of Channel 4 data to be recorded every scan.	LREAL#0.0
V	Channel5	LREAL	The value of Channel 5 data to be recorded every scan.	LREAL#0.0
V	Channel6	LREAL	The value of Channel 6 data to be recorded every scan.	LREAL#0.0
VAR_OUTPUT				

B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error) Busy will be high when recording data.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

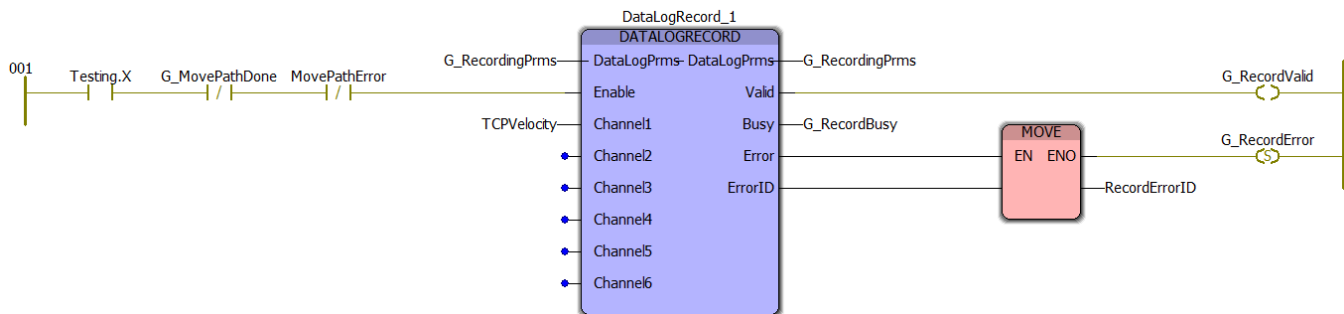
Notes

- Three function blocks work together in this solution: DataLogRecord, DataLogWrite, and DataLogCompare.
- Start and Stop the DataLogRecord function in conjunction with an event signifying when the data should be recorded. This is intended to be a single event, not a series of intermittent events. At some time after DataLogRecord is Valid, Execute the DataLogWrite function, typically in a slower task than DataLogRecord. Typical usage includes DataLogRecord in a task as fast as the Mechatrolink rate if necessary, DataLogWrite in a 32 to 64 mSec task, and DataLogCompare in a 32 to 64 mSec task. DataLogWrite and DataLogCompare work most most efficiently and quickly in the Default task.
- These functions use a double buffer technique, swapping from one to other. DataLogRecord accesses one buffer while DataLogWrite accesses the other. The two activities must never interfere with each other, or an error will be generated if there is a timing conflict when trying to switch buffers. If this occurs, it is possible that DataLogWrite must be executed at a faster interval.
- The comparison algorithm first checks for a match within the offset specified. If the data does not match, then it will check within the allowed percentage. If still a match cannot be determined, the process repeats by iterating through the number of samples before and after the equivalent time sample. If a SampleDeviation of 20 is provided, the process will check up to 20 samples before and 20 after the sample in question.

Error Description

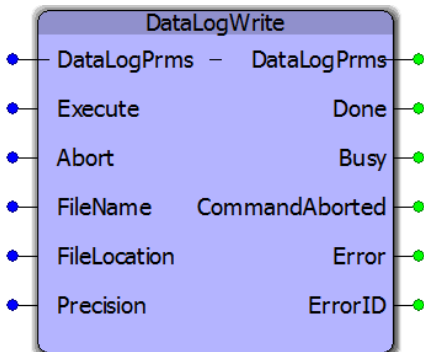
See the [Function Block ErrorID](#) list.

Example:





DataLogWrite



This function block writes data recorded by [DataLogRecord](#) to a CSV file on the controller.

Library

FileRW Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	DataLogPrms	DataLoggingStruct	Structure containing the parameters for writing the data to a CSV file.
VAR_INPUT			
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
V	Abort	BOOL	Cancel writing data to the file. CommandAborted output will be true once the function block successfully aborts.
V	FileName	STRING	The name of the file being written. Typically 'Reference' or 'Comparison'. Cannot be left blank. Do not include an extension, .txt will be automatically appended.
V	FileLocation	FTB_FileLocation	Specify where on the controller the Comparison file is stored as either: FTB_FileLocation#Ram or FTB_FileLocation#Flash
V	Precision	INT	The number of decimal places to write (0 - 15).
VAR_OUTPUT			
	CommandAborted	BOOL	True when the function block successfully aborts.
	Error	BOOL	True when an error occurs.
	ErrorID	INT	The error ID when an error occurs.
	Done	BOOL	True when the function block successfully completes.
	Busy	BOOL	True when the function block is busy.

B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error) Busy will be high when writing data to the file.
E	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

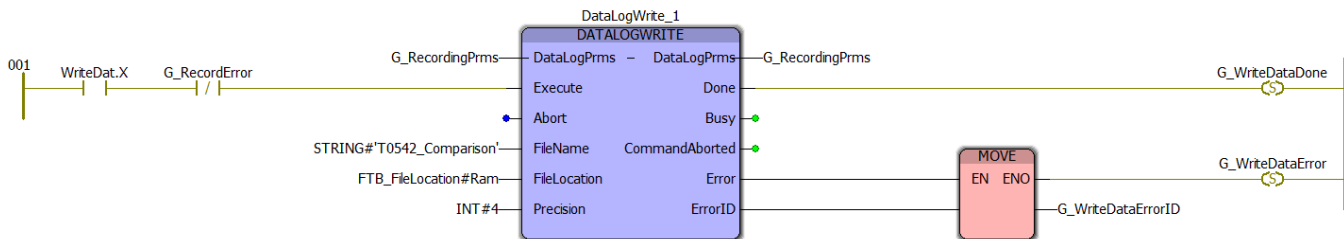
Notes

- Three function blocks work together in this solution: DataLogRecord, DataLogWrite, and DataLogCompare.
- Start and Stop the DataLogRecord function in conjunction with an event signifying when the data should be recorded. This is intended to be a single event, not a series of intermittent events. At some time after DataLogRecord is Valid, Execute the DataLogWrite function, typically in a slower task than DataLogRecord. Typical usage includes DataLogRecord in a task as fast as the Mechatrolink rate if necessary, DataLogWrite in a 32 to 64 mSec task, and DataLogCompare in a 32 to 64 mSec task. DataLogWrite and DataLogCompare work most most efficiently and quickly in the Default task.
- These functions use a double buffer technique, swapping from one to other. DataLogRecord accesses one buffer while DataLogWrite accesses the other. The two activities must never interfere with each other, error will be generated if there is a timing conflict when trying to switch buffers. If this occurs, it is possible that DataLogWrite must be executed at a faster interval.
- The comparison algorithm first checks for a match within the offset specified. If the data does not match, then it will check within the allowed percentage. If still a match cannot be determined, the process repeats by iterating through the number of samples before and after the equivalent time sample. If a SampleDeviation of 20 is provided, the process will check up to 20 samples before and 20 after the sample in question.

Error Description

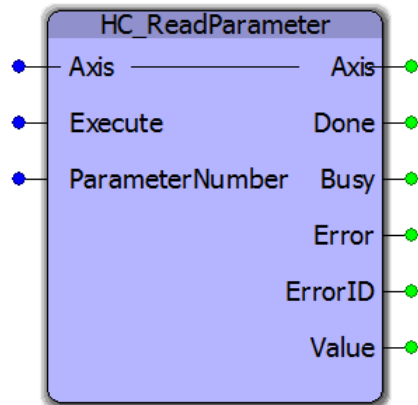
See the [Function Block ErrorID](#) list.

Example:





HC_ReadParameter



This Function Block is an extension of MC_ReadParameter from the PLCopenPlus_2_2a firmware library and reads configuration specific parameters from the appropriate XML configuration file(s). See the table below for the parameters supported by this function.

Parameters

*	Parameter	Data type	Description
VAR_IN_OUT			
B	Axis	AXIS REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
B	ParameterNumber	UINT	Hardware Configuration parameter number.
VAR_OUTPUT			
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
B	Value	LREAL	Value of the parameter.

Supported Parameters

Name	Parameter	R/W	Default	Min	Max	Comments
PositionScale	999	Write	1.0	1.0	1.0	
External Encoder Phase A	1016	Read/Write	0.0	0.0	1.0	Invert the signal to change the counting direction. Use LREAL#1.0 format to set TRUE or FALSE.
External Encoder Phase B	1017	Read/Write	0.0	0.0	1.0	Invert the signal to change the counting direction. Use LREAL#1.0 format to set TRUE or FALSE.
External Encoder Phase C	1018	Read/Write	0.0	0.0	1.0	Invert the signal to change the active state. Use LREAL#1.0 format to set TRUE or FALSE.
External Encoder Resolution	1047	Read/Write	1.0	10	16777216	
Load Type	1807	Read/Write	1.0	0	1	Rotary or linear
FeedConstant	1808	Read/Write	1.0	0.001	1000000000000	
OutputRatio	1815	Read/Write	1.0	1	2147483647	
MachineCycle	1833	Read/Write	1.0	0.001	1000000000000	
InputRatio	1834	Read/Write	1.0	1	2147483647	
User Units	1813	Read/Write	0.0	0	5	0=inches, 1=mm, um=2, 3=pulses, 4=d-degrees, 5=revolutions

Notes

- This function block includes some very large datatypes. On the MP2600iec controller, the memory can be exceeded when including this function. When using an MP2600iec, it may be required to edit the File R/W library's datatype file by modifying the size of `XMLLines: ARRAY[0..9999] OF STRING;` to `0..500`.
- Another recommendation when using this function is to reuse the same instance if several parameter reads are required. This will save considerable memory, important for all controller platforms.
- Finally, consider executing this function in the DEFAULT task to avoid the risk of a PLC watchdog, or execute it in a CYCLE task of 100 mSec.

Error Description

See the [Function Block ErrorID](#) list.

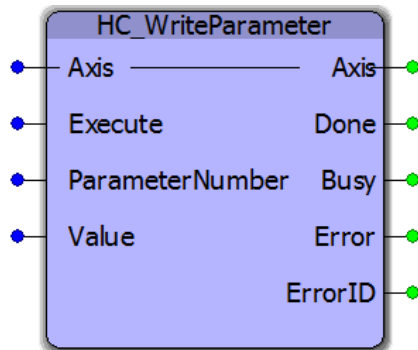
Related Function Blocks

[HC_WriteParameter](#): Writes a Hardware Configuration parameter.

[MC_WriteParameter](#): Writes an axis specific parameter available at run time.



HC_WriteParameter



This Function Block is an extension of MC_WriteParameter from the PLCopenPlus_2_2a firmware library and writes configuration specific parameters to the appropriate XML configuration file(s). See the table below for the parameters supported by this function. Changes to parameters written by this function requires controller power cycle or reboot to take effect.

Parameters

*	Parameter	Data type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
B	ParameterNumber	UINT	Hardware Configuration parameter number.	UINT#0
B	Value	LREAL	Value of the parameter to be updated.	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Supported Parameters

Name	Parameter	R/W	Default	Min	Max	Comments
PositionScale	999	Write	1.0	1.0	1.0	
External Encoder Phase A	1016	Read/Write	0.0	0.0	1.0	Invert the signal to change the counting direction Use LREAL#1.0 format to set TRUE or FALSE.
External Encoder Phase B	1017	Read/Write	0.0	0.0	1.0	Invert the signal to change the counting direction. Use LREAL#1.0 format to set TRUE or FALSE.
External Encoder Phase C	1018	Read/Write	0.0	0.0	1.0	Invert the signal to change the active state. Use LREAL#1.0 format to set TRUE or FALSE.
External Encoder Resolution	1047	Read/Write	1.0	10	16777216	
Load Type	1807	Read/Write	1.0	0	1	Rotary or linear
FeedConstant	1808	Read/Write	1.0	0.001	1000000000000	
OutputRatio	1815	Read/Write	1.0	1	2147483647	
MachineCycle	1833	Read/Write	1.0	0.001	1000000000000	
InputRatio	1834	Read/Write	1.0	1	2147483647	
User Units	1813	Read/Write	0.0	0	5	0=inches, 1=mm, um=2, 3=pulses, 4=d-degrees, 5=revolutions

Notes

- This function block includes some very large datatypes. On the MP2600iec controller, the memory can be exceeded when including this function. When using an MP2600iec, it may be required to edit the File R/W Library's datatype file by modifying the size of `XMLLines: ARRAY[0..9999] OF STRING;` to `0..500`.
- Another recommendation when using this function is to reuse the same instance if several parameter writes are required. This will save considerable memory, important for all controller platforms.
- Finally, consider executing this function in the DEFAULT task to avoid the risk of a PLC watchdog, or execute it in a CYCLE task of 100 mSec.

Error Description

See the [Function Block ErrorID](#) list.

Related Function Blocks

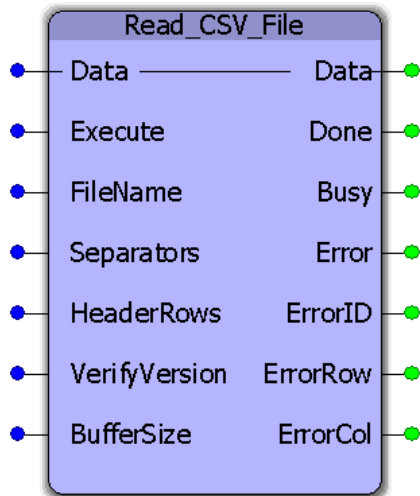
[HC_ReadParameter](#): Reads a Hardware Configuration parameter.

[MC_ReadParameter](#): Reads an axis specific parameter.

Example



Read_CSV_File



This function block will read CSV (ASCII) data from a file on the controllers flash or ram disk. The raw file data will be parsed and copied into a user defined data structure. This function block requires customization to accommodate application specific data requirements. Any variety of rows and columns and datatypes can be specified. Read_CSV_File must be customized to accommodate the application specific data. See the example customization below.

Library

File RW Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Data	MyDataStruct	A user customized data structure containing the definition of the rows and columns of data to be processed.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	FileName	STRING	The file to be read. Example STRING#'/-flash/user/data/mydata.csv'	STRING#''

V	Separators	SeparatorList	Optional. If unconnected, the default separator is a comma (BYTE#44) to detect each value column by column. If a different or multiple characters must be treated as a value separator, populate the SeparatorList with up to four byte values equating to the ASCII value of the separators.	Comma (BYTE#44)
V	HeaderRows	UINT	Optional. If connected, the value indicates the number of rows this function block must ignore before starting to look for actual data.	UINT#0
V	VerifyVersion	BOOL	Optional. If TRUE, this function block will expect the first line of the file to contain a version code for identifying the data format of the file, i.e columns, datatypes, etc.. This allows for future changes to the MyDataStruct while retaining the ability to parse older files created before a change was made to the structure of the file.	FALSE
V	BufferSize	UDINT	Specifies the number of bytes in the file to process at one time. If unconnected, the default is 2048 bytes. Buffer-Size can be adjusted up or down if necessary to accommodate various file sizes and will depend upon the CYCLIC task in which the Read_CSV_File function block is executed.	UDINT#2048
VAR_OUTPUT				
E	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	ErrorRow	INT	If Error is true and pertains to a problem with the source data, this value will indicate the location of processing when the error occurred.	
V	ErrorCol	INT	If Error is true and pertains to a problem with the source data, this value will indicate the location of processing when the error occurred.	

Notes

- Don't forget to include the ProConOS firmware library in the project. It is required for this function block.
- The filename must conform to 8.3 format, but is not case sensitive.
- Any separator can be specified provided it is an ASCII byte, and will not be confused with the actual data.
- Header rows are not required to contain the same number of separators as the data content. (Separators are not checked in the header rows.)
Supports Carriage Return and Line Feed as end of line delimiters.
- It takes 6 scans per processing of each BufferSize of data. If a file has 20480 bytes, and the BufferSize is 2048, and the function block is placed in a 100mSec scan, then the total time to process the file will be 60 scans, or 6 seconds. (20480/2048 * 6 * 100) = 6000 mSec.
- See Yaskawa's Youtube Webinar - [CSV File Transfer with the File_RW Template](#).

Error Description

See the [Function Block ErrorID](#) list.

Example Customization

Read_CSV_File must be customized to accommodate the application specific data. Some supporting functions used by Read_CSV_File (ReadBuffer and ReadValue) do not require customization and can remain in the File_RW_Toolbox. To effectively use this function, follow these steps:

1) Copy & paste the MyDataStruct and associated datatypes into the main project, and rename them to avoid conflict with MyDataStruct in the File_RW_Template.

```
64 (***** Structure information relating to a CSV file *****)
65 MyData : STRUCT
66   XData : LREAL;
67   YData : LREAL;
68   ZData : LREAL;
69 END_STRUCT;
70
71 MyDataArray : ARRAY [UINT#0..UINT#300] OF MyData;
72
73 MyDataStruct: STRUCT
74   File: MyDataArray;
75   Version:STRING; (* If file versioning is used, apply a unique value to allow the identification of different file formats *)
76   Columns:INT; (* Configure this value to indicate the number of columns in the data file. *)
77   Records:INT; (* This value will be updated by the function as the data is processed *)
78   MaxRecords:INT; (* Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above *)
79 END_STRUCT;
80
81 (***** Structure information relating to a CSV file *****)
```

2) Modify the "MyData" dataType definition shown above such that it represents the number of columns and the relevant data-types. An example follows:

```
2 (***** Job *****)
3 JobData : STRUCT
4   Move_X : DINT;
5   Move_Y : DINT;
6   Outs_01 : DINT;
7   Outs_02 : DINT;
8   Outs_03 : DINT;
9   Vel_X : BYTE;
10  Acc_X : BYTE;
11  Vel_Y : BYTE;
12  Acc_Y : BYTE;
13  Execute : INT;
14  Jump : BYTE;
15  Wait : INT;
16  Loop : BYTE;
17  AltX : BYTE;
18  LinkTo : INT;
19 END_STRUCT;
20
21 JobArray : ARRAY [UINT#0..UINT#3399] OF JobData;
22
23 JobStruct: STRUCT
24   Job: JobArray;
25   Version:STRING; (* If file versioning is used, apply a unique value to allow the identification of different file formats *)
26   Columns:INT; (* Configure this value to indicate the number of columns in the data file. *)
27   Records:INT; (* This value will be updated by the function as the data is processed *)
28   MaxRecords:INT; (* Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above *)
29 END_STRUCT;
30
31 (***** Job *****)
```

The 15 columns of data defined above relate to the data shown in the following Excel file. Notice that the data has three header rows before the actual data begins. In this case, set the HeaderRows function block input correctly at UINT#3, otherwise, the data will not be read properly.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Move_x	Move_y	Outs_01	Outs_02	Outs_03	Vel_x	Acc_x	Vel_y	Acc_y	Execute	Jump	Wait	Loop	Alt_x	Link_to
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
3	1E+08	1E+08	2.1E+09	2.1E+09	2.1E+09	100	100	100	100	999	100	9999	1	1	3400
4	1	1	1	0	3401	100	100	100	100	1	0	0	0	0	0
5	2	2	2	1	3400	100	100	100	100	1	0	0	0	0	0
6	3	3	3	2	3399	100	100	100	100	1	0	0	0	0	0
7	4	4	4	3	3398	100	100	100	100	1	0	0	0	0	0
8	5	5	5	4	3397	100	100	100	100	1	0	0	0	0	0
9	6	6	6	5	3396	100	100	100	100	1	0	0	0	0	0
10	7	7	7	6	3395	100	100	100	100	1	0	0	0	0	0
11	8	8	8	7	3394	100	100	100	100	1	0	0	0	0	0
12	9	9	9	8	3393	100	100	100	100	1	0	0	0	0	0
13	10	10	10	9	3392	100	100	100	100	1	0	0	0	0	0
14	11	11	11	10	3391	100	100	100	100	1	0	0	0	0	0
15	12	12	12	11	3390	100	100	100	100	1	0	0	0	0	0
16	13	13	13	12	3389	100	100	100	100	1	0	0	0	0	0
17	14	14	14	13	3388	100	100	100	100	1	0	0	0	0	0
18	15	15	15	14	3387	100	100	100	100	1	0	0	0	0	0
19	16	16	16	15	3386	100	100	100	100	1	0	0	0	0	0
20	17	17	17	16	3385	100	100	100	100	1	0	0	0	0	0
21	18	18	18	17	3384	100	100	100	100	1	0	0	0	0	0
22	19	19	19	18	3383	100	100	100	100	1	0	0	0	0	0
23	20	20	20	19	3382	100	100	100	100	1	0	0	0	0	0
24	21	21	21	20	3381	100	100	100	100	1	0	0	0	0	0
25	22	22	22	21	3380	100	100	100	100	1	0	0	0	0	0
26	23	23	23	22	3379	100	100	100	100	1	0	0	0	0	0
27	24	24	24	23	3378	100	100	100	100	1	0	0	0	0	0
28	25	25	25	24	3377	100	100	100	100	1	0	0	0	0	0
29	26	26	26	25	3376	100	100	100	100	1	0	0	0	0	0
30	27	27	27	26	3375	100	100	100	100	1	0	0	0	0	0
31	28	28	28	27	3374	100	100	100	100	1	0	0	0	0	0
32	29	29	29	28	3373	100	100	100	100	1	0	0	0	0	0
33	30	30	30	29	3372	100	100	100	100	1	0	0	0	0	0
34	31	31	31	30	3371	100	100	100	100	1	0	0	0	0	0
35	32	32	32	31	3370	100	100	100	100	1	0	0	0	0	0
36	33	33	33	32	3369	100	100	100	100	1	0	0	0	0	0
37	34	34	34	33	3368	100	100	100	100	1	0	0	0	0	0
38	35	35	35	34	3367	100	100	100	100	1	0	0	0	0	0
39	36	36	36	35	3366	100	100	100	100	1	0	0	0	0	0

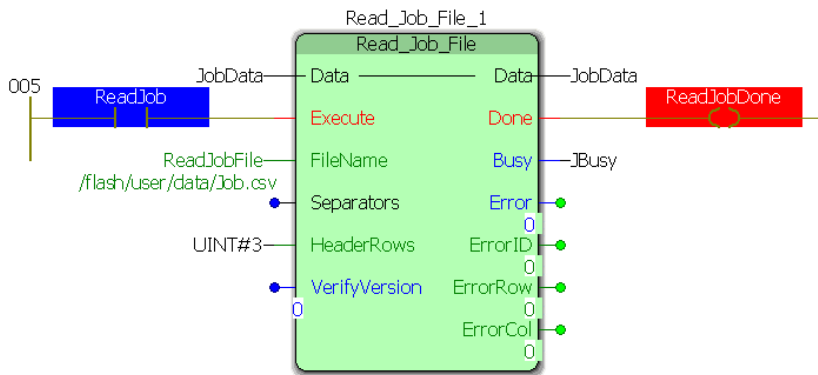
3) Initialize the data required for "MyDataStruct" as shown below. Most importantly, set Columns and MaxRecords.

```

12 ReadJobFile:='/flash/user/data/job.csv';
13 WriteJobFile:='/flash/user/data/JobW.csv';
14 JobData.Columns:=INT#15;
15 JobData.MaxRecords:=INT#3400; (* Set to same as DataType Definition *)

```

4) Copy & paste the Read_CSV_File function block into the main project so it can be customized. This will allow you to retain the original function in the template for future reference. Rename the function to avoid name conflict with Read_CSV_File in the Toolbox.



Variable	Value	Default value	Type
JobData			JobStruct
Job			JobArray
[0]			StruBlock
Move_X	1		DINT
Move_Y	1		DINT
Outs_01	1		DINT
Outs_02	0		DINT
Outs_03	3401		DINT
Vel_X	100		BYTE
Acc_X	100		BYTE
Vel_Y	100		BYTE
ACC_Y	100		BYTE
Execute	1		INT
Jump	0		BYTE
Wait	0		INT
Loop	0		BYTE
AIK	0		BYTE
LinkTo	0		INT
[1]			StruBlock
Move_X	2		DINT
Move_Y	2		DINT
Outs_01	2		DINT
Outs_02	1		DINT
Outs_03	3400		DINT
Vel_X	100		BYTE
Acc_X	100		BYTE
Vel_Y	100		BYTE
ACC_Y	100		BYTE
Execute	1		INT
Jump	0		BYTE
Wait	0		INT
Loop	0		BYTE
AIK	0		BYTE
LinkTo	0		INT
[2]			StruBlock

Customizing the code in the function block

- To customize the function block, go to the variables grid and rename the datatype used as the VAR_IN_OUT to the datatype you customized in step 2 above (Use the name as modified from ST code line 23 above).
- Locate the comments near the middle of the Read_CSV_File function indicating the area to be customized. Modify the lines that convert the STRING data from the file into the MyDataStruct structure.

```

151 (* Data is ignored if HeaderRows are specified until after the specified number of header rows have been read. *)
152 IF ReadColumn.Valid AND HeaderRowsRead THEN
153
154 0
155 CASE UINT_TO_INT(VersionCode) OF (* Extract the CSV values from the file as specified by the VersionCode *)
156     CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
157     CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
158     CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
159
160 0: (***** Non Versioned file format *****
161
162     CASE ActiveColumn OF
163         1:Data.Job[Row].Move_X :=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
164         2:Data.Job[Row].Move_Y:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
165         3:Data.Job[Row].Outs_01:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
166         4:Data.Job[Row].Outs_02:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
167         5:Data.Job[Row].Outs_03:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
168         6:Data.Job[Row].Vel_X:=STRING_TO_BYTE(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
169         7:Data.Job[Row].Acc_X:=STRING_TO_BYTE(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
170         8:Data.Job[Row].Vel_Y:=STRING_TO_BYTE(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
171         9:Data.Job[Row].ACC_Y:=STRING_TO_BYTE(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
172         10:Data.Job[Row].Execute:=STRING_TO_INT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
173         11:Data.Job[Row].Jump:=STRING_TO_BYTE(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
174         12:Data.Job[Row].Wait:=STRING_TO_INT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
175         13:Data.Job[Row].Loop:=STRING_TO_BYTE(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
176         14:Data.Job[Row].AltX:=STRING_TO_BYTE(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
177         15:Data.Job[Row].LinkTo:=STRING_TO_INT(ReadColumn.Value); (* last one handled below *)
178     END_CASE;
179
180 1822
181 FALSE ***** CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
182     CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
183     CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
184
185 2
186 FALSE ELSE
187     UnsupportedCase:=TRUE;
188     END_CASE;
189
190 IF ReadColumn.EOL THEN (* The end of a row was detected *)
191     IF ActiveColumn=Data.Columns THEN (* Set to the number of columns in the data *)
192         Row:=Row + INT#1;
193         Data.Records:=Row;
194         ActiveColumn:=INT#1;
195     ELSE
196         RowError:=TRUE;
197     END_IF;
198 END_IF;
199
200 Y_ReadAlarm_2(Enable:=TRUE);
201 IF Y_ReadAlarm_2.Valid THEN
202     ConversionError:=(Y_ReadAlarm_2.AlarmID = UDINT#16#340CC134);
203 END_IF;

```

Customizing for file versioning

The function has the capability to read multiple versions of the same file. For example, assume that initially, the design requires a data file to contain 4 columns of data to be used as INT. Later, after some machines are in the field, a design change requires that the data file must now contain 5 columns of DINT. If a version code is applied as the first row, the function block can determine how to read the file for any number of variations. That may come later. This will allow the use of older data files as well as newer formats.

Original file specification:

```

original.txt - Notepad
File Edit Format View Help
20111118
3,4,7,4
234,456,344,3223
984,435,7346,333
123,4534233,9445

```

Modified file specification:

```

modified.txt - Notepad
File Edit Format View Help
20120105
767653,4786789,742323,4758656,78654
23645304,45456456,34756434,89076456,32923
98641214,4354395,7534111,7300846,3332439
1276543,4534233,9445,789786,90753

```

To use file versioning, follow the steps below:

1. Set the VerifyVersion function block input to TRUE.
2. The first line of the data file must contain a version code. The version code does NOT count as a header row. See the graphics above showing original and modified file specification
3. Customize the DataType to reflect the most current data specification.

Original DataType:

```

66 (***** JobRef *****)
67   PartData : STRUCT
68     Ref12 : INT;
69     Ref34 : INT;
70     Ref56 : INT;
71     Ref78 : INT;
72   END_STRUCT;
73
74   JobRefArray : ARRAY [UINT#0..UINT#401] OF PartData;
75
76   JobRefStruct: STRUCT
77     Ref: JobRefArray;
78     Version:STRING; (* If file versioning is used, apply a unique value to allow the identification of different file formats *)
79     Columns:INT; (* Configure this value to indicate the number of columns in the data file. *)
80     Records:INT; (* This value will be updated by the function as the data is processed *)
81     MaxRecords:INT; (* Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above *)
82   END_STRUCT;
83 (***** JobRef *****)

```

Modified DataType:

```

66 (***** JobRef *****)
67   PartData : STRUCT
68     Ref12 : DINT;
69     Ref34 : DINT;
70     Ref56 : DINT;
71     Ref78 : DINT;
72     Ref91 : DINT;
73   END_STRUCT;
74
75   JobRefArray : ARRAY [UINT#0..UINT#401] OF PartData;
76
77   JobRefStruct: STRUCT
78     Ref: JobRefArray;
79     Version:STRING; (* If file versioning is used, apply a unique value to allow the identification of different file formats *)
80     Columns:INT; (* Configure this value to indicate the number of columns in the data file. *)
81     Records:INT; (* This value will be updated by the function as the data is processed *)
82     MaxRecords:INT; (* Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above *)
83   END_STRUCT;
84 (***** JobRef *****)

```

- 3) Customize the Read-CSV_File function block to determine if the version code detected is supported.

Original code:

```

107 (***** MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
108 (***** MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
109 (***** MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
110
111 (* Verify that the file version matches one of the formats supported by this function (ADD MORE COMPARISONS AS NEEDED) *)
112 IF EQ_STRING(Data.Version, '20111118') THEN
113   VersionCode:=UINT#1;
114 END_IF;
115
116 (***** MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
117 (***** MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
118 (***** MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)

```

Modified code:

```

107 (***** MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
108 (***** MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
109 (***** MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
110
111 (* Verify that the file version matches one of the formats supported by this function (ADD MORE COMPARISONS AS NEEDED) *)
112 IF EQ_STRING(Data.Version, '20111118') THEN
113   VersionCode:=UINT#1;
114 ELSIF EQ_STRING(Data.Version, '20120105') THEN
115   VersionCode:=UINT#2;
116 END_IF;
117
118 (***** MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
119 (***** MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)
120 (***** MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS *****)

```

- 4) Customize the Read_CSV_File function block to read multiple versions.

Original code:

```

154 CASE UINT_TO_INT(VersionCode) OF (* Extract the CSV values from the file as specified by the VersionCode *)
155 (*****
156 CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
157 (*****
158 CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
159
160 1: (*****
161 2011118 file format *****
162
163 CASE UINT_TO_INT(ActiveColumn) OF
164 1:Data.Ref[Row].Ref12:=STRING_TO_INT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
165 2:Data.Ref[Row].Ref34:=STRING_TO_INT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
166 3:Data.Ref[Row].Ref56:=STRING_TO_INT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
167 4:Data.Ref[Row].Ref78:=STRING_TO_INT(ReadColumn.Value); (* last one handled below *)
168 END_CASE;
169
170 CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
171 (*****
172 CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
173 (*****
174 CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
175
176 ELSE
177 UnsupportedCase:=TRUE;
178 END_CASE;
179

```

Modified code:

```

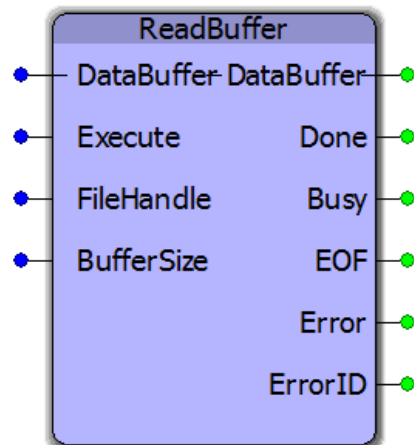
152 CASE UINT_TO_INT(VersionCode) OF (* Extract the CSV values from the file as specified by the VersionCode *)
153 (*****
154 CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
155 (*****
156 CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
157
158 1: (*****
159 2011118 file format *****
160
161 CASE UINT_TO_INT(ActiveColumn) OF
162 1:Data.Ref[Row].Ref12:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
163 2:Data.Ref[Row].Ref34:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
164 3:Data.Ref[Row].Ref56:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
165 4:Data.Ref[Row].Ref78:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
166 5:Data.Ref[Row].Ref78:=DINT#0; (* Initialize new data *) (* last one handled below *)
167 END_CASE;
168
169 2: (*****
170 20120105 file format *****
171
172 CASE UINT_TO_INT(ActiveColumn) OF
173 1:Data.Ref[Row].Ref12:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
174 2:Data.Ref[Row].Ref34:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
175 3:Data.Ref[Row].Ref56:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
176 4:Data.Ref[Row].Ref78:=STRING_TO_DINT(ReadColumn.Value); ActiveColumn:=ActiveColumn + INT#1;
177 5:Data.Ref[Row].Ref78:=STRING_TO_DINT(ReadColumn.Value); (* last one handled below *)
178 END_CASE;
179
180 (*****
181 CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
182 (*****
183 CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
184 (*****
185 CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS *****
186
187 ELSE
188 UnsupportedCase:=TRUE;
189 END_CASE;
190

```

NOTE: The capability of the function block to read multiple file versions is limited by the changes that can be made to the DataType Definition. It is not practical to use the version code to read completely different data formats. Make two copies of the Read_CSV_File and customize accordingly.



ReadBuffer



This function block will read an ASCII value from the DataBuffer. It determines where to locate the value by referencing information in the DataBuffer. It determines the end of a line by looking for a CR (BYTE#13) or LF (BYTE#10) or both.

Library

File RW Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	DataBuffer	ByteBufferStruct	Structure containing a character buffer and other information for managing parsing operations.	
VAR_INPUT			Default	
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	FileHandle	UINT	The FileHandle as output from the FILE_OPEN function block from the ProConOS firmware library. The file must be opened before calling ReadBuffer.	UINT#0
V	BufferSize	UDINT	Specify the portion of the file to read during each task interval.	UDINT#2048
VAR_OUTPUT				
B	Done	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	

B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
V	EOF	BOOL	EOF = End of File.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Error Description

See the [Function Block ErrorID](#) list.

Example

See the [Read_CSV_File](#) function block to learn more about how ReadBuffer can be utilized.

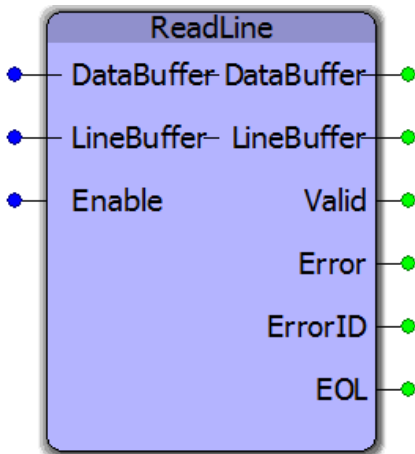
```

195 (*-----*)
196 (*-----*)
197 (*-----*)
198 (*-----*)
199 (* File Open *)
200 OpenExe:=iActive AND OpenExeReq;
201 FILE_OPEN_1(Execute:=OpenExe, Name:=FileName);
202 OpenError:=FILE_OPEN_1.Error AND (FILE_OPEN_1.ErrorID <> UINT#4); (* Ignore Error if File is already open *)
203
204
205
206 (* Read n rows of data from the file and puts it in "DataBuffer" *)
207 (* The ReadBuffer FB is from the File_RW Toolbox. It prepares the DataBuffer by looking for the last delimiter, the sets the filepointer back to the start of the next PARTIAL line
208 ReadExe:=iActive AND ((ReadBufferReq OR ReadBuffer_1.Busy) AND NOT(ReadBuffer_1.Done) OR ReadError);
209 ReadBuffer_1(DataBuffer:=DataBuffer, Execute:=ReadExe, FileHandle:=FileHandle, BufferSize:=iFileBufferSize);
210 DataBuffer:=ReadBuffer_1.DataBuffer;
211 ReadBusy:=ReadBuffer_1.Busy;
212 ReadError:=ReadBuffer_1.Error;
213 ReadErrorID:=ReadBuffer_1.ErrorID;
214
215
216
217 (* Close the file *)
218 CloseExe:=iActive AND ((CloseExeReq AND NOT(Complete)) OR Error);
219 FILE_CLOSE_1(Execute:=CloseExe, Handle:=FileHandle);
220 CloseError:=FILE_CLOSE_1.Error AND (FILE_CLOSE_1.ErrorID <> UINT#1);

```




ReadLine



This function block will read an ASCII value from the DataBuffer. It determines where to locate the value by referencing information in the DataBuffer. It determines the end of a line by looking for a CR (BYTE#13) or LF (BYTE#10) or both.

Library

File RW Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	DataBuffer	ByteBufferStruct	Structure containing a character buffer and other information for managing parsing operations.	
V	LineBuffer	ByteBufferStruct	Structure containing a character buffer and other information for managing parsing operations. This is a subset of the data in DataBuffer; it will contain one line of data based on the DataBuffer.CharPointer when the function is executed.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				
E	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	EOL	BOOL	EOL = End of Line. This bit will be set if one or both of the ASCII characters CR LF (bytes 13 and 10 Hex) were found while searching for Separators.	

B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Error Description

See the [Function Block ErrorID](#) list.

Example

This ST code snippet assumes that the ReadBuffer function block was previously executed, which copied file contents into DataBuffer.

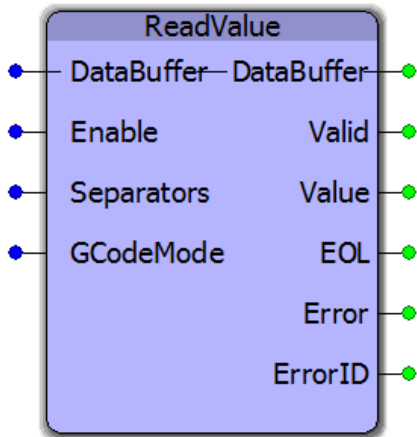
```

215  (***** File version 20160328p *****)
216  CASE ActiveSection OF
217  1: (* Next ReadLine is to get past this section, which is only for user reference *)
218     ReadLine_2(DataBuffer:=DataBuffer2, Enable:=TRUE, LineBuffer:=HeaderRow);
219     DataBuffer2:=ReadLine_2.DataBuffer;
220     ReadHeaderError:=ReadLine_2.Error;
221     HeaderRow:=ReadLine_2.LineBuffer;
222     HeaderValid:=ReadLine_2.Valid;
223
224     BUF_TO_STRING_1(REQ:=TRUE, BUF_OFFS:=DINT#0, BUF_CNT:=UINT_TO_DINT(HeaderRow.CharsRead), BUFFER:=HeaderRow.Char, DST:=StringLine);
225     HeaderRow.Char:=BUF_TO_STRING_1.BUFFER;
226     StringLine:=BUF_TO_STRING_1.DST;
227     err:=BUF_TO_STRING_1.Error;
228     sts:=BUF_TO_STRING_1.STATUS;
229
230     IF BUF_TO_STRING_1.DONE THEN
231         IF FIND(StringLine, 'Firmware') > INT#0 THEN
232             IncSection:=TRUE;
233         ELSE
234             DataError:=TRUE;
235         END_IF;
236     ELSE
237         StringConvError:=TRUE;
238         StringErrorID:=INT_TO_UINT(BUF_TO_STRING_1.STATUS);
239     END_IF;
240
241  2: (* Next ReadLine is to get past this section, which is only for user reference *)
242     ReadLine_2(DataBuffer:=DataBuffer2, Enable:=TRUE, LineBuffer:=HeaderRow);
243     DataBuffer2:=ReadLine_2.DataBuffer;
244     ReadHeaderError:=ReadLine_2.Error;
245     HeaderRow:=ReadLine_2.LineBuffer;
246     HeaderValid:=ReadLine_2.Valid;
247
248     BUF_TO_STRING_1(REQ:=TRUE, BUF_OFFS:=DINT#0, BUF_CNT:=UINT_TO_DINT(HeaderRow.CharsRead), BUFFER:=HeaderRow.Char, DST:=StringLine);
249     HeaderRow.Char:=BUF_TO_STRING_1.BUFFER;
250     StringLine:=BUF_TO_STRING_1.DST;
251     err:=BUF_TO_STRING_1.Error;
252     sts:=BUF_TO_STRING_1.STATUS;
253
254     IF BUF_TO_STRING_1.DONE THEN
255         IF FIND(StringLine, 'Controller') > INT#0 THEN
256             IncSection:=TRUE;
257         ELSE
258             DataError:=TRUE;
259         END_IF;
260     ELSE
261         StringConvError:=TRUE;
262         StringErrorID:=INT_TO_UINT(BUF_TO_STRING_1.STATUS);
263     END_IF;

```



ReadValue



This function block will read an ASCII value from the DataBuffer. It determines where to locate the value by referencing information in the DataBuffer and the Separators VAR_INPUT.

Library

File RW Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	DataBuffer	ByteBufferStruct	Structure containing a character buffer and other information for managing parsing operations.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Separators	SeparatorList	Optional. If unconnected, the default separator is a comma (BYTE#44) to detect each value column by column. If a different or multiple characters must be treated as a value separator, populate the SeparatorList with up to four byte values equating to the ASCII separators.	Comma (BYTE#44)
V	GCodeMode	BOOL	If set, it causes the function to interpret space characters or a change from numeric to alpha as a delimiter. For example, when ReadValue is passed a DataBuffer containing "G1X10Y15" it will return "G1" on the first call, "X10" on the second call, and "Y15" on the third.	UINT #0
VAR_OUTPUT				

E	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	Value	STRING	The ASCII value copied from the DataBuffer.
V	EOL	BOOL	EOL = End of Line. This bit will be set if one or both of the ASCII characters CR LF (bytes 13 and 10 Hex) were found while searching for Separators.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Error Description

See the [Function Block ErrorID](#) list.

Example

This ST code snippet assumes that the ReadBuffer function block was previously executed, which copied file contents into DataBuffer.

See the [Read_CSV_File](#) function block to learn more about how ReadBuffer can be utilized. In that function, "ReadVersion" and "ReadColumn" are ReadValue function blocks.

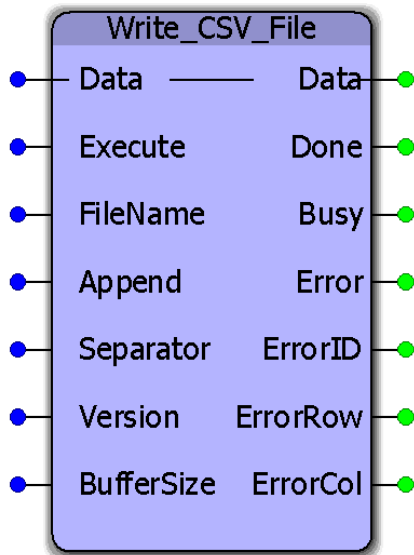
```

164         ReadCode.DataBuffer:=DataBuffer;
165         ReadCode.GCodeMode:=TRUE;
166         ReadCode.Separators:=Separators;
167         ReadCode(Enable:=DataBuffer.Charbuffer > UINT#0);          (* Execute the ReadValue function from File Read Write Toolbox,
168         DataBuffer:=ReadCode.DataBuffer;                          (* Byte buffer of data given to us by calling function *)
169         StringValue:=ReadCode.Value;                               (* This is the extracted parameter value *)
170         ValueError:=ReadCode.Error;
171
172         IF ReadCode.Valid AND (LEN(StringValue) > INT#1) THEN      (* Reason for checking length of StringValue is to ignore blank

```



Write_CSV_File



This function block will format and write a CSV (ASCII) file to the controllers flash or ram disk. The original data is a user specified structure. This function block requires customization to accommodate application specific data requirements. Any variety of rows and columns and datatypes can be customized. See the example customization below.

Library

File RW Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Data	MyDataStruct	A user customized data structure containing the information (possibly still in binary format) to be written to a CSV file.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, this function block will prepare to engage the RampIn cam profile at the master position specified in the BlendData structure.	FALSE
V	FileName	BOOL	The file to be written. Example: STRING#'ram-disk/user/data/mydata.csv'	STRING#''

V	Append	BOOL	This flag indicates whether to delete an existing file and create new data, or add to an existing file. If Append-d=TRUE, data will be appended. Data.MaxRecords must be set by the user to indicate the number of rows in MyDataStruct to append.	FALSE
V	Separator	BYTE	The byte value of the ASCII character to be used for separating values of data on a line. If unconnected, the comma (BYTE#44) will be used.	BYTE#44
V	Version	UDINT	Optional. If used, this function block has the ability to be customized to select between multiple output formats.	UDINT#0
V	BufferSize	UDINT	Specifies the number of bytes in the file to process at one time. If unconnected, the default is 2048 bytes. BufferSize can be adjusted up or down if necessary to accommodate various file sizes and will depend upon the CYCLIC task in which the Read_CSV_File function block is executed.	UDINT#0
VAR_OUTPUT				
B	Done	BOOL	Set high when the axis or group is synchronized with the axis or group it is commanded to follow. Synchronized means that the two are position locked, any transitional period required to achieve synchronization has been completed.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	ErrorRow	INT	If Error is true and pertains to a problem with the source data, this value will indicate the location of processing when the error occurred.	
V	ErrorCol	INT	If Error is true and pertains to a problem with the source data, this value will indicate the location of processing when the error occurred.	

Notes

- Do not reference this function block from your project. Follow the customization examples below to make a renamed copy of it and add it to your project.
- Don't forget to include the ProConOS firmware library in the project. It is required for this function block.
- It is strongly recommended to write files only to the Ramdisk portion of memory, not flash. Ramdisk is a temporary storage location, so the file should be read by another device using an HTTP file get command.
- See Yaskawa's Youtube Webinar - [CSV File Transfer with the File_RW Template](#).

Error Description

See the [Function Block ErrorID](#) list.

Customization Example 1

Write_CSV_File must be customized to accommodate your data. Some supporting functions used by Write_CSV_File (ReadBuffer and ReadValue) do not require customization and can remain in the File_RW_Toolbox. Two locations requiring customization are identified in the function block by several rows of comments indicating the need to customize. To effectively use this function, follow these steps:

1) Copy & paste the MyDataStruct and associated datatypes into your project, and rename them to avoid conflict with MyDataStruct in the File_RW_Template.

```

64 (***** Structure information relating to a CSV file *****)
65 MyData : STRUCT
66   XData : LREAL;
67   YData : LREAL;
68   ZData : LREAL;
69 END_STRUCT;
70
71 MyDataArray : ARRAY [UINT#0..UINT#300] OF MyData;
72
73 MyDataStruct: STRUCT
74   File: MyDataArray;
75   Version:STRING; (* If file versioning is used, apply a unique value to allow the identification of different file formats *)
76   Columns:INT; (* Configure this value to indicate the number of columns in the data file. *)
77   Records:INT; (* This value will be updated by the function as the data is processed *)
78   MaxRecords:INT; (* Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above *)
79 END_STRUCT;
80
81 (***** Structure information relating to a CSV file *****)

```

2) Modify the "MyData" dataType definition shown above such that it represents the data to be written. An example follows which shows a customized datatype:

```

2 (***** Job *****)
3 JobData : STRUCT
4   Move_X : DINT;
5   Move_Y : DINT;
6   Outs_01 : DINT;
7   Outs_02 : DINT;
8   Outs_03 : DINT;
9   Vel_X : BYTE;
10  Acc_X : BYTE;
11  Vel_Y : BYTE;
12  Acc_Y : BYTE;
13  Execute : INT;
14  Jump : BYTE;
15  Wait : INT;
16  Loop : BYTE;
17  AltX : BYTE;
18  LinkTo : INT;
19 END_STRUCT;
20
21 JobArray : ARRAY [UINT#0..UINT#3399] OF JobData;
22
23 JobStruct: STRUCT
24   Job: JobArray;
25   Version:STRING; (* If file versioning is used, apply a unique value to allow the identification of different file formats *)
26   Columns:INT; (* Configure this value to indicate the number of columns in the data file. *)
27   Records:INT; (* This value will be updated by the function as the data is processed *)
28   MaxRecords:INT; (* Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above *)
29 END_STRUCT;
30
31 (***** Job *****)

```

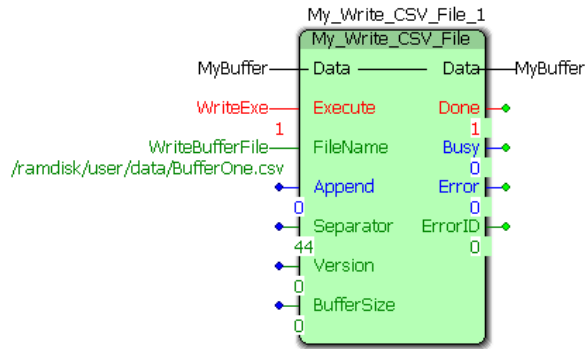
3) Initialize the data required for "MyDataStruct" as shown below. Most importantly, set Columns and MaxRecords. MaxRecords indicates how many lines of data are to be written to the file. In the case of Append mode =TRUE, set MaxRecords to the number of lines from the MyDataStruct to be appended. Appending always starts from the first line (array element 0) of the structure and adds data to the end of the file. It is not necessary to initialize (clear) the other data elements beyond MaxRecords that may be from a previous use.

```

12 ReadJobFile:= '/flash/user/data/job.csv';
13 WriteJobFile:= '/flash/user/data/JobW.csv';
14 JobData.Columns:=INT#15;
15 JobData.MaxRecords:=INT#3400; (* Set to same as DataType Definition *)

```

4) Copy & paste the Write_CSV_File function block into your main project so it can be customized. This will allow you to retain the original function in the template for future reference. Rename the function to avoid name conflict with Write_CSV_File in the Toolbox. To copy & paste the function block, open a second copy of MotionWorks IEC, and open the File_Read_Write toolbox as a project. From the second MotionWorks IEC, copy & paste the function block into your project.



Customizing the code in the function block

5) To customize the function block, go to the variables grid and rename the datatype used as the VAR_IN_OUT to the datatype you customized in step 2 above (Use the name as modified from ST code line 23 above).

6) Locate the comments near the middle of the Write_CSV_File function indicating the area to be customized. Modify the lines that convert binary data from the MyDataStruct structure to STRING data for the file.

Customizing for file versioning

The function has the capability to write multiple versions of the same structure. For example, a portion of the data from the structure can be written to one file, and a different set of data can be written to another file.

To use file versioning, follow the steps below:

- 1) Set the 'Version' function block input to a unique value (Non zero).
- 2) Customize the DataType to reflect the most current data specification.

Original DataType:

```

66 (***** JobRef *****)
67   PartData : STRUCT
68     Ref12 : INT;
69     Ref34 : INT;
70     Ref56 : INT;
71     Ref78 : INT;
72   END_STRUCT;
73
74   JobRefArray : ARRAY [UINT#0..UINT#401] OF PartData;
75
76   JobRefStruct: STRUCT
77     Ref: JobRefArray;
78     Version:STRING; (* If file versioning is used, apply a unique value to allow the identification of different file formats *)
79     Columns:INT; (* Configure this value to indicate the number of columns in the data file. *)
80     Records:INT; (* This value will be updated by the function as the data is processed *)
81     MaxRecords:INT; (* Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above *)
82   END_STRUCT;
83 (***** JobRef *****)

```

Modified DataType:

```

66 (***** JobRef *****)
67   PartData : STRUCT
68     Ref12 : DINT;
69     Ref34 : DINT;
70     Ref56 : DINT;
71     Ref78 : DINT;
72     Ref91 : DINT;
73   END_STRUCT;
74
75   JobRefArray : ARRAY [UINT#0..UINT#401] OF PartData;
76
77   JobRefStruct: STRUCT
78     Ref: JobRefArray;
79     Version:STRING; (* If file versioning is used, apply a unique value to allow the identification of different file formats *)
80     Columns:INT; (* Configure this value to indicate the number of columns in the data file. *)
81     Records:INT; (* This value will be updated by the function as the data is processed *)
82     MaxRecords:INT; (* Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above *)
83   END_STRUCT;
84 (***** JobRef *****)

```

3) Customize the Write_CSV_File function block to determine if a specific version if the file should be written.

Original code:

Modified code:

4) Customize the Write_CSV_File function block to write multiple versions.

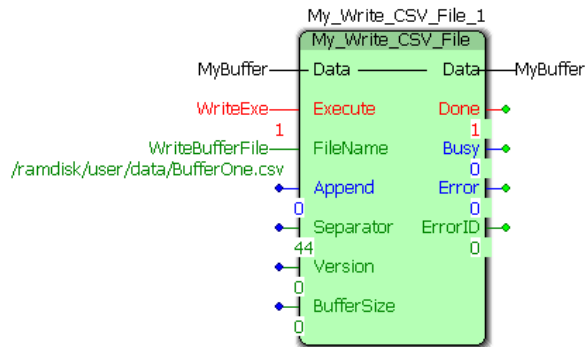
Original code:

Modified code:

Application Example

Set MyBuffer.MaxRecords to the number of items to be written to the file.

Variable	Value	Default value
MyBuffer		
Buff		
[0]		
Sensor1Data	8098925.4772730	
Sensor2Data	8098927.3813410	
[1]		
Sensor1Data	8098929.3104715	
Sensor2Data	8098931.2347956	
[2]		
Sensor1Data	8098933.0863352	
Sensor2Data	8098934.8403711	
[3]		
Sensor1Data	8098936.5405054	
Sensor2Data	8098938.2251902	
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		
Sensor1Data	8098958.1382532	
Sensor2Data	8098959.8236246	
[10]		
Sensor1Data	0.0000000	
Sensor2Data	0.0000000	





Getting Started with Group Toolbox

Group Toolbox is only supported on the MP3000iec series controllers.

Requirements for v375

Firmware version 3.7.4 or higher is strongly recommended.

To use the Group Toolbox, the main project must also contain the following:

Firmware libraries:

- YMotion
- YCoordinatedMotion
- PROCONOS
- Y_DeviceComm - if using [Read_GCode_Stream](#), or [GroupCommManager](#).

User libraries:

The following User Libraries must be listed above the Group Toolbox and in the following order:

It is strongly recommended to use the latest version of all supporting libraries, even though in some cases older versions are acceptable.

- DataTypes_Toolbox
- Math_Toolbox
- Yaskawa Toolbox - **v375 required, new datatype added.**
- PLCopen_Toolbox
- PLCopenPart4
- FileRW Toolbox - if using G-Code function blocks.
- Comm Toolbox - if using [Read_GCode_Stream](#) or [GroupCommManager](#). **v374 required.**



Getting Started with G-Code

Complete Solution: Yaskawa Compass

Additional resources and products are available from Yaskawa to make a complete G-Code solution.

- 1) Yaskawa Compass - Customizable CNC software interface. Part number PDE-U-GUICE.
- 2) Compass Configuration and Customization Guide. Document number [AN.MWIEC.06](#).
- 3) MotionWorks IEC template project and required libraries. Document number [AN.MPIEC.35](#).

The G-Code operation is divided into two main activities; data processing and motion processing.

Data Processing

This toolbox contains support for reading G-Codes in two different ways; use either the [Read_GCode_File](#) or [Read_GCode_Stream](#) function blocks to prepare the data for motion on the mechanism. For the best performance, Yaskawa recommends executing either of these functions in the DEFAULT task with its Watchdog disabled. See the examples in the function block pages which demonstrate how to link the two activities (Data Processing and Motion Processing) together. The data link between the data processing and motion is the [MC_PATH_DATA_REF](#) structure.

Data is managed in a circular buffer, meaning the path length can be virtually infinite. The default size of [MC_PATH_DATA_REF](#) is 500 instructions. Contact Yaskawa for applications that may require a larger buffer.

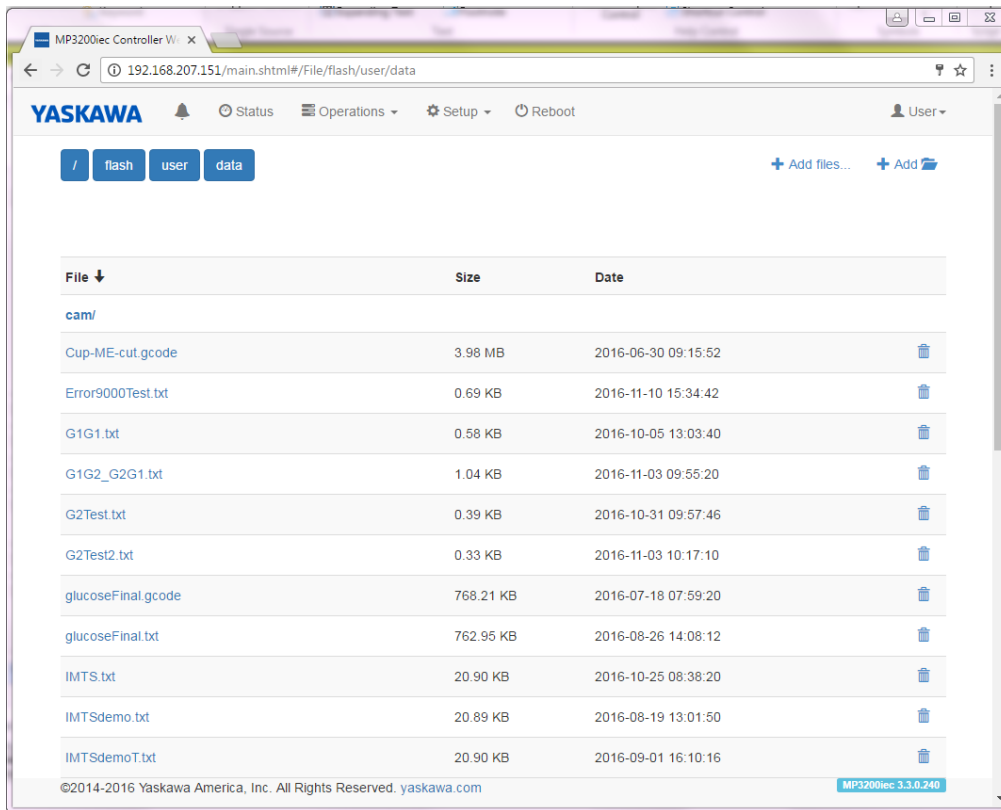
Tool Compensation (G41, G42) is calculated in the data processing section. If a different tool is selected from a source other than the G-Code data itself (some other PLC code writes to [ToolStruct](#) to change the tool), changes will not take effect until [Read_GCode_File](#) or [Read_GCode_Stream](#) have processed the data again.

File Method:

G-Code data must be loaded onto the Flash or ramdisk file system of the MPiec controller. This can be accomplished using the web sever, or if an automated method is required, files can be sent to the MPiec controller using the HTTP FILE POST method. Click [here](#) for C Sharp application notes on [www.yaskawa.com](#) for transferring files. Please note that [www.yaskawa.com](#) requires a login to access these documents. Writing files to flash is only recommended if the same part file will be used for an extended period of time. (Frequent flash writes are not recommended.) Otherwise, consider writing files to ramdisk, or stream the data via an Ethernet socket.

If the path is small such that the entire contents can fit into [MC_PATH_DATA_REF](#) at once, the Path can be re executed over and over by re executing [MC_MovePath](#). There is no need to re read the data with [Read_GCode_File](#).

The following image is the MPiec controller's web interface showing the "/Flash/User/Data" directory.



Streaming Method

This solution requires a PC application program such as Yaskawa Compass to communicate with the MPiec controller.

The [Read_GCode_Stream](#) function block will listen for data on the user specified port to receive the incoming byte stream of G-Code instructions.

When using this method, the MPiec controller sends status information ([StreamStruct](#)) via UDP packets back to the PC application at 50 mSec intervals. The packet contains position, buffer status data, and other info. Buffer status information is important for the PC application to throttle the G-Code stream effectively. The data buffers must not be overloaded nor should they be starved. If the data stops and all motion becomes completed, MC_MovePath will remain Busy, waiting for new commands. This is a possible under-run condition, or a normal pause between motion sequences that may be intended.

A DLL is available for use when creating a custom PC application. See [AN.MPIEC.24](#) on www.yaskawa.com.

Motion Processing

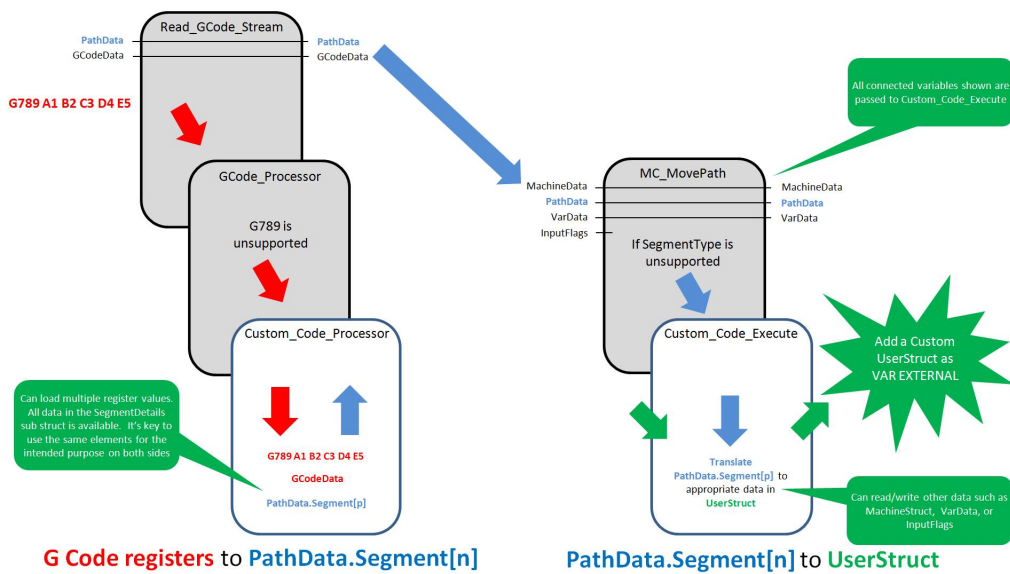
The [MC_MovePath](#) function block executes all motion and other sequence related activities. It is typically executed in a relatively fast task (~4 to 20 mSec), although this is only necessary if discrete input and output data must be read and set in a critical time period. For applications which require smooth motion with many short segments, more than one motion instruction can be loaded into the firmware motion buffer during one IEC application task scan. See the [MachineStruct](#) for details on configuring this feature.



Adding Custom G and M Codes

G and M codes not recognized by the G-Code solution from Yaskawa can be added to special function blocks in the User project without modification to the Group Toolbox.

Overview



The overview graphic shows two halves of the G-Code solution divided into left (Processing) and right (Motion.) The colors Red, Blue and Green depict data flow from G or M Codes to the MotionWorks IEC user application. There are two special function blocks required to customize the solution. [Custom_Code_Processor](#) and [Custom_Code_Execute](#). The help for these blocks contains detailed instructions for customization. A key point of this customization technique is the ability to add any data structures necessary to the Custom_Code_Execute function block as VAR EXTERNAL, which gives the solution the ability to link G-Code instructions and parameters to application specific data which is unique to the equipment. See the examples included for each of the Custom function blocks.

Customizable Features

- Read / Write values into the MachineStruct.
- Read / Write G-Code 'VarData.'
- Reference the InputFlags.
- Read / Write values into a user defined data structure.
- Wait for specific conditions to be met

Non Customizable Features

The customization solution described here was not intended to allow the user to add standard G-Codes involving motion which are not already supported in the Toolbox. Contact Yaskawa Motion Application Engineering to discuss your application.



Overriding G and M Codes

Overview

G-Codes supported by the Group Toolbox can be configured to pass execution of the command to user customizable code rather than executing code prepared by Yaskawa for a particular feature. A very common override is the T command for a tool changer. The build in T command simply assigns the new active tool number. If the machine requires the operation of a tool changer, this motion logic must be created by the OEM.

Overrides are declared by editing the [OverrideList](#) function block. Once the Overrides are declared, the process of handling overrides is nearly identical to the process of adding custom G and M codes. Two special function blocks are required to customize the solution. [Custom_Code_Processor](#) and [Custom_Code_Execute](#). The help for these blocks contains detailed instructions for customization. A key point of this customization technique is the ability to add any data structures necessary to the Custom_Code_Execute function block as VAR EXTERNAL, which gives the solution the ability to link G-Code instructions and parameters to application specific data which is unique to the equipment. See the examples included for each of the Custom function blocks.

Customizable Features

- Read / Write values into the MachineStruct.
- Read / Write G-Code 'VarData.'
- Reference the InputFlags.
- Read / Write values into a user defined data structure.
- Wait for specific conditions to be met

Non Customizable Features

The customization solution described here was not intended to allow the user to add standard G-Codes involving motion which are not already supported in the Toolbox. Contact Yaskawa Motion Application Engineering to discuss your application.

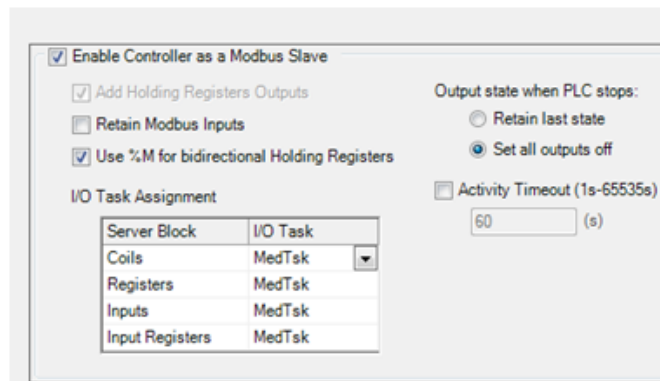


Getting Started with Pendant_Driver

Configure the MPiec Controller as a Modbus TCP server

1) Go online with the Hardware Configuration and configure the MPiec controller as a Modbus slave (server). The pendant is the Modbus master (client).

- a) Check the box to use %M for bidirectional Holding Registers.
- b) Select "Set all outputs off" when PLC stops. This is required for the safe operation of groups.
- c) IO Task Assignment - Set the four server blocks to the task in which the pendant driver function block will run.



2) Save the configuration and cycle power on the MPiec controller.

3) Four Modbus groups will automatically be created in the Global variables worksheet upon saving. Create variables under the FC16 and FC04 groups with the data types and addresses as shown in the figure below. The variable created in the FC16 group transmits data FROM the pendant TO the MPiec controller. The variable created in the FC04 group sends data FROM the MPiec TO the pendant.

Note: The FS100 pendant with the Indusoft runtime uses a 7500 register offset from the base address for Holding Registers and Input Registers. This is to minimize the chances that MotionWorks IEC projects using Modbus TCP for other functionality in the 4x and 3x memory areas will not conflict with the pendant.

Modbus FC#05 Qty: 512 Coils, Address Range: %IB73728 - %IB73791			
Modbus FC#03,06,16 Qty: 10000 Registers, Address Range: %MB3.483328 - %MB3.503327			
FC16Array	HoldingRegisters_ByteArray	VAR_GLOBAL	%MB3.498328
Modbus FC#02 Qty: 512 Inputs, Address Range: %QB73728 - %QB73791			
Modbus FC#04 Qty: 10000 Input Registers, Address Range: %QB73792 - %QB93791			
FC04Array	InputRegisters_ByteArray	VAR_GLOBAL	%QB88792
Pendant Group			
UserAppData	UserApplicationData	VAR_GLOBAL	

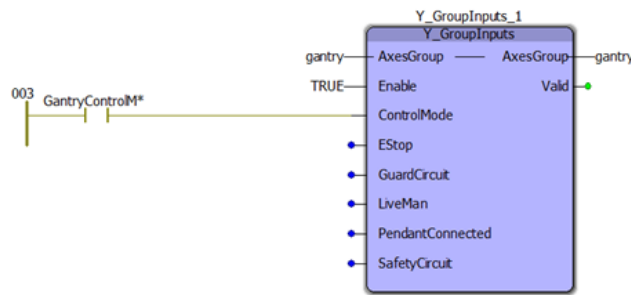
4) Add a variable of type UserApplicationData in the Global variables worksheet. This variable does not require an Address, but set the 'retained' checkbox.

Pendant Group			
UserAppData	UserApplicationData	VAR_GLOBAL	

[UserApplicationData](#) can store 20 teach point lists with 50 points in each job, 24 part frames, 24 tools etc.

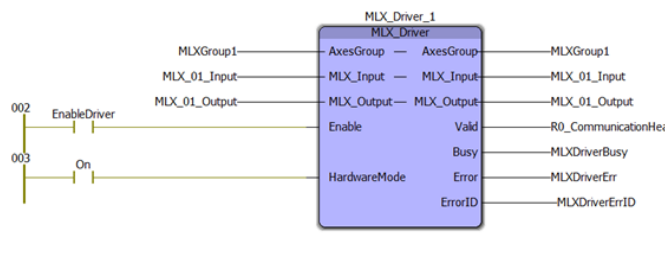
5) Add the Group_Toolbox to the project.

6) When using a pendant to control a Mechatrolink group add the Y_GroupInputs function block from the PLCopenPart4 library. Y_GroupInputs must be enabled and running for the Group to function.

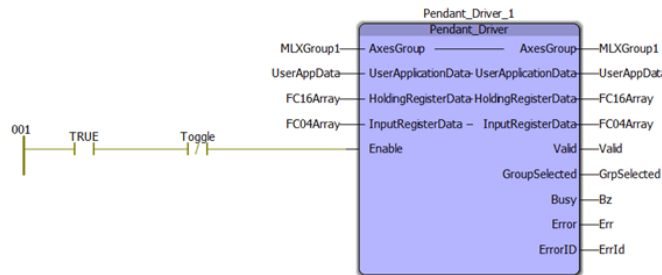


The Y_GroupInputs block monitors machine safety flags like E-Stop, pendant liveman, guard circuit, safety circuit etc. Yaskawa recommends that the user supplies these externally managed inputs to the function block such that these flags can be referenced by various motion function blocks. The Y_GroupInputs function block does not play a part in satisfying the safety requirements of the machine or group. If external safety devices are not used, the inputs can be left unconnected.

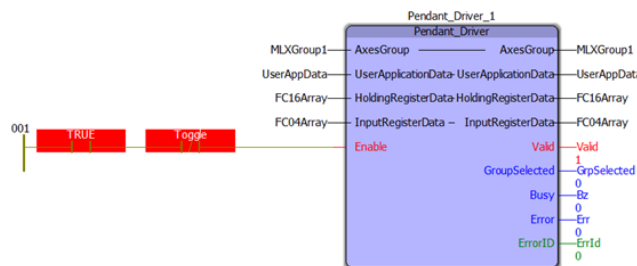
When using the pendant to control a remotely hosted group (MLX, DX, YRC, UC), add the MLX_Driver or MS_Driver function block (from the relevant user library, see the PLCopenPlus function block help manual for details.



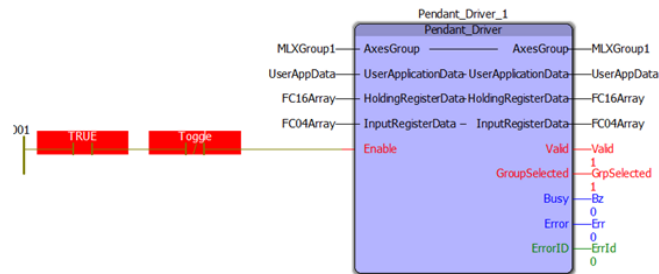
7) Add the [Pendant_Driver](#) function block to the main project. A 50 mSec CYCLIC task is recommended. Connect the variables described in Steps 3 and 4.



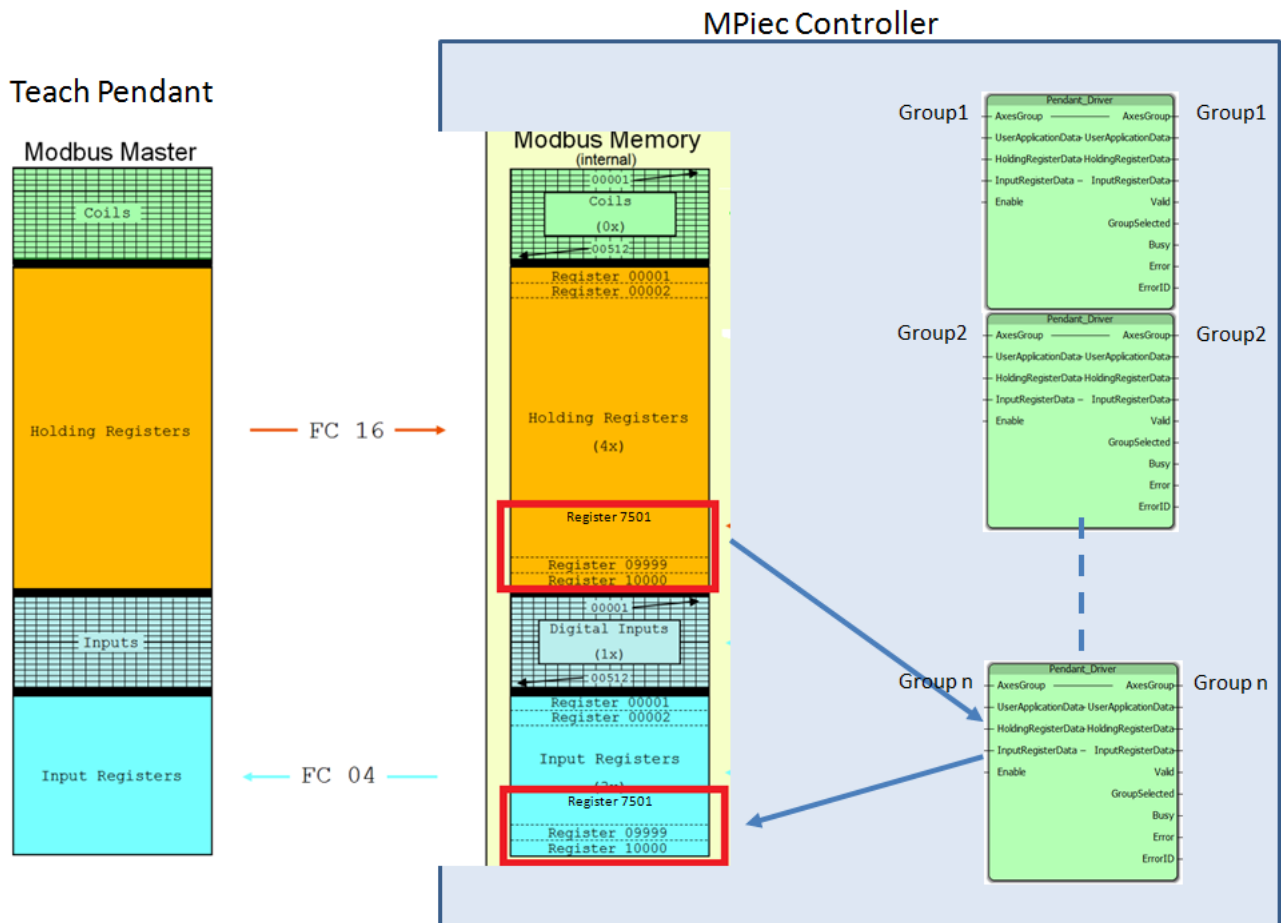
8) When the [Pendant_Driver](#) block is enabled, it checks for a heartbeat signal from the pendant. If the heartbeat is updating, the Valid output is set to TRUE.



9) The second check is to confirm the Group name entered on the pendant matches the Group name of AxesGroup connected to the function block. If the Group name entered on the pendant matches the AxesGroup's configured name, the GroupSelected output will be set TRUE. This is the indication that the pendant is ready to operate the Group..



10) To control another Group, use another instance of the Pendant_Driver. Change only the AxesGroup VAR_IN_OUT, the other VAR_IN_OUT data must remain the same. The figure below illustrates how multiple Groups can be controlled from the pendant.





Getting Started with Secondary Axes

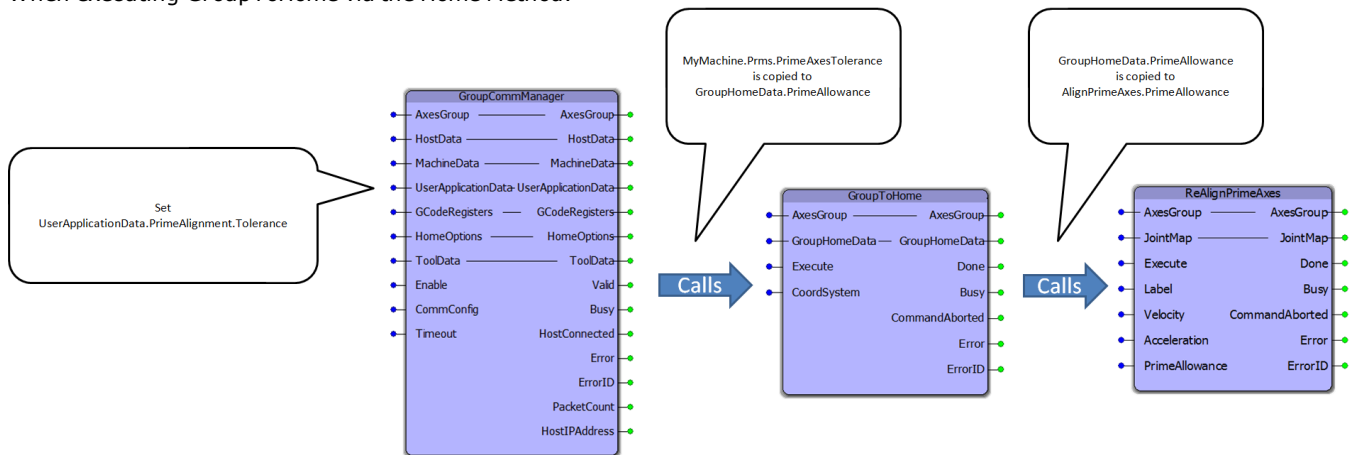
If the group mechanism has a joint which is operated by two or more servos, they must have the exact same commanded position before the group can be enabled using MC_GroupEnable.

Group Toolbox includes [ReAlignPrimeAxes](#) and [GroupReAlignPrimeAxes](#) function blocks which make enabling a group with secondary axes easier. Functions such as [GroupCommManager](#), [Pendant_Driver](#) and [GroupToHome](#) execute those function blocks if necessary to re-align the axes prior to executing MC_GroupEnable. This will prevent error 8966 from occurring.

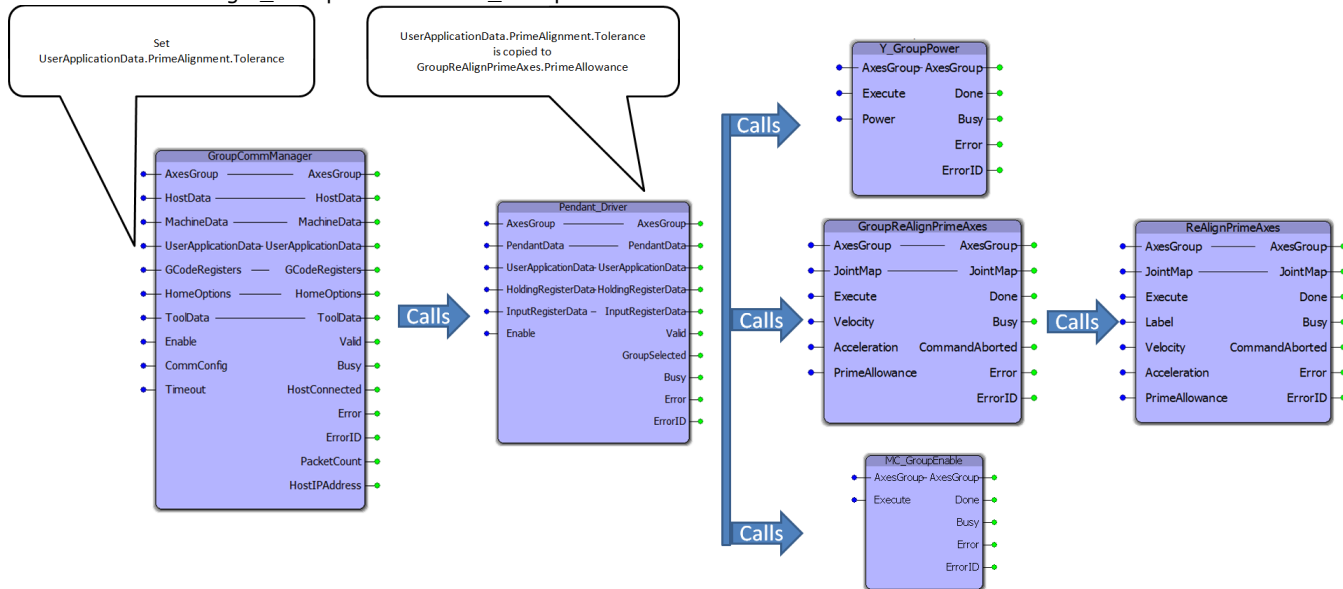
For the alignment procedure to complete successfully, a Prime Axes Allowance must be specified. There are several ways in which the functions in Group Toolbox can be applied; the following graphics explain each usage scenario and indicate the required initialization.

Using GroupCommManager and a PC application / GroupComm.DLL (Compass solution)

When executing GroupToHome via the Home Method.

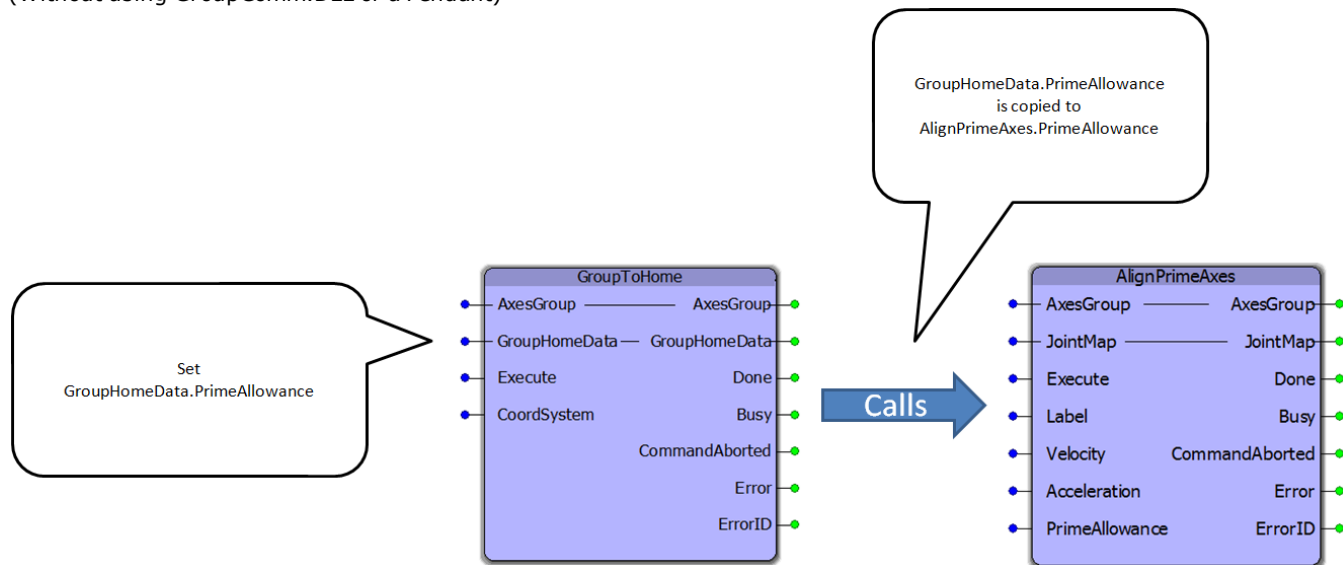


Same when executing Y_GroupPower and MC_GroupEnable via the ServoPowerAll Method.



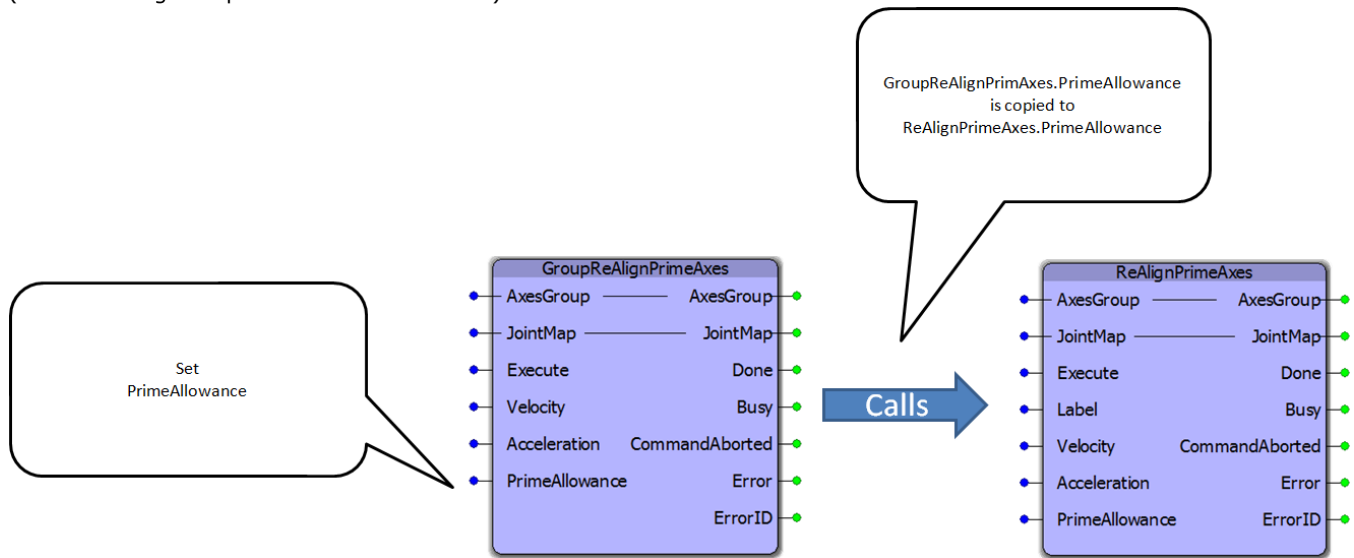
When using the GroupToHome function block in the MotionWorks IEC project directly

(Without using GroupComm.DLL or a Pendant)



When using the GroupReAlignPrimeAxes function block in the MotionWorks IEC project directly

(Without using GroupComm.DLL or a Pendant)



You must provide a value which represents the maximum safe expected motion the mechanism can tolerate. The presumption is that the mechanism has previously been physically aligned and all motors operating the joint have been set to the same position using MC_SetPosition. GroupReAlignPrimeAxes is only intended to perform small realignment motion to account for small position changes that may occur when the system is powered off.

For example, a mechanism has two motors without brakes operating the Z axis. The mechanical rigidity of the system is not perfect which allows both sides of the axis to settle due to gravity by differing amounts. In this example, assume that the axes fell out of alignment by approximately 6 user units. In this situation, setting User-ApplicationData.PrimeAlignment.Tolerance:=LREAL# 10.0 will permit the GroupReAlignPrimeAxes function block to move the Z prime axis to the same commanded position as the main Z axis. If the initial commanded positions differ by more than 10 user units, error 10138 will be generated. Determine why the axes are so far out of alignment.

It is unsafe to set the PrimeAlignment variable to an arbitrarily large value to get around the ErrorID 10138. If the commanded positions of each motor differ by an unexpected, significant amount (such as during commissioning or after an absolute encoder battery failure), mechanical damage could occur while the axes are attempting to realign themselves to a mechanically impossible position. Instead, check the system, re-align the servos and reset the Positions if necessary.

If the group does not contain joints with prime axes, the PrimeAxesTolerance parameter will never be referenced, and no pre-alignment motion will be attempted.



G-Code Emulation Modes

The default is no emulation mode selected (MyMachine.Emulation:=GTB_Emulation#na). Only a single mode can be selected. Emulation modes alter the behavior of G-Code solution in the following ways:

Mode1

Feature	Behavior
E Register	Interpret E register as positive values for acceleration and negative values for deceleration in units/sec ² . Values are copied to MachineData.Prms.Acceleration and MachineData.Prms.Deceleration.
F Register	Velocity is units/sec rather than units/min.

Mode2

Feature	Behavior
F	Feedrate maximums for XY, Z, and rotational motion can be specified separately using axes letters as the parameters. For example: F 300 XY F 50 Z F 360 A Feedrates will be checked such that the limit for a each axis is not exceeded. For example, if the Z feedrate has been set at 50 units/min, and a command such as G1 X40 Y200 Z12 F600 is executed, the overall velocity of the command may be reduced to less than 600 if the Z component of the move would require it to exceed 50.
G0	Rapid moves unaffected by Feedrate Override.
G5	To display a dialog message on Compass. This command must be added as an override. See the G5 Dialog Example .
G92	If executed by itself, clears all MachineData.LocalZeroOffset[] to zero. If Axes registers are provided, sets the new LocalZeroOffset for the specified axes.
G53	If executed with no axes specified, the Z axis will be moved to MachineData.Origin[3] .
H	Home Preset commands can be used with G1. For example G1 H#3 will move the machine to the pre determined location identified as Home #3. Position data is stored in MyMachine.Home.Preset []. See the Home Preset panel in Compass.
L10	Looping command. See the L10 Looping examples.
L30 / L31	Commands are recognized and support Tangent mode. L30: Enables tangent mode just before the next G1, G2, or G3.

	<p>L31: Disables tangent Mode.</p> <p>L30.1: Sets parameters A,P,Z as follows but does not enable tangent mode.</p> <p>A register: Maximum Angle between segments for tangent mode to proceed without auto inserting an "up, rotate and down" sequence.</p> <p>P register: Z position when tangent mode is actively cutting.</p> <p>Z register: Z Position when the Z axis must be raised to rotate the knife to align to the next tangent operation.</p> <p>See Tangent Mode.</p>
M3 / M4 / M5	<p>Additional spindles can be controlled using the parameter suffix M3.1 or M3.2 and likewise for the spindle reverse and stop command.</p>



Group Toolbox Revision History

Current Version:

2023-09-26 v375 released. fw 3.7.4 or higher recommended

New Features, Improvements:

- 1) LaserControl FB - Add Compensation delay feature. DCR 6886.
- 2) Make rotary moves default to shortest path if axis configured as rotary in Hardware Configuration. DCR 6895.
- 3) G-Code: Consolidate references to MinimumPosition tolerances instead of hard coded assumptions. DCR 7292.
- 4) IsMotionSegment FB - Modified to pass new MotionFlags WORD. DCR 7297.
- 5) MC_MovePath FB - Dynamically set Group Parameter 2203 'BufferMin' to reduce CPU usage when MachineData.Prm-s.MaxSegmentsPerScan is greater than one. DCR 7299.
- 6) MotionTimeFull FB - Improvement for Lookahead starvation conditions. DCR 7305.
- 7) Lookahead FB - Move code that switches small arcs to straight lines earlier in the algorithm. DCR 7336.
- 8) G-Code: When Tangent Mode is enabled, G53 commands must not be considered as tangent moves. DCR 7337.
- 9) G-Code: Set ExactStopCheck flag on previous segment when there is no delta position. DCR 7353.
- 10) Add Advanced Lookahead Feature. DCR

Bug Fixes

- 1) SyncGroupToGroup FB - Disabling function block did not disable the CustomGroup. DCR 4972.
- 2) ConvertMotionPointsCS FB - Improvement to use the complete LocalZeroOffset data. This affects work offsets with rotational components. DCR 6853.
- 3) Sequence Lookahead - SyncSegments must wait to advance if ExactStopCheck is TRUE. The following commands types were affected: 20: Set Fan Speed, 21,27: Set temperature, 32: Set Parameter, 34: Message. DCR 6901.
- 4) G92 command can report Error 8999. DCR 6902. This is related to the previous bug DCR 6901.
- 5) GridMeasurement FB - Error did not reset when Execute is set low. DCR 6946.
- 6) MotionTimeFull FB can cause Divide By Zero PLC Error based on Compass connection timing. DCR 6953.
- 7) G0 acceleration issue. DCR 6974, related to improvement DCR 7292.
- 8) Lookahead can produce instantaneous decelerations to zero. DCR 7044.
- 9) Lookahead can produce Error 4658 due to not calculating a reduced velocity. DCR 7053.
- 10) Pendant_Driver FB - Jog increment feature did not work for auxiliary axes / extruders. DCR 7080.
- 11) No rotary motion when first commanded, such as G1 C45, need to send command twice. DCR 7161.
- 12) MC_MovePath FB - Added range check for position delta calculations to better determine Cartesian, Rotational and Auxiliary motion. DCR 7012.
- 13) Constant Accel Feedrate Override (Feature.X0) can result in abnormally super slow velocities with small Cartesian, large rotational moves. DCR 7283.
- 14) GroupControl FB - should not disable group when executing AlarmClear. DCR 7288.
- 15) CalcSegmentDistance FB - didn't handle RotationalOnly moves. DCR 7293.
- 16) Lookahead FB - Should not process segments that are RotationalOnly moves. DCR 7295.

- 17) GCodeProcessor FB - AxesFlagsPending can get confused about emulation mode2 G53. DCR 7335.
- 18) DetermineFeedrate FB - Did not allow for Feedrate Overrides greater than 100%. DCR 7338.
- 19) IsMotionSegment FB - Did not detect 'Z only' instructions in CartesianPath mode. DCR 7343.
- 20) MovePath FB - Could generate Error 8999 while executing G92 E0 when printing. DCR 7345.
- 21) NOTE! This fix is in the File RW Toolbox, but applies to G-Code. ReadValue FB - (G Code Emulation mode 2 issue) F60Z is not accepted, no error. Did not see the single instruction as the last on the line. DCR 7118.

Previous Versions:

2022-09-14 v374b released. fw 3.7.4 or higher recommended

New Features, Improvements:

- 1) MC_MovePath performance: Prevent 61713 Error when TransitionMode 4 set on back to back segments both shorter than the TransitionParameter value. DCR 6425.
- 2) MC_MovePath performance: Transition mode changes to eliminate 9001 errors. DCR 6467.)
- 3) MC_MovePath performance: Path performance - prevent 8971 / 9030 error while not affecting path shape. DCR 6469.
- 4) G-Code: Add Feedrate override as Feature.X5 option for G0 commands. DCR 6584.
- 5) G-Code: Add the ability to set individual extruder temperature zones via M104 / M109. DCR 6585.
- 6) G-Code: L10 and M98 commands are preprocessed by Compass when using Compass 1.4.0 or higher. DCR 6590.
- 7) G-Code: Use MyMachine.G92RotationInhibit to disable support for G92 for PCS rotation. DCR 6642.
- 8) G-Code: Support modulo axis positioning for rotary axes via G-Code. DCR 6675.
- 9) G-Code: Add support for simultaneous extruder positioning. DCR 5879.
- 10) Read_GCode_Stream / StreamParser FB performance increase for max data rate of 25KBits/Sec. DCR 6647.
- 11) Improve MyPath.Buffer.UnderRunWarning detection. DCR 6652.
- 12) LaserControl FB added. DCR 6636.

Changes:

- 1) MC_MovePath: Change AuxiliaryOnly moves to UseNSpaceScalers instead of UseRotationalScalers. DCR 6637.
- 2) Change PathData.StreamStatus.TCPVelocity (and individuals) to report units / minute DCR 6664.

Bug Fixes

- 1) GroupToHome FB did not home rotational axes if no Cartesian motion. (because of SCR 12883) DCR 3698.
- 2) GroupToHome FB - added ErrorIDs 11084 and 11085 to report out of range data in HomeDataStuct. DCR 3699.
- 3) G-Code: States in (GCodeData.ProcessingMode) like G91 must be retained when switching to a subroutine and back. DCR 6408.
- 4) G-Code: Lookahead was using wrong segment length to back calculate ReducedVelocity requirement. DCR 6481.
- 5) MC_MovePath FB - Fixed bad check in Rotational Motion indicator. DCR 6433.
- 6) MC_MovePath FB - CheckAccelLimit passed wrong input for Rotational or AuxiliaryOnly segments. DCR 6460.
- 7) G-Code: MyMachine.Feature.X0 - Don't throw error 11075 if SubDivideDone. DCR 6601.
- 8) GetGroupUnits FB - Done output gets stuck on if R_TRIG the input. DCR 6644.
- 9) Feed Hold takes a long time to decelerate in some cases - seems to use Cartesian deceleration applied to Rotational deceleration. DCR 6662.
- 10) MC_MovePath FB - Can erroneously get Loop Error Alarm (10136) during M98 subroutine operation. DCR 6666.
- 11) G-Code: M109 Set Temp and Wait for individual zones waited for all zones instead of specified zone. DCR 6702.
- 12) G1 Fxxx (without motion registers) did not store feedrate correctly. DCR 6616.

2021-12-14 v373d released. fw 3.7.3 or higher recommended

New Features, Improvements:

- 1) Support GCodes to display messages on screen (M117, M118, msg) DCR 3180.
- 2) Add ErrorID that variables and tool compensation cannot be used at the same time. DCR 3259.
- 3) M66 command is retaining input conditions from previous path executions. DCR 5195.
- 4) Add OEM option for jog speed functionality in Jog Increment mode. DCR 5599.
- 5) Allow option for MoveOptions.VelocityUnit Rotational to take precedence over Translational. DCR 5736.
- 6) DetermineFeedrate needs better initialization of IrFeedrate for Z only moves. DCR 5814.
- 7) Support relative (G91) mode for Arcs during Feature.X0 subinterpolation for feedrate override. DCR 5834.
- 8) Auto inserted Extruder Change instruction needs explicit setting for G0. DCR 5881.
- 9) Custom / Override feature needs to suspend processing - specifically to re-read group position after potential group motion. DCR 5899.
- 10) Toolbox M98 Subroutine support for Compass / Streaming implementation. DCR 5904.
- 11) Error 8971 during execution of circular segment + z motion (helix). DCR 5941.
- 12) Suppress 9001 Errors when 3D Printing, Using Transition Mode 4 (Lookahead) high acceleration configuration. DCR 5953.
- 13) Suppress 8971 Errors when radius is very small, segment time very small. DCR 6057.
- 14) Improve clearing of MyMachine.Control.Cancel. DCR 6290.
- 15) Set GCode_Processor and MC_MovePath local vars "BaseToolOffset" as Retain - helps to resume prior setup after power down. DCR 6257.

Changes:

- 1) Require Valid output from Custom_Code_Processor FB. DCR 5982.
- 2) IF condition cannot be used with variable mode "B" MyMachine.Feature.X3=TRUE, must throw error. DCR 5972.
- 3) Add ErrorID indicating G-Code variables and tool compensation cannot be used at the same time. DCR 5958.

Bug Fixes:

- 1) Block Sequence Error 11069 will occur if segment requires subdivision and has other commands previously processed on the same line. DCR 5748.
- 2) GroupReAlignPrimeAxes "FBAbort" var not initialized. DCR 5782.
- 3) DetermineFeedrate FB was checking the wrong AxesFlag for Extruder moves. DCR 5815.
- 4) Use MachineData.Prms.ZAcceleration for Z only moves. DCR 5822.
- 5) Change CheckAccelLimit FB to utilize the min and max better. DCR 5823.
- 6) Prevent occasional Lookahead hangups during feedrate override motion time limiting. DCR 5859.
- 7) Extruder tool change no previous G-Code command can cause incorrect CmdPos in LoadPositions. DCR 5862.
- 8) Improve Legacy support for GroupComm (pre PLCi) for Yaskawa CNC. DCR 5908.
- 9) Avoid processing lockup / hang when 3D printing with Lookahead and constant accel feedrate override enabled. DCR 5954.
- 10) MC_MovePath must initialize MotionPoints[PrevMCoord].SelectedTool for 3D printers. DCR 6001.
- 11) First Cycle Start after Single step is engaged did nothing. DCR 6111.
- 12) DetermineBlockSequence FB not detecting spindle as one of several commands on a line. DCR 6116.
- 13) Z Tool height may move to incorrect position if restart on a (overridden) T command. DCR 6323
- 14) G92 with no arguments must clear the current BaseToolOffset. DCR 6287.
- 15) Improve MC_MovePath CommandAborted timing. DCR 6238.
- 16) G04 dwell incorrect units when using X parameter. DCR 6212.

2020-07-19 v372 released. fw 3.7.1 or higher recommended

New Features, Improvements:

- 1) G-Code - GCode_Procesor, Provide constant acceleration for feedrate overrides with a responsiveness of approximately 500 msec. DCR 5328.
- 2) SyncGroupToGroup / Pendant_Driver FBs - Report SecondaryGroup Power/ Enable status when secondary group is configured in PendantData for better reflection of true machine state at the GUI level. DCR 6467.
- 3) Add Feature select bits as DWORD to MachineStruct. This allows for turning on/off various things such as constant accel feedrate override. DCR 5178.
- 4) G-Code - Added support for G73 and G83 - chip break drilling cycle. DCR 5398.
- 5) G-Code - Added support for G80 - Cancel modal motion. DCR 1096.
- 6) Add provision to change an Overridden G-Code command's PostGCode STRING upon completion of the override. DCR 5568.

Changes:

- 1) G-Code - G0 rapid, if "F" is supplied, use the feedrate provided instead of MachineData.MaxDirectVelocity. DCR 5029.
- 2) G-Code - Change M104/M109 to not change the selected extruder. DCR 5111.
- 3) CustomTCPInWIndow FB - Change 'Tolerance' from REAL to LREAL. DCR 5116.
- 4) Tool Length compensation (G43) not to be applied until next Z command is included in G-Code. DCR 5698.
- 5) L30 Tangent - Segregate use of PathData.Colinearity and MachineData.Tangent.MaxAngle. DCR 5584.
- 6) Don't set Spindle directions to 0 (off) when in MDI (M30 is auto appended, which shuts off spindles). DCR 5531.

Bug Fixes:

- 1) Pendant_Driver jogging did not use UserData.Config.JogSpeed.Orientation when jogging Rx, Ry, Rz. DCR 5408.
- 2) Lookahead FB - Change var 'ContiguousMotionSegments' from UINT to UDINT to prevent failure when there are more than 65536 contiguous segments. DCR 5395.
- 3) Lookahead FB - Fix treatment of long segments when not using application layer feedrate override. DCR 5327.
- 4) Custom_Code_Execute FB -added an iActive bit so that it can be busy with Enable false.
- 5) G-Code - G02 Arc went counterclockwise when an arc greater than 180 degrees was specified. DCR 5268.
- 6) G-Code - Use of registers A B C is now OK when used with a G92 command even if not a -ER controller. DCR 5264.
- (7) G-Code - Motion (G1 / G2 / G3) intermixed with custom or overrides was not taking the user setting 'ExactStopCheck=k=FALSE' to provide custom codes synced with path motion. DCR 5256.
- 8) GroupStatus FB - Changed to NOT report ServoPack warnings as alarms. DCR 5224.
- 9) Compass Terminal MDI - G92 E0 followed by a G1 E move gives ExtruderOffsetInvalid. DCR 5211.
- 10) G-Code - Tangent disable G79 and L31 get stuck (tangent axis stuck in Stopping state). This was actually due to a firmware bug introduced in v3.7.0. A workaround solution was applied in this Toolbox as well. DCR 5131.
- 11) MC_MovePath - Was throwing Error 8999 after changing Y_GroupSetFrameOffset to use Queued mode. Fixed in PLCopenPart4 library. DCR 5102.
- 12) G-Code Variables did not work with S Register. DCR 5011.
- 13) G-Code - Home Presets (G0 H3) not executing full sequence specification. DCR 4980.
- 14) ReadJointMap FB - Sometimes reported 10126 (no data) ErrorID after successfully reading XML file. DCR 4216.
- 15) Lookahead not considering arcs which would exceed centripetal acceleration. DCR 5709.
- 16) When changing from MCS to PCS or vice versa, update PreviousMotion position to be representative of new Coordinate System. DCR 5704.
- 17) DetermineFeedrate RotationalOnly move feedrate is not always correct. DCR 5607.
- 18) G92 E0 commands following a Tool change command (with tool offsets) sets local zero offset on non-specified axis. DCR 5602.
- 19) Compass may not handle conditional M30 before or at the end of file. DCR 5536.

2020-11-30 v371 released. fw 3.7.1 or higher recommended

New Features, Improvements:

- 1) G-Code - Add support for M82 / M83 command - Extruder Only Absolute / Relative Mode. DCR 3254.
- 2) Add Separate jog speed Z vs. XY as part of Pendant_Driver FB / Compass solution. DCR 3808.
- 3) G-Code - Add support for G1 H3 (Home Presets) under Mode 2 emulation. DCR 4020.
- 4) Pendant_Driver / Compass - allow jog keys to jog more than one axis simultaneously. DCR 4067.
- 5) G-Code - Added support for 'Set Chamber / Enclosure Temperature' codes M141 / M191. DCR 4153.
- 6) G-Code Processor - Implement Lookahead algorithm that integrates with StreamParser function block. DCR 4361.
- 7) MachineType = 3D printer can use separate Z velocities when MachineData.Emulation is set to Mode2 emulation. DCR 4388.
- 8) Pendant_Driver / Compass solution - added option to suppress soft overtravel alarms when jogging. DCR 4486.
- 9) G-Code - Added support for multiple spindles via M143 / M144 / M145 / M153 / M154 / M155. M3.1/ M5.1 DCR 4633.
- 10) GroupReAlignPrime Axes FB - Increased support for Multiple prime axes to 6. DCR 4701.
- 11) PendantData structure - Added status bits to indicate software limits exceeded. DCR 4721.

Changes:

- 12) Custom_Code_Processor stub FB - Added PrmSuffix as VAR_INPUT. Existing implementations of this function block must add the new VAR_INPUT to the main project. DCR 4230.

Bug Fixes:

- 13) G-Code spindle - Make spindle speed go to zero when Feed Hold, system now waits for spindle to resume previous speed before continuing path. DCR 4277.
- 14) G-Code - Local zero offsets were not accounting for PCS offsets when using G92 command. DCR 4366.
- 15) G-Code - T commands could cause CPU exception (invalid array index) if Tool number is out of bounds of the ToolData.Tool array. DCR 4374.
- 16) CheckColinearity FB - was errantly assuming Colinearity if the previous segment had ExactStopCheck flag set. DCR 4512.
- 17) G-Code spindle - If Spindle was on before a Feed Hold, and part canceled during hold, previous spindle speed must be cleared. DCR 4513.
- 18) MC_MovePath FB - Velocity & Accel were not limit checked properly - need to consider the feedrate override scaler in effect. DCR 4518.
- 19) G92 command was not working properly with G-Code Variables. DCR 4666.
- 20) MC_MovePath - G92 / set PCS / Segment type 17 was not working with MotomanSync groups. DCR 4746.
- 21) G-Code - Tool height offsets compounded if commands issued with no Z element. DCR 4781.
- 22) Calc3DCenter had divide by zero issues. DCR 4787.
- 23) Tool Height offsets were not being properly displayed, retained. DCR 4853.
- 24) Compass - Extruder positions were not updated until MC_MovePath has been executed. DCR 4681.
- 25) StramPaser FB - Initialize DataBuffer.BufferRead on FB startup to prevent G-Code lines with multiple bad commands from causing additional errors. DCR 4925.
- 26) G0 commands now set to explicitly stop at the end of each segment, there were cases where blending G0 commands may have been applied.

2020-05-03 v370 released. fw 3.7.0 or higher recommended

New Features, Improvements:

- 1) GridLookup FB - New function block added. DCR 1729.
- 2) GroupToHome - More efficient / faster by using previously populated JointMap as VAR_IN_OUT. DCR 1914.
- 3) GroupComm support for PLCi communication mode for Compass Software. DCR 2977.
- 4) G-Code - Support M190. (Wait for bed temperature to reach target temp.) DCR 2290.

- 5) PendantDriver did not respond to servo power toggle unless in manual mode. Now OK in Auto or manual when used with Compass. DCR 2372.
- 6) PendantDriver / TeachPendant - Added ability for user customizable acceleration rates for jogging. DCR 2472.
- 7) PlaneMeasurement/GridMeasurement - Change the point from where the offsets are normalized. DCR 2692.
- 8) G-Code - Added recognition of % character for start of G-Code file comment. DCR 2715.
- 9) MC_MovePath - Limit check velocities and accelerations differently when a MotomanSync robot is used. DCR 2739.
- 10) 3D-Printing - Extruder Jog now possible when set as an auxiliary axis in the group config. DCR 2901.
- 11) PlaneMeasurement/GridMeasurement - Added Analog measurement method. DCR 2960.
- 12) GridMeasurement - Changed path from "scanning" to "zig zag" to save time. DCR 2976.
- 13) PendantDriver - Now supports incremental jogging. DCR 3063.
- 14) HomeToSurface - New function block created. DCR 3086.
- 15) MC_MovePath - Allow infinite repeat mode when streaming. DCR 3104.
- 16) G-Code - Support separate feedrates for XY and Z. DCR 3181.
- 17) G-Code - Support Local Zeros (LZ) via G92. DCR 3186.
- 18) G-Code - CalcToolComp (G41, G42) reduction in Tolerance strictness. DCR 3256.
- 19) G-Code - Add support for more than one spindle in MachineStruct. DCR 3261.
- 20) G-Code - Add support for configuration of G-Code overrides (Macros). DCR 3262.
- 21) GridControl - Implement new function block to consolidate GridLookup and SyncGroupToGroup. DCR 3273
- 22) Pendant_Driver - Turn servos off when heartbeat with host is lost. DCR 3410.
- 23) MC_MovePath - Auto set Colinearity parameter 2111 to 10.0 degrees if left at firmware default. DCR 3414.
- 24) GroupToHome - Improve logic to handle servos being shut off in the middle of homing. DCR 3455.
- 25) GroupToHome - Added support for CommandAborted output. DCR 3460.
- 26) G-Code - Add support for G43, G44, G49 for Tool Length Offset. DCR 3478.
- 27) Pendant_Driver - If no name is specified for Tool File, User Frame or Teach Point List file, use a default name. DCR 3484
- 28) Pendant_Driver - (Compass) Reset Feed Hold when TriggerReset (Reset) is pressed. DCR 3500.
- 29) Pendant_Driver / GroupCommManager - Improve servo off state logic to ensure MC_GroupDisable succeeds (to successfully clear 8966 alarm). DCR 3507.
- 30) Improve G0 usage of DirectAccel / Decel. DCR 3573.
- 31) GroupReAlignPrimeAxes - Improve support for multi (3) prime joints. DCR 3585.
- 32) G-Code - Repurpose items in PathStatusStruct to provide GUI with byte offsets for Errors from Processing and MC_MovePath. DCR 3586.
- 33) G-Code - Provided way to run G-Code commands (coming from Compass buttons) while in manual mode. DCR 3587.
- 34) ReadJointMap FB - Increase performance to execute in very fast task w/o causing watchdog. DCR 3631.
- 35) G-Code - Allow G92 offsets for all axes (Previously only E axis for 3D printer extruder allowed). DCR 3640.
- 36) G-Code - Added support for inclusion of additional G-Codes before and after a G-Code override is executed. DCR 3663.
- 37) G-Code - Added support for Tangent knife operation under emulation mode2. DCR 3668.
- 39) Add support for Mode2 B feedrate for rotational axes. DCR 3867.

Changes

- 40) G-Code - Support G53 as described by the G-Code standard. DCR 3260.
- 41) 3D Printing - Added Definition for number of configured temp sensors to PrinterStruct. DCR 3420.
- 42) Pendant_Driver / GroupCommManager - Clearing alarms from Compass should be allowed in Auto/Manual/MDI. DCR 3512.

- 43) G-Code - No longer initialize G-Code registers (GCodeData structure) when Read_GCode_Stream / Read_GCodeFile starts up or when ClearBuffer. DCR 3571.
- 44) Revert GetGroupDOF FB to rely on AxesGroup info for MotomanSync groups - (SCR 12547 is completed). DCR 3604.

Bug Fixes:

- 45) M30 failed to stop MC_MovePath from processing additional segments when there were motion segments in the firmware buffer. DCR 2807.
- 46) G-Code - Support multiple actionable G-Code segments on a single line. DCR 2489.
- 47) PlaneMeasurement: Rotational axes spin back to zero (often causing overspeed) when block is executed. DCR 1787.
- 48) MC_MovePath - Velocity spike if ExactStopCheck = FALSE for segment before StandStill segment. DCR 2124.
- 49) G-Code - Fixed multiple issues with G28. DCR 2260.
- 50) G-Code - MachineData.Printer.ActiveExtruder did not indicate which extruder is Active. DCR 2292.
- 51) G-Code - Machinedata.Spindle.RPM did not get updated every time the 'S' G-Code parameter was issued. DCR 2327.
- 52) ReadJointMap - Information was incorrect if more than one Mechatrolink group was configured. DCR 2500.
- 53) 3D Printing - The velocity should not change if a M104 or M106 command is executed in the middle of motion commands. DCR 2557.
- 54) G-Code - Fix Custom Code execution / path incrementation for scenario like [motion motion motion custom dwell other]. DCR 2688.
- 55) 3D Printing - Fix Extruder printer extruder offsets so as not to repeatedly add an offset for unspecified axes. DCR 2694.
- 56) MC_MovePath / LoadPositions FB - UnconfiguredExtruderError caught, but not preventing an array index out of bounds crash. DCR 2781.
- 57) MC_MovePath - Spindle had potential for divide by zero exception. DCR 2971.
- 58) MC_MovePath - Make Spindle commands M3 M4 M5 G97 sync with other motion. DCR 2981.
- 59) 3D Printing - M140 now synced with motion. DCR 3055.
- 60) MC_MovePath - InputFlagsRequired was not getting reset after abort/done/error. DCR 3095.
- 61) Pendant_Driver - ReadPartFrameFile function block only worked once. DCR 3228.
- 62) GroupStatus - Now EXITs if MC_ReadAxisError is ever Busy to avoid watchdog. DCR 3253.
- 63) G-Code - CalcToolComp not propagating Errors to user level. DCR 3258.
- 64) Read_GCode_Stream / Read_GCode_File - ErrorRow output was off by one depending on when Error occurred. DCR 3266.
- 65) G-Code - G2 and G3 commands switching between R and IJK mode not using IJK parameters. DCR 3304.
- 66) Pendant_Driver - Errors when setting a tool not being caught in error handling. DCR 3322.
- 67) MC_MovePath - Tangent Axis Pre-alignment Velocity was referencing Rotational Acceleration instead of Velocity. DCR 3569.
- 68) GroupControl FB / MC_GroupReset was unable to clear group alarms while AxesGroup.Status.Active = TRUE. DCR 3602.
- 69) 3D Printing / MC_MovePath - Fixed infinite velocity error 8978 when executing a Tool Change. DCR 3653.
- 70) MC_MovePath / MotomanSync - Motion sequencing froze when also using conveyor tracking. DCR 3665.
- 71) Support G00 / G01 with no axes as just a mode setting, no motion should occur. DCR 3832.

New Features, Improvements

- 1) Updated StreamStatus struct and GCodeComm DLL to structure version 20190127. **When using Group Toolbox v352 with GCodeComm.DLL, version 3.4.2.0 of the DLL is required.**
- 2) Updated MachineStruct to new structure version 0120190110. **When using Group Toolbox v352 with GroupComm.DLL, version 1.0.3.1 of the DLL is required.**
- 3) ExternalExtruderControl - New function block added for controlling an extruder mounted to a MotomanSync robot. DCR 2614.

- 4) Custom_Code_Processor and Custom_Code_Execute - New function blocks created for custom G & M Codes as user customizable FBs. DCR 2073.
- 5) Custom_Code_Execute - Provide mechanism for custom G & M codes to select execution synchronized with motion that does not stop. DCR 2355.
- 6) Support unique Transition modes for each segment rather than just a single setting from MachineStruct.Prms. DCR 2026.
- 7) GetGroupDOF - Updated FB to support auxiliary axes (support coming in MotionWorks IEC 3.6) DCR 2198.
- 8) G-Code - Improved validation for numeric parameters, and allow special cases like #var assignments to pass. DCR 2243.
- 9) G-Code - Add Special SegmentType for G10 - to change Work Coordinate System Offset. DCR 2247.
- 10) PendantDriver - Changed to execute MC_GroupDisable when the PowerStateReq request is FALSE. DCR 2255.
- 12) G-Code - M2 / M30 End of Program commands - If more than one, first one encountered in program flow prevails. DCR 2268.
- 12) MC_MovePath - Added SegmentType to indicate the end of the sequence. DCR 2271. Related to DCR 2268.
- 13) MC_MovePath - Improve determination of positions to apply that were not specified in the Segment data. DCR 2273.
- 14) G-Code - Refactor Extruder position management. DCR 2381.
- 15) G-Code - Add support for M0 and M1. DCR 2331.
- 16) PathStruct.InUse flag - Improvement for CNCWorks Cancel behavior. DCR 2341.
- 17) G-Code - Added support for Block Skip Mode. DCR 2181.
- 18) Support 6 axis 3D printer plus three extruders. DCR 2113.
- 19) MachineStruct to GroupComm DLL - changed MachineData.Torque[] to report all axes torque including prime axes. (Change MCS labeling.) DCR 2448.
- 20) StreamStatus to GCodeComm DLL - Added ToGoDistance fields to the StreamStatus structure for use by GUI applications. DCR 2450. Requires GroupComm DLL version 1.0.3.1.
- 21) StreamStatus to GCodeComm DLL - Added G and M code Active code status for each G and M code as published in the CNC Programmer Handbook.
- 22) GetTorques - New function block added to read the actual torque for all axes in a group including secondary axes. DCR 2494.

Changes

- 23) Pendant_Driver - MC_GroupSetOverride should only be effective in auto mode. DCR 2217.
- 24) Use MachineData.Prms.MaxDirectVelocity when G0 move specified. DCR 2225.
- 25) Read_GCode_Stream no longer sets the Error / ErrorID outputs for unrecognized commands, but still requires 'ClearBuffer' to acknowledge and reset. This improvement means the connection is no longer shut down for bad commands. DCR 2934.
- 26) MachineStruct.Printer.Extruder[] - Changed MinTargetTemp to TempTolerance so M109 can support waiting for temperature decrease.

Bug Fixes

- 27) Variable assignment to registers should be persistent for future blocks / Segments until a constant is given again. DCR 1469.
- 28) G10 should only update offsets in MachineStruct, G54 ~ 59 actually applies specified offsets. DCR 1604.
- 29) PlaneMeasurement/GridMeasurement: Checking for ToleranceTooWideError was broken since change of TouchDirection definition. DCR 1835.
- 30) MC_MovePath - Reporting error 10623 incorrectly when OperationMode = GT_OperationMode#InfiniteRepeat. DCR 2066.
- 31) MC_MovePath - If VelocityScaler input = 0, first segment was still executed before Error output. DCR 2115.
- 32) MC_MovePath - StandStill segments are not counted in ExecutedTotal/ProcessedTotal. DCR 2117.
- 33) GroupReAlignPrimeAxes - Improved for multiple sets of prime axes in a group. DCR 2161.

- 34) MC_MovePath - Fix Rotational Move acc/decel limit to properly limit to MyMachine.Prms.MaxRotationalAcceleration. DCR 2162.
- 35) MC_MovePath - Improved update of ProccesedTotal and ExecutedTotal outputs for non motion segments. DCR 2183.
- 36) G-Code Processing - Expression evaluator not accepting decimal point or the negative of another variable in a variable assignment. DCR 2185.
- 37) GetDOFIndex - Rectify issue with GroupToHome / GetDOFIndex. (wrong implementation to find axes in AxesGroup.) DCR 2215.
- 38) Machine_Driver - Not propagating Errors to be caught by Pendant_Driver / GroupCommDLL. When invoked via GroupComm DLL, GroupToHome could silently Error. DCR 2216.
- 39) G-Code - Feedrate for Rotational moves must not have ScaleFactor applied. DCR 2223.
- 40) G-Code - IF statements with equal to condition incorrectly sets variable equal to the value being checked. DCR 2224.
- 41) MC_MovePath - Limit check feedrate based on move type (linear, Rotary, direct) DCR 2226.
- 42) G-Code - Add GCode ModalMode for motion type (G 0,1,2,3)
- 43) G-Code - G4 with X parameter sets SegmentType 1 (linear move) instead of a dwell. DCR 2259.
- 44) G-Code - Improve IF logic to throw error if jump-to line number not found. DCR 2272.
- 45) G-Code - M109 is not updating specified minimum temperature. DCR 2291.
- 46) MachineData.Printer.ActiveExtruder did not indicate which extruder is running. DCR 2292.
- 47) MC_MovePath - Velocity limit checking must take into account the current MC_GroupSetOverride. DCR 2324.
- 48) G-Code - Dont increment PathData.Buffer.StorePointer if there was an Error in the PathStruct Update code. DCR 2348.
- 49) G-Code - CNCWorks MDI mode causes re execution of all segments in PathData each time MDI command is sent. DCR 2365.
- 50) MC_MovePath - Output labels / ExecutedTotals incorrect in SingleStep mode for non motion segments. DCR 2417.
- 51) GroupControl - Changed so GroupStatus output does not flicker. DCR 2077.
- 52) MC_MovePath - Force a delay when setting OutputFlags if SegmentType is a motion type but Group Prm 2201 reports 0. Maybe motion not started yet, especially for MotoManSync groups. DCR 2080.
- 53) G-Code Processor / Tool Compensation - code refactoring for the v340 release broke Tool Compensation. DCR 2118.
- 54) MC_Movepath - SetTangent SegmentType not setting Acceleration and Velocity.DCR 2126.
- 55) Read_GCode_File - Did not properly shut down GCode_Processor if there was an error, next Execute falsely showed lingering Error. DCR2182.
- 56) GroupCommManager / Pendant_Driver -Servo power could not be toggled in some cases after a communication loss and recovery. DCR 2372.
- 57) MC_MovePath - In ExecutionMode=SingleStep, Output Labels / ExecutedTotals were not accurate for non motion segments. DCR 2417.
- 58) Read_GCode_File / Read_GCode_Stream - improved checking of string lengths to avoid causing "Error on String Conversion" Error. DCR 2455
- 59) Pendant_Driver - change behavior so servos for Mechatrolink group disable if there is a Comm watchdog Error. DCR 2616.

2018-08-07 v350 released. fw 3.4.0 or higher required.

New Features, Improvements

- 1) GroupCommManager - New function block added to support GroupComm.DLL and CNCWorks PC application.
- 2) Merged Pendant Toolbox into Group Toolbox, first official release of Pendant features. DCR 1570.
- 3) 3D Printing - Added support for Extruder scaling. DCR 1497.
- 4) 3D Printing - Added support for multiple extruder temperatures. DCR 1508.
- 5) 3D Printing - Added support for M106 processing - FanSpeed. DCR 1538.
- 6) 3D Printing - Added support for multiple filament / pellet extruders. DCR 1704.

- 7) 3D Printing - Added support for M109: Set Extruder Temperature and Wait.
- 8) MC_PATH_DATA_REF - Added support for stream byte offset referencing. DCR 1299.
- 9) Read_GCode_Stream - Allow Errors to be reset without dropping socket connection. DCR 1464. DCR 2394.
- 10) G-Code parsing - Added support for G93 - Inverse Time mode. DCR 1609.
- 11) Pendant Driver - Using Memory map version 3.5.0.7. DCR 1631 / 1953.
- 12) Pendant Driver - Added support for MC_GroupSetOverride. DCR 1429.
- 13) Pendant_Driver - 'TriggerReset' bit now clears controller, group, and axes alarms if present. DCR 1750.
- 14) Machine_Driver FB - Added support for Group_To_Home FB via Memory Map interface. DCR 1448.
- 15) GetGroupNames - New function blocked. DCR 1386.
- 16) GridMeasurement - New Function block added. DCR 1728.
- 17) GridLookup - New Function block added. DCR 1729.
- 18) WriteGridFile & ReadGridFile - New Function blocks added. DCR 1745.
- 19) MachineStruct - XYZ extruder offsets Added. DCR 1752.
- 20) Added support for G-Codes G28, M82 and M84. DCR 1779.
- 21) Provide method to specify PrimeAllowance for the GroupReAlignPrimeAxes FB inside Pendant_Driver. DCR 1807.
- 22) GroupReAlignPrimeAxes - more efficient / faster, now uses previously populated JointMap as VAR_IN_OUT. DCR 1908.

Changes:

- 23) Repurposed *.JOB file output to *.TPL (Teach Point List) DCR 1419.
- 24) Pendant_Driver - Refactored VAR_IN_OUTs 4 separate structures into one structure (PendantData) DCR 1478.
- 25) Refactored DetectPrimeAxes FB -> GetPrimeAxesRef FB. DCR 1515.
- 26) PlaneMeasurement - Output changed to provide MC_CARTESIAN_REF for use with Y_GroupSetFrameOffset. DCR 1873.
- 27) GroupReAlignPrimesAxes and AlignPrimeAxes renamed to GroupReAlignPrimes etc. in prep for future function. DCR 2034

Bug Fixes

- 28) MC_MovePath - Improved JoinMode / PathMode determination for first segment. DCR 1205.
- 29) MC_MovePath - Improved determination of RotationalOnly Motion for specific CoordinateSystem. DCR 1273.
- 30) MC_MovePath - MC_MovePath.Abort was causing group to get stuck in Stopping state. DCR 1461.
- 31) Fixed G-Code variable support for Feedrate to apply conversion from per/in to per/sec and scalefactor. DCR 1484.
- 32) G-Code parsing - (CheckColinearity FB) - TangentAngle should never be negative. DCR 1485.
- 33) Improved CheckColinearity FB to also check for Rotational combinations that would cause blend Errors. DCR 1506.
- 34) G-Code parsing - Fix reporting of FileByteOffset to support resuming jobs in progress. DCR 1545.
- 35) G-Code parsing - G10 should only update offsets in MachineStruct, G54 ~ 59 should only apply specified offsets.
- 36) Pendant Driver - Fix reading of group name, use ChangeCase and GetGroupName. DCR 1727.
- 37) MC_MovePath - Rotational velocity was only taken from MachineStruct, ignoring Pathstruct.Segment.Feedrate. DCR 1770.
- 38) GroupReAlignPrimeAxes - Improved for groups with multiple sets of joints with prime servos. DCR 1793.
- 39) MC_MovePath - Improved synchronization of outputs to queued motion segments. DCR 1926.

2017-12-07 v340 released. firmware 3.4.0 or higher required.

New Features and Improvements:

- 1) AlignPrimeAxis - New function block added. DCR 1053.

- 2) PlaneMeasurement - New function block added. DCR 1103.
- 3) GroupToHome - New function block added. DCR 1234.
- 4) MachineStruct - Added .MachineType to select milling, lathe, printer modes.
- 5) GroupControl - Added ability to execute MC_Reset on all axes configured in Group. DCR 1080 / 1090.
- 6) MC_MovePath - Improved handling of non colinear segments to avoid ErrorID 9000,9001,9002,9003. DCR 1012
- 7) MC_MovePath - Improved Segment tracking so path outputs can be operated accurately. DCR 1083.
- 8) Read_GCode_File - Added Abort input to cancel reading a file. DCR 1093.
- 9) Support G-Code variables - DCR 1095.
- 10) Read_GCode_File - Improve scan time stability. Improvements to minimize scan time spikes caused when the function processes bytes into the PathStruct. DCR 1222.
- 11) MachineStruct.EmulationMode - Support for alternate way of setting vel and accel in units/sec.
- 12) MC_PATH_DATA_REF.StreamStatus - Added support for byte offset referencing for G-Code so host can determine the live motion segment. DCR 1224.
- 13) Read_GCode_Stream - Expose ErrorRow and ErrorString as VAR_OUTPUT. DCR 1208.
- 14) MachineStruct - Added MaxAcceleration and MaxDeceleration. DCR 1239.
- 15) G-Code - Added support for mathematical expressions and subprograms. DCR 1240.

Changes:

- 16) MC_MovePath - Some Outputs changed for clarity. Now label and segment are identified as 'Processed' and 'Executed'. DCR 1092.
- 17) MC_MovePath - changed VAR_INPUT VelocityOverride to VelocityScaler to differentiate from new MC_GroupSetOverride released in firmware v3.4.0. DCR 1238.

Bug Fixes:

- 18) MC_MovePath - Fix bug with total motion block count. Sometimes the value would go negative. DCR 1094.
- 19) G-Code processing - Bug fix in ColinearityCheck for Line to Arc, Arc to Line cases. DCR 1283.
- 20) G-Code - Processing of I,J,K registers for circle definition was not being properly scaled if required. DCR 1305.
- 21) G-Code G04 was interpreting the P parameter in seconds but the G-Code RS 274 specification indicates milliseconds. Now the P parameter is milliseconds, and the X parameter can be used to specify seconds. DCR 1382.

2017-01-07 v330 released

This was the first release.



G-Code Support

G-Codes

Command	Description	Support	Comment
G00	Rapid positioning.	Yes	G0 moves the machine using MachineData.Prms.MaxDirectVelocity, MachineData.Prms.DirectAcceleration, and MachineData.Prms.DirectDeceleration. If the F register is included on the same line as the G0, F will be used instead of MyMachine.Prms.MaxDirectVelocity unless F exceeds MaxDirectVelocity.
G01	Linear interpolation	Yes	G1, G2, and G3 moves reference the F register for feedrate. If the F register is zero, then MachineData.Prms.MaxVelocity is applied. MachineData.Prms.Acceleration and MachineData.Prms.Deceleration are also referenced.
G02	CW circular interpolation	Yes	Provide R for radius mode or I,J,K for circle center mode.
G03	CCW circular interpolation	Yes	Provide R for radius mode or I,J,K for circle center mode.
G04	Dwell	Yes	Provide the dwell time as milliseconds with the P parameter or as seconds with X.
G05	High-precision contour control (HPCC).	No	Uses a deep look-ahead buffer and simulation processing to provide better axis movement acceleration and deceleration during contour milling.
G05	Message	Yes	A message can be displayed on Compass by configuring G5 as an override command. See " Configuring a G5 Message Dialog " in the Compass OEM Customization and Configuration Guide.
G06	Non uniform rational B Spline machining.	No	Uses a deep look-ahead buffer and simulation processing to provide better axis movement acceleration and deceleration during contour milling.
G07	Imaginary axis designation	No	Uses a deep look-ahead buffer and simulation processing to provide better axis movement acceleration and deceleration during contour milling.
G09	Exact Stop, non-modal.	Yes	Forces the move to come to a complete stop rather than blending with the next move if there is another motion command following. Modal version is G61.
G10	Select work coordinate and tool offsets.	No	Not supported.
G11	Data write cancel	No	
G12	Full-circle clockwise interpolation	No	
G13	Full-circle counterclockwise interpolation	No	
G17	XY Plane Selection	No	This is the default.
G18	ZX Plane Selection	No	
G19	YZ Plane Selection	No	

Command	Description	Support	Comment
G20	Programming in inches	Yes	The Group Toolbox G-Code feature will default to the user units configured in the Hardware Configuration. If all G-Code files will contain position data in the configured user units of the machine, G20 / G21 are not required. If G-Code files may contain different user units, the application program must read parameter 1813 using the HC_ReadParameter function block to obtain the configured user units. Copy this parameter value into PathData.HC_UserUnits. This allows the G-Code parser to convert positions in the G-Code file to the configured units of the machine.
G21	Programming in millimeters	Yes	
G28	Return to the Home Position	Yes	Return to the machine's reference or Origin position. Specify this position in the MachineStruct.Origin[]. At least one of X, Y, or Z axes must be specified with G28 to indicate a via point to pass through before moving to the Origin.
G30	Return to the Secondary Home Position	No	Takes a P address specifying which machine zero point is desired, if the machine has several secondary points (P1 to P4).
G31	Skip Function	No	Used for probes and tool length measurement systems.
G32	Single-point threading, longhand style	No	Similar to G01 linear interpolation, except with automatic spindle synchronization for single-point threading.
G33	Constant-pitch threading	No	
G34	Variable-pitch threading	No	
G40	Tool radius compensation off	Yes	Turns off cutter radius compensation. Cancels G41 or G42.
G41	Tool radius compensation left	Yes	Creates a left tool compensation along the XY plane. Provide Tool Data via the ToolStruct connected to Read_GCode_File or Read_GCode_Stream.
G42	Tool radius compensation right	Yes	Creates a right tool compensation along the XY plane. Provide Tool Data via the ToolStruct connected to Read_GCode_File or Read_GCode_Stream.
G43	Tool height offset compensation negative	Yes	Uses the H or T register as the tool length offset. Provide Tool Data via the ToolStruct connected to Read_GCode_File or Read_GCode_Stream.
G44	Tool height offset compensation positive	No	
G45	Axis offset single increase	No	
G46	Axis offset single decrease	No	
G47	Axis offset double increase	No	
G48	Axis offset double decrease	No	
G49	Tool length offset compensation cancel	Yes	
G50	Define the maximum spindle speed/Scaling function cancel	No	
G52	Local Coordinate System. This is an offset from the current offset	No	
G53	Machine coordinate system. Takes absolute coordinates (X,Y,Z,A,B,C) with reference to machine zero.	Yes	Make a move in the Machine Coordinate System (MCS.) NOTE: Behavior changed for Group Toolbox v361 to comply with traditional (non modal) G-Code behavior. Prior to v361, executing a G53 would not initiate any motion by itself. See Emulation Modes for other behavior variations.
G54	Work Coordinate System 1	Yes	Selects the offsets in MachineStruct.CoordinateSystem.Offset [1] to be used for moves such as G0, G1, G2, G3.
G55	Work Coordinate System 2	Yes	Selects the offsets in MachineStruct.CoordinateSystem.Offset [2] to be used for moves such as G0, G1, G2, G3.

Command	Description	Support	Comment
G56	Work Coordinate System 3	Yes	Selects the offsets in MachineStruct.CoordinateSystem.Offset [3] to be used for moves such as G0, G1, G2, G3.
G57	Work Coordinate System 4	Yes	Selects the offsets in MachineStruct.CoordinateSystem.Offset [4] to be used for moves such as G0, G1, G2, G3.
G58	Work Coordinate System 5	Yes	Selects the offsets in MachineStruct.CoordinateSystem.Offset [5] to be used for moves such as G0, G1, G2, G3.
G59	Work Coordinate System 6	Yes	Selects the offsets in MachineStruct.CoordinateSystem.Offset [6] to be used for moves such as G0, G1, G2, G3. G59.1 through G59.3 are also supported.
G61	Exact stop check, modal. Can be canceled with G64. Non-modal version is G09.	Yes	
G62	Automatic Corner Override	No	
G64	Default cutting mode. Cancels G61	Yes	
G70	Fixed cycle, multiple repetitive cycle, for finishing (including contours)	No	
G71	Fixed cycle, multiple repetitive cycle, for roughing (Z-axis emphasis)	No	
G72	Fixed cycle, multiple repetitive cycle, for roughing (X-axis emphasis)	No	
G73	Drilling Chip Break Cycle	Yes	In this cycle, the drill makes a rapid move to the retract position above the piece (R), then moves into the piece an increment (Q) at the specified feedrate (F) until the final drill position is reached (Z). At the end of each increment, the drill will rapidly move up a fixed amount of 0.050" (1.27 mm) to "break the chip", then proceed with the next peck. This macro remains enabled until a G80 is executed. Supported Parameters: G73 Xn Yn Zn Qn Rn Fn Pn where: X = X position to drill Y = Y position to drill Z = Final drill depth position. Q = Peck Depth - the amount to be added to the drilling depth each peck cycle. R = Retract position - initial position above material, safe for moving among multiple holes. F = Feedrate for drilling. P = Dwell in mSec after each drill cycle.
G74	Tapping cycle for milling, left hand thread, M04 spindle direction	No	
G75	Peck grooving cycle for turning	No	
G76	Fine boring cycle for milling	No	
G78	Tangent Motion Enable	Yes	Synchronizes a theta axis (which is not part of the AxesGroup) to the XY plane of a path. The external axis must be defined as shown in the example for MC_MovePath . Requires firmware 3.3.0 or higher. This tangent command does not automatically prepare the tangent angle nor automatically raise and lower the Z axis. These activities must be included as commands in the G-Code file. See Examples, and the alternate commands for Tangent Applications, L30 / L31.

Command	Description	Support	Comment
G79	Tangent Motion Disable	Yes	
G80	Cancel Modal Command	Yes	Designates the end of a modal cycle operation such as G73.
G81	Drilling Cycle with Dwell	No	
G82	Spot Drilling cycle (full retraction from pecks)	No	
G83	Deep Drilling Cycle	Yes	<p>In this cycle, the drill makes a rapid move to the retract position above the piece (R), then moves into the piece an increment (Q) at the specified feedrate (F) until the final drill position is reached (Z). At the end of each increment, the drill will rapidly return to the retract position, then proceed with the next peck. This macro remains enabled until a G80 is executed.</p> <p>Supported Parameters: G73 Xn Yn Zn Qn Rn Fn Pn where: X = X position to drill. Y = Y position to drill. Z = Final drill depth position. Q = Peck Depth - the amount to be added to the drilling depth each peck cycle. R = Retract position - initial position above material, safe for moving among multiple holes. F = Feedrate for drilling. P = Dwell in mSec after each drill cycle.</p>
G84	Tapping cycle, right hand thread, M03 spindle direction	No	
G90	Absolute Positioning	Yes	
G90	Lathe: Straight Cutting Cycle	No	If MachineStruct.MachineType is set to Lathe, G90 commands will use a Straight Cutting Cycle macro.
G91	Incremental Positioning	Yes	
G92	Set Position	Yes	Starting in v361, all axes are supported. G92 with no axes specified clears any existing offsets. These "LocalZeroOffsets" are applied to all Work offsets G54 ~ G59 in addition to any offsets established for G54 ~ G59 in MachineData.CoordinateSystem[] . The redefined positions (stored in MachineData.LocalZeroOffsets[]) are temporary, meaning they are lost after power cycle or PLC restart, unless the MachineData variable is set as 'retain' data in the MotionWorks IEC Global variables list.
G93	Inverse Time Mode	Yes	Support added for Group Toolbox v350 release.
G94	Units per Minute Feed Rate Mode	Yes	This was always the default feedrate mode, but the G94 command was officially recognized (No Error) as of the v350 release.
G96	Constant surface speed (CSS)	No	Varies spindle speed automatically to achieve a constant surface speed.
G97	Constant spindle speed	Yes	Takes an S address integer, which is interpreted as RPM. The Group Toolbox does not have any built in support for spindles, other than datatypes to hold information for operating a spindle by customizing the main project using this toolbox. See MachineStruct.Spindle[] .
G98	Absolute Programming	Yes	Only valid if MachineStruct.Emulation:=GTB_Emulation#Mode1
G99	Incremental Programming	Yes	Only valid if MachineStruct.Emulation:=GTB_Emulation#Mode1

M-Codes

Command	Description	Support	Comment
M00	Non-optional Stop. Machine always stops here.	Yes	If using Compass software, press the Cycle Start button to resume after executing an M0. (MC_MovePath.Execute must transition back to TRUE.) This commands requires Group Toolbox v352 or higher. See the Compass Project template "PathMotion POU" which shows a required contact "ProgramStop" for this feature.
M01	Optional Stop. Only stops if Optional stop flag is set. See MachineStruct.Command.ProgramStop	Yes	If using Compass software, press the Cycle Start button to resume after executing an M1. (MC_MovePath.Execute must transition back to TRUE.) Requires Group Toolbox v352 or higher. See the Compass Project template "PathMotion POU" which shows a required contact "ProgramStop" for this feature.
M02	End of Program	Yes	
M03	Spindle On (clockwise rotation)	Yes	Supported as MC_MovePath.OutputFlag.X3. The Group Toolbox does not have any built in support for spindles, other than datatypes to hold information for operating a spindle by customizing the main project using the Group toolbox. See MachineStruct.Spindle[] . For multi spindle support, use commands M143, M144, M145 and M153, M154, M155. Emulation mode 2 also supports additional spindle commands.
M04	Spindle On (counterclockwise rotation)	Yes	Supported as MC_MovePath.OutputFlag.X4, also see MachineStruct.Spindle[] .
M05	Spindle Stop	Yes	Supported as MC_MovePath.OutputFlag.X3 and X4, also see MachineStruct.Spindle[] .
M06	Automatic Tool Change	No*	Can be customized by the OEM.
M07	Coolant On (mist)	Yes	Supported as MC_MovePath.OutputFlag.X0, also see MachineStruct.Spindle[] .
M08	Coolant On (Flood)	Yes	Supported as MC_MovePath.OutputFlag.X1, also see MachineStruct.Spindle[] .
M09	Coolant Off	Yes	Supported as MC_MovePath.OutputFlag.X0 and X1, also see MachineStruct.Spindle[] .
M10	Pallet Clamp On	Yes	Supported as MC_MovePath.OutputFlag.X2, also see MachineStruct.Spindle[] .
M11	Pallet Clamp Off	Yes	Supported as MC_MovePath.OutputFlag.X2, also see MachineStruct.Spindle[] .
M13	Spindle on (clockwise rotation) and coolant on (flood)	Yes	Supported as MC_MovePath.OutputFlags X1 and X3, also see MachineStruct.Spindle[] .
M19	Spindle Orientation	No	
M21	Mirror, X-Axis	No	
M22	Mirror, Y-Axis	No	
M23	Mirror Off	No	
M24	Thread gradual pullout off	No	
M26	Axis Clamping	No	
M27	Axis Clamping	No	
M30	End of program	Yes	
M41	Gear select 1	No	
M42	Gear select 2	No	
M43	Gear select 3	No	

Command	Description	Support	Comment
M44	Gear select 4	No	
M48	Feedrate override allowed	No	
M49	Feedrate override NOT allowed	No	
M52	Unload last tool from spindle	No	
M62	Set Digital Output On	Yes	The P parameter specifies the digital output number 1-32 (Bit of MC_MovePath.OutputFlags)
M63	Set Digital Output Off	Yes	The P parameter specifies the digital output number 1-32 (Bit of MC_MovePath.OutputFlags)
M60	Automatic Pallet change (APC)	No	
M66	Wait for Input	Yes	Reference http://linux-cnc.org/docs/html/gcode/m-code.html#mcode:m66 . P parameter: 0 ~ 31 is mapped to MC_MovePath.InputFlags. P1 is bit 0, P2 is bit 1, and so on. E parameter: Not supported. L Parameter: Only Mode 4 is supported; MC_MovePath simply holds up sequencing until the required input conditions (bit flags) are met. If other input states are desired, connect the necessary logic to MC_MovePath.InputFlags. Q parameter: timeout in seconds.
M98	Subprogram call	Yes	Two methods are supported based on the function block / solution being implemented: #1) Read_GCode_File: Sub program must be within the same file as the main program. This limited feature only supports one file at a time. 16 sub "programs" supported so long as they are all contained within one single file which fits into the PathData structure as a whole. #2) Read_GCode_Stream (Compass): Subroutine file support was added to Compass 1.3.0 and Group Toolbox v373. For more information, see the M98 page. Both implementations support the L parameter to indicate the number of times to repeat the subroutine call.
M99	Subprogram end	Yes	
M104	Set Extruder Temperature	Yes	http://reprap.org/wiki/G-code#M104:_Set_Extruder_Temperature . The P register, if provided, specifies a specific zone in MyMachine.Printer.Extruder.TempSetting[]
M106	Set Fan On (and speed)	Yes	http://reprap.org/wiki/G-code#M106:_Fan_On
M107	Set Fan Off	Yes	http://reprap.org/wiki/G-code#M107:_Fan_Off
M109	Set Extruder Temperature and Wait	Yes	https://reprap.org/wiki/G-code#M109:_Set_Extruder_Temperature_and_Wait . The P register, if provided, specifies a specific zone in MyMachine.Printer.Extruder.TempSetting[]
M117	Display Message	Yes	https://reprap.org/wiki/G-code#M117:_Display_Message The string message will be copied to the MotionWorks IEC variable MyPath.SyncSegments.Message. This string may be referenced from a Compass string display object. Up to 32 messages can be buffered.

Command	Description	Support	Comment
M118	Display Message	Yes	https://reprap.org/wiki/G-code#M118:_Display_Message The string message will be copied to the MotionWorks IEC variable MyPath.SyncSegments.Message. This string may be referenced from a Compass string display object. Up to 32 messages can be buffered.
M140	Set Bed Temperature	Yes	https://reprap.org/wiki/G-code#M140:_Set_Bed_Temperature_.28Fast.29
M190	Set Bed Temperature and Wait	Yes	https://reprap.org/wiki/G-code#M190:_Wait_for_bed_temperature_to_reach_target_temp
M143	Spindle [1] On (clockwise rotation)	Yes	The Group Toolbox does not have any built in support for spindles, other than datatypes to hold information for operating a spindle by customizing the main project using this toolbox. See MachineStruct.Spindle[] .
M144	Spindle [1] On (counterclockwise rotation)	Yes	
M145	Spindle [1] Stop	Yes	
M153	Spindle [2] On (clockwise rotation)	Yes	
M154	Spindle [2] On (counterclockwise rotation)	Yes	
M155	Spindle [2] Stop	Yes	
M207	Retract filament	Yes	

Parameters

Command	Description	Support	Comment
A	Absolute or incremental position of A axis (rotational axis around X axis)	No	This is a Rotational position. (Rx) Actual implementation handled in firmware based on Hardware Configuration support for selected mechanisms, or must implement custom kinematics in the application layer of the MotionWorks IEC project.
B	Absolute or incremental position of B axis (rotational axis around Y axis)	No	This is a Rotational position. (Ry) Actual implementation handled in firmware based on Hardware Configuration support for selected mechanisms, or must implement custom kinematics in the application layer of the MotionWorks IEC project.
C	Absolute or incremental position of C axis (rotational axis around Z axis)	Yes	This is a Rotational position. (Rz) Actual implementation handled in firmware based on Hardware Configuration support for selected mechanisms.
D	Defines diameter or radial offset used for cutter compensation. D is used for depth of cut on lathes.	No	
E	3D Printer - Extruder position	Yes	MachineStruct.MachineType must be set to GTB_MachineType#Printer. Lathe mode not supported for precision feedrate for threading
E	Mode1 emulation: Set acceleration in units/sec.	Yes	If MachineStruct.Emulation:=GTB_Emulation#Mode1 then the E register is used as acceleration in units/sec ² . This emulation mode is mutually exclusive with MachineType GTB_MachineType#Printer.
F	Feed rate	Yes	Specify in units / minute. If MachineStruct.Emulation:= GTB_Emulation#Mode1 then the F register is interpreted as feedrate in units/sec. The system supports sub group feedrate settings for XY, Z, and rotational axes. For example, when G1 Z15 F250 is executed, the feedrate value is applied as the maximum feedrate for the Z axis for this and future Z moves. Later, if a command such as G1 X100 Y100 Z15 F1000 is executed, the feedrate of 1000 might be reduced if it is determine that Z would exceed its specified limit of 250. If an ambiguous Feedrate is provided (such as G1 Fxxx with no motion registers, all subgroup feedrates are cleared to zero. The subgroup feedrates can be viewed in the variable GCodeData .Register GCodeData.ScaledRegister connected to Read_GCode_Stream.
H	Tool length offset	Yes	Same as T register.
I	Arc Center in X axis for G02 or G03	Yes	This is the relative X distance to the center of the circle from the beginning of the arc. If this register is not provided for a G2 / G3, it is assumed to be zero rather than referring to the previous value.
J	Arc Center in Y axis for G02 or G03	Yes	This is the relative Y distance to the center of the circle from the beginning of the arc. If this register is not provided for a G2 / G3, it is assumed to be zero rather than referring to the previous value.
K	Arc Center in Z axis for G02 or G03	Yes	This is the relative Z distance to the center of the circle from the beginning of the arc. If this register is not provided for a G2 / G3, it is assumed to be zero rather than referring to the previous value.

Command	Description	Support	Comment
L	Fixed cycle loop count, other various commands supported when using Emulation Mode 2 .	Yes	Use with M98. See Emulation Modes for addition L command support.
N	Line number (optional)	Yes	If provided, this information is copied to PathData.Segment [].Label and is useful for troubleshooting.
O	Program Name	Yes	Use with M98.
P	Dwell time in milliseconds for G04, and parameter used by some canned cycles, and for jumps.	Yes	This parameter has multiple functions based on the G-Code.
Q	Peck increment in canned cycles (G73 and G83)	No	
R	Radius of an arc	Yes	For use with G02 and G03.
S	Defines speed, either spindle RPM or surface speed depending on the mode. (G96 or G97)	Yes	
T	Tool Selection	Yes	For 3P Printers, the single command on a line T0, T1, or T2 will switch the active Extruder. A G1 instruction is auto inserted to move the selected Extruder head the offset specified in MachineData.Printer.Extruder[]. For milling configurations, the T command can be configured as an Override to call customized code in the MotionWorks IEC project to operate a Tool Changer.
U	Incremental X axis (Ignores G90 and G91)	No	
V	Incremental Y axis (Ignores G90 and G91)	No	
W	Incremental W axis (Ignores G90 and G91)	No	
X	X axis position	Yes	This is a Cartesian position within the working space of the mechanism. Actual implementation handled in firmware based on Hardware Configuration support for selected mechanisms.
Y	Y axis position	Yes	This is a Cartesian position within the working space of the mechanism. Actual implementation handled in firmware based on Hardware Configuration support for selected mechanisms.
Z	Z axis position	Yes	This is a Cartesian position within the working space of the mechanism. Actual implementation handled in firmware based on Hardware Configuration support for selected mechanisms.

Other Supported Commands

Variable and basic logical commands are supported when using Read_GCode_File with the following limitations:

- When using logical statements (M98 or IF command), the entire G-Code file contents must be loaded into MyPath.Segment[]. The MyPath.Buffer.Overwritten flag must not have been set TRUE by the G-Code Processor. This is because the function blocks can only reference data already processed and in the PLC memory area; they cannot search for data in the G-Code file on the controller flash file system, and if the file is being streamed, they have no way to randomly access data which may already be overwritten by the circular buffer, or may not have been sent to the MPiec controller yet.
- Only simple assignments are supported. No nested parenthesis or math equations involving more than one operator are supported.
- Nested subroutines can be called with a maximum depth of 16, otherwise MC_MovePath will output the ErrorID 10626 - Stack Overflow.
- Up to 255 variables assignments or expressions can exist in one G-Code file.

Logic / Expression support

Operator	Example	Comment
IF	IF [expression] 'N' line number	Must include square brackets, parenthesis not allowed. If the logical expression is true, then the line number specified by the N parameter will be executed next. The G-Code file must contain N line numbers for the IF command to work.
LT or <	Less than	IF [#5 < #18] N32
LE or <=	Less than or equal	IF [#362 LE 27] N458
NE or <>	Not equal	IF [#1 <> #2] N3
EQ or =	Equal	IF [#1 = 0] N27
GT or >	Greater than	IF [#500 GT #501] N123
GE or >=	Greater than or equal	IF [#5 >= 18.5] N21

The keywords THEN ELSE END_IF and GOTO are not supported.

Other Keyword support

Character	Example	Comment
/	/G1 X6 Y17	This is an optional block skip command. Lines with a single leading '/' character will be processed only if the MachineStruct.Control.BlockSkip flag is set. Set this flag by customizing the MotionWorks IEC application or GUI software as necessary. Double slashes are always recognized as the start of a comment.



3D Printing

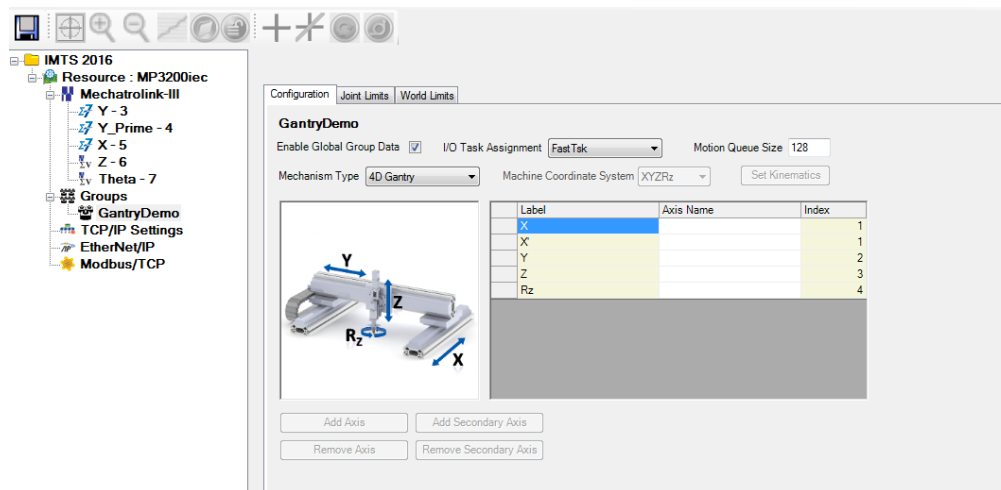
Some special configuration is required to enable 3D printing support.

Configure the Group in Hardware Configuration

Starting with MotionWorks IEC 3.6, there are two methods available to configure extruders; either as rotational axes of an nD gantry, or as additional axes. Configure either a 3D or nD gantry.

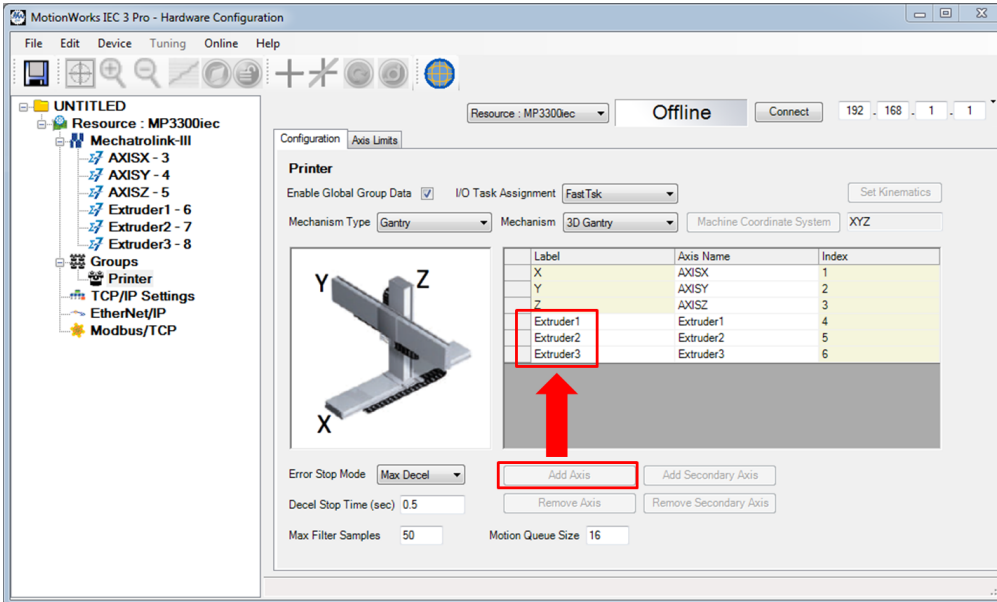
Extruder Configuration Required for MotionWorks IEC 3.5 and Older

Configure the extruder [E0] as the Rz axis. Also consider the FeedConstant value for the Extruder on the Configuration tab for the Extruder axis. The user units for the Extruders must be the same as for the Cartesian (XYZ) axes. Two additional Extruders can also be configured. See the Extruder Mapping Chart that follows.



Extruder Configuration for MotionWorks IEC 3.6 and Newer

Configure the Extruders as additional axes to a 3D or nD gantry. This is the preferred method, although the older method is still supported and there is no need to change an existing configuration when upgrading. Extruders configured as additional axes can be observed in the MotionWorks IEC variable AxesGroup.Machine[7] [8] and [9] or AxesGroup.Part[7] [8] and [9].



Extruder Mapping

			Extruder axes as configured in Hardware Configuration	
			MotionWorks IEC 3.5 and Older	MotionWorks IEC 3.6 and Newer
Description	Reference	Tool #	Group Toolbox v350 and Older	Group Toolbox v352 and Newer
Default Extruder	E0	T0	Rz	Rz or Add Auxiliary Axis
Additional Extruder	E1	T1	Ry	Ry or Add Auxiliary Axis
Additional Extruder	E2	T2	Rx	Rx or Add Auxiliary Axis

MachineStruct Configuration for a 3D Printer

The system must be configured so the function blocks can correctly interpret some special features unique to a 3D Printer. Configure the following in the "Initialize" POU.

```
MyMachine.MachineType:=GTB_MachineType#Printer;
```

Temperature & Fan Management

The function blocks in the Group toolbox recognize G-Codes that pertain to extruder, bed, enclosure temperature and fan speed and copy data into the MachineData structure. There is no built in functionality for physically setting these features. The MPiec project must be customized by connecting the data in the MyMachine.Printer structure.

Temperature related G-Codes:

M104	Set Extruder Temperature
M109	Set Extruder Temperature and Wait
M140	Set Bed Temperature
M141	Set Chamber Temperature
M190	Wait for bed temperature to reach target temp
M191	Wait for chamber temperature to reach target temp



Tangent Mode

As it applies to a G-Code solution, Tangent mode operates a theta axis tangent to the path and can be activated / deactivated using specially designated G-Codes G78 and G79, or if using Emulation Mode2, L30 and L31. Under Emulation Mode 2, the tangent axis is more automatically operated compared to using G78 and G79. When the path exceeds a specified angle between segments, an auto inserted sequence will be implemented to raise, rotate and lower the Z axis. Managing pre-alignment and controlling the Z axis is not required.

Configuring a System for Tangent Operation

Configure All Axes in the Hardware Configuration.

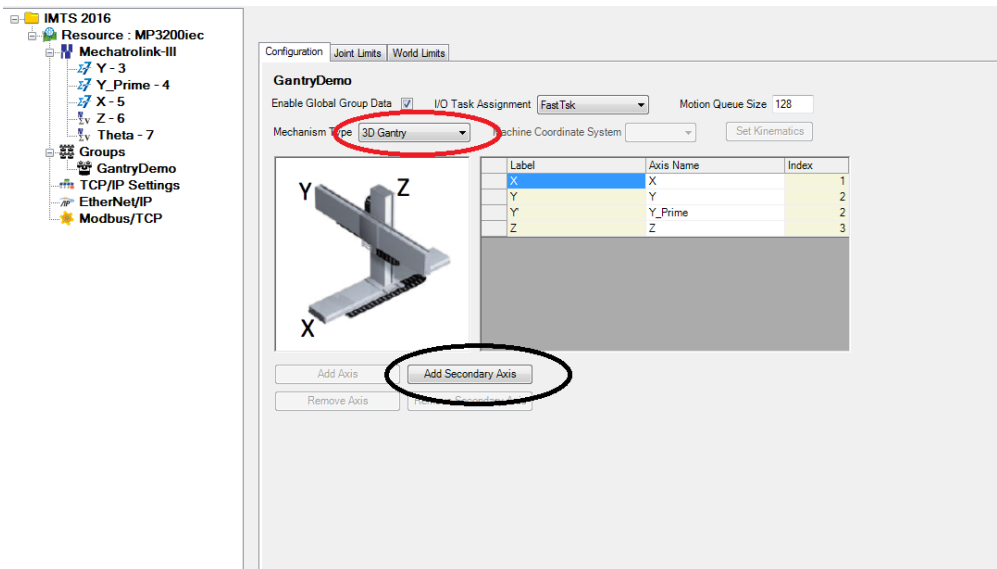
The screenshot shows the 'Hardware' configuration tab in the software. The left sidebar lists the system components, with 'Mechatrolink-III' expanded to show axes: Y-3, Y_Prime-4, X-5, Z-6, and Theta-7. The main configuration area shows the 'Machine Cycle' set to 1. The 'Feed Constant' is 25 Units, and the 'Gear Ratio' is 1. The 'Position Scale' is set to 25 mm. The 'Reference Units per User Unit' is 41943.04. Below the configuration fields is a table of parameters:

Parameter #	Parameters	Current Value	Units	Min	Max	Default Value
1300	Moving Average Filter 1 Enable	False				False
1301	Moving Average Filter 1 Time Constant	0.1	s	0	5	0.1
1807	Load Type	Linear		0	1	Linear
1809	Axis Name	Y				
1831	Logical Axis Number	3		1	512	1
2028	Enable Timeout	300	ms from t	5	10000	300
Pn205	Multi-Turn Limit Setting	65535	Revoluc	0	65535	65535

Important: Configure the Theta axis' LoadType as **Rotary** (prm 1807 above) otherwise it will generate function block ErrorID 61713 when moved.

Configure the Group

Add secondary axes as necessary. The tangent axis is operated externally to the group; do not include it in the Group configuration.



Create an Initialization POU.

In the AxesGroup variable, determine the next available array index beyond the basic group axes and manually add Theta's information. In this example, the GantryDemo group has 4 servos, so the fifth array index is populated with Thetas AXIS_REF.

Although the Theta axis is not officially part of the Group, by manually adding Theta's [AXIS_REF](#) to the AxesGroup.AxisRef structure as shown below on line 5, the Y_GroupPower / [GroupControl](#) function blocks will also power on / off the Theta axis. The functions that manage servo power iterate through the list until finding an invalid AXIS_REF (zero value).

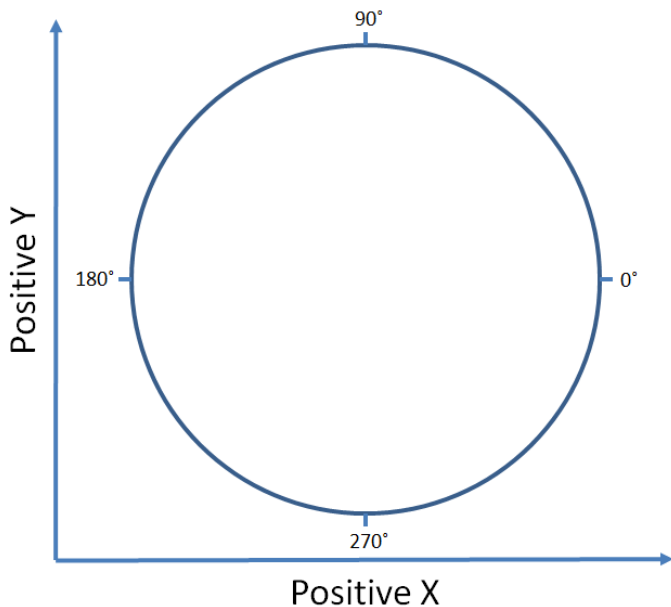
```

5 GantryDemo.AxisRef[5].AxisNum:=INT#5;          (* Add Tangent axis to AxesGroup for Servo control *)
6 MyMachine.ExternalTangent.AxisNum:=INT#5;    (* Configure Tangent axis in MachineStruct for use within MC_MovePath *)

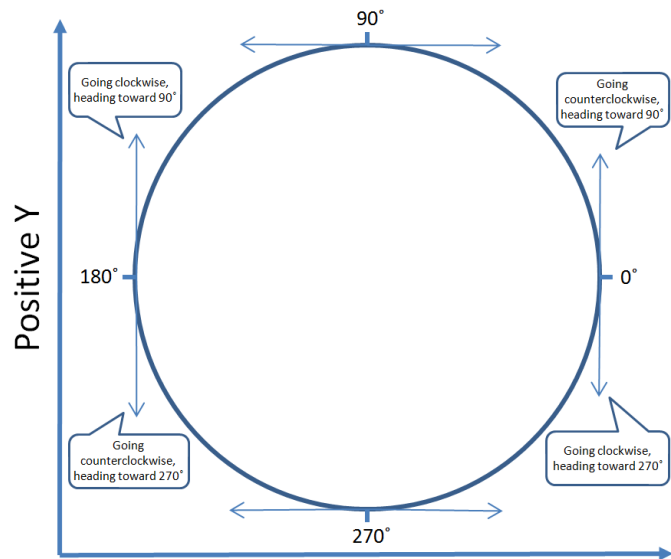
```

About Positioning the Theta Axis

The absolute positions of the Theta are based on the 'unit circle' concept. When looking down at the workspace, with X positive to the right, and Y positive upward, the positions around the unit circle start where 0 degrees = the 3 O'clock position, and increase counterclockwise.



When preparing Theta for tangent motion, its location on the unit circle is not the correct position! Consider the direction of travel, and preset the position according to the tangent heading. Refer to the next graphic.



Example G-Code file for tangent operation

Note: C = Rz = Tangent axis

```

N20 G79 M80 // Tangent off, Head up
N30 G0 X15.4252 Y-57.0835 C270 // Prepare XY and Tangent for next interpolated move
N40 G78 M81 // Tangent On, Head Down
N50 G3 X47.967 Y-120.981 R78.99985 F150 // CCW Arc
N60 G3 X167.9303 Y-120.981 R101.9998 // CCW Arc
N70 G3 X200.4748 Y-57.0835 R78.99985 // CCW Arc
N80 G1 X200.4748 Y-50.2305 // Linear
N90 G3 X179.9749 Y-29.7305 R20.49995 // CCW Arc
N100 G1 X35.92515 Y-29.7305 // Linear
N110 G3 X15.4252 Y-50.2305 R20.49995 // CCW Arc
N120 G1 X15.4252 Y-57.0835 // Linear
N130 G79 M80 // Tangent off, Head up
N140 G0 X20.55605 Y-35.7153 C119.0546 // Prepare XY and Tangent for next interpolated move
N150 G78 M81 // Tangent On, Head Down
N160 G1 X1.68875 Y-1.7541 // Linear
N170 G79 M80 // Tangent off, Head up
N180 G0 X39.4662 Y-116.511 C228.6302 // Prepare XY and Tangent for next interpolated move
N190 G78 M81 // Tangent On, Head Down
N200 G1 X2.7357 Y-158.198 // Linear
N210 G79 M80 // Tangent off, Head up
N220 G0 X177.6449 Y-114.284 C309.9102 // Prepare XY and Tangent for next interpolated move
N230 G78 M81 // Tangent On, Head Down
N240 G1 X214.7583 Y-158.656 // Linear
N250 G79 M80 // Tangent off, Head up
N260 G0 X194.7144 Y-33.784 C58.1595 // Prepare XY and Tangent for next interpolated move
N270 G78 M81 // Tangent On, Head Down
N280 G1 X214.5987 Y-1.7644 // Linear
N290 G79 M80 // Tangent off, Head up
N300 M30 // done

```

Example G-Code file for tangent operation. (Emulation Mode 2)

```

N5 G90 [ABSOLUTE MODE]
N10 L30.1 A15 Z1.050 [SET TANGENT PARAMETERS, MAXANGLE and CLEARANCE HEIGHT]
N15 F300 XY
N20 F75 Z
N25 L30 [TANGENT ENABLE, BUT NOT UNTIL FIRST G1, G2, G3]
N30 G00 X1.006652 Y2.607940
N35 G0
N40 L30.1 P-0.010000 [SET MORE TANGENT PARAMETERS, CUT DEPTH]
N45 G01 X1.007020 Y2.582090 [OVAL, NOW TANGENT AUTOMATICALLY ALIGNS AND PREPARES]
N50 G01 X1.008122 Y2.556319
N55 G01 X1.009952 Y2.530629
N60 G01 X1.008122 Y2.659561
N65 G01 X1.007020 Y2.633790
N70 G01 X1.006652 Y2.607940

[OVAL ABBREVIATED]

N75 G00 [TANGENT WILL RAISE DURING G0 EVEN THOUGH L30 STILL SET]
N80 G00X2.912394 Y6.945807
N85 G0
N90 G01 Y4.945807 [SQUARE, TANGENT AUTOMATICALLY ALIGNS AND PREPARES]
N95 G01 X7.912394
N100 G01 Y8.945807
N105 G01 X2.912394
N110 G01 Y6.945807
N115 G00
N120 G00 X11.012612 Y2.732000
N125 L31 [DISABLE TANGENT]
N130 G53
N135 M30

```



Spindle Support

Up to three spindles can be operated via the standard commands M3, M4, M5 and the S register. There is minimal built-in support for spindles in the Group Toolbox. Support is limited to recognizing the spindle commands, and managing data in MachineData.Spindle[] structure. The user must add the necessary PLC code to the project to operate spindles based on this information.

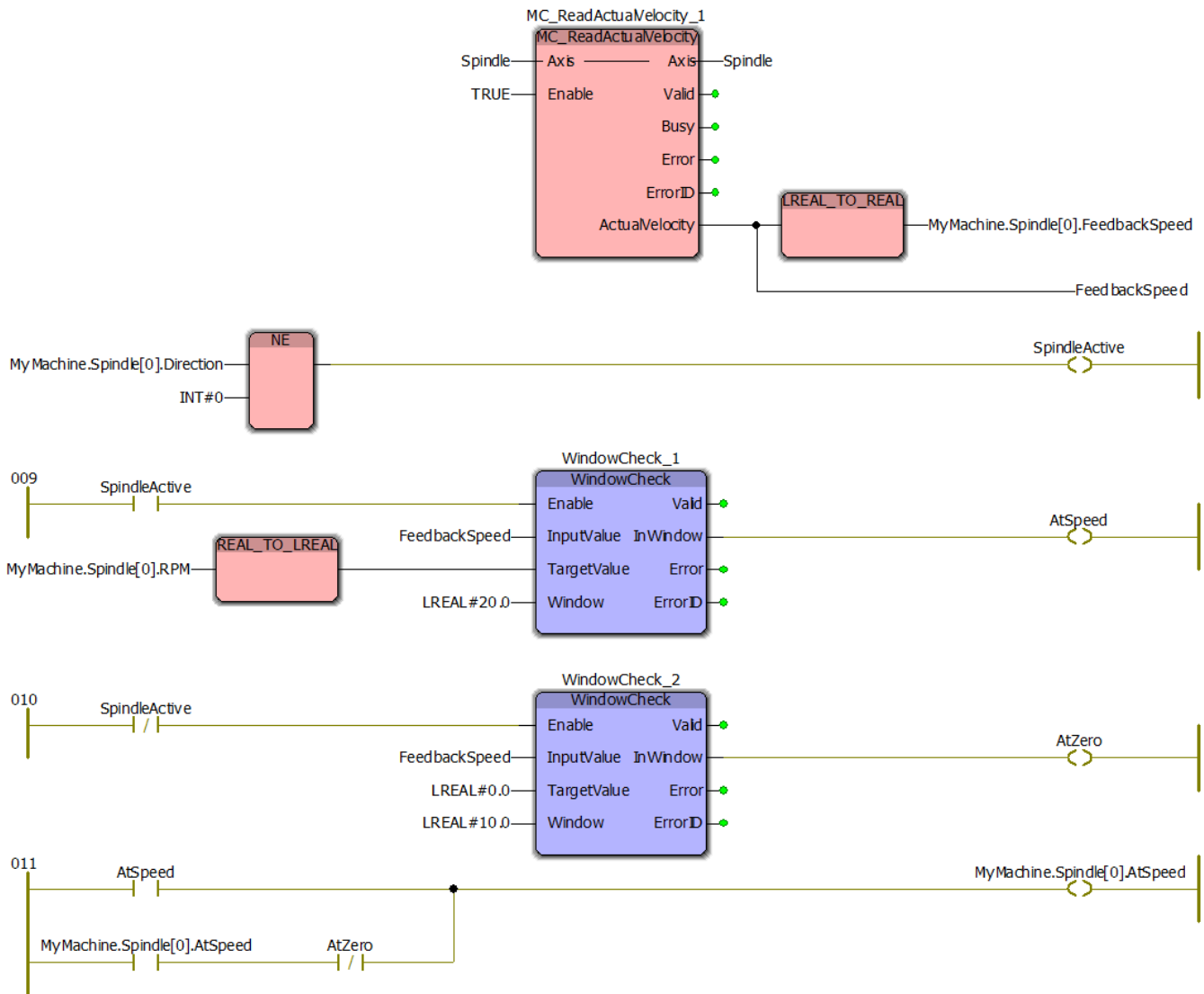
Note: The system will wait for acknowledgment from the user indicating that it's OK to move on with G-Code execution after executing M3, M4, M5 commands. See the structure element MachineData.Spindle[n].AtSpeed. This BOOL must be set high after an M3 or M4 when it's OK to continue on with G-Code execution, and after an M5, it must be set low when it's OK to continue executing G-Codes. See an example for managing the AtSpeed status flag below.

Configuring a System for Spindle Operation

The spindle can be any practical mechanism such as a traditional high speed spindle drive, variable frequency drive, or servo. Due to the various types of devices that can act as the spindle, configuration will vary based on the actual device and is not documented here.

Example

Managing the MachineData.Spindle[].AtSpeed status flag.





Data Type: BufferStruct

This is a sub structure of [MC_PATH_DATA_REF](#) which contains data about the circular buffer of Segments. Most of this data is managed for internal use by the blocks which load and consume Segment information ([Read_GCode_File](#) , [Read_GCode_Stream](#), [MC_MovePath](#))

The G-Code function blocks manage and use this data without any intervention required by the application program, but this information may be useful to view in the Watch Window when debugging. The ExecutedSegment variable listed below is particularly useful when creating an error recovery sequence because it indicates the last Segment Executed when an Error occurred.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyPathStruct.Buffer.	BufferStruct		
C	Size	INT	The Read_GCode_File or Read_GCode_Stream function blocks will set this value on the rising edge of operation by using the UPPER_BOUND and LOWER_BOUND functions for MyPathStruct.Segment.	MyPathStruct.Buffer.Size
C	Max	INT	The Read_GCode_File or Read_GCode_Stream function blocks will set this value on the rising edge of operation by using the UPPER_BOUND function for MyPathStruct.Segment.	MyPathStruct.Buffer.Max
C	Segments	UDINT	The total number of segments in the path.	MyPathStruct.Buffer.Segments
C	StorePointer	INT	Used by Read_GCode_File and Read_GCode_Stream to manage the next available location for segment data.	MyPathStruct.Buffer.StorePointer
C	UsePointer	INT	Used by MC_Move_Path to manage the next segment to execute.	MyPathStruct.Buffer.UsePointer
C	FillCount	INT	Reports the number of Segment between StorePointer and UsePointer.	MyPathStruct.Buffer.FillCount
U	FillMaximum	INT	If the user leaves this value as zero, Read_GCode_File and Read_GCode_Stream will set Fillmaximum to PathData.Buffer.Max. The user can choose to set FillMaximum to a value smaller than Pathdata.Buffer.Max, such as 50% of the max value for cases where debugging the data in the circular buffer is necessary. (When FillMaximum is reduced, old Segment data will remain in the circular buffer for a longer time period.)	MyPathStruct.Buffer.FillMaximum
U	HasCapacity	BOOL	Flag which indicates that there is free space in the circular buffer for additional Segment information to be loaded.	MyPathStruct.Buffer.HasCapacity
C	FillPercent	REAL	Reports the percentage of MC_PATH_DATA_REF that is filled and waiting to be processed.	MyPathStruct.Buffer.FillPercent
C	OverWritten	BOOL	Indicates that the Segment buffer has written to the end, and started writing over the segment data at the beginning again. When this occurs, is not possible to re execute MC_MovePath to make another move sequence. The path data must be re processed, such as by Read_GCode_File or Read_GCode_Stream. Small paths which do not exceed the buffer size of MyPathStruct.Segment can be re run without re processing the original data.	MyPathStruct.Buffer.Overwritten
U	MaxBytesPerScan	UDINT	Allows user to configure number of bytes that StreamParser will allow per scan. Default 500 if not set by user.	MyPathStruct.Buffer.MaxBytesPerScan
U	MaxScansPerLoop	INT	Allows user to configure number of while loop scans allowed in StreamParser (inside Read_GCode_Stream) per task scan. Default 25 if not set by user.	MyPathStruct.Buffer.MaxScansPerLoop
C	MotionAvailable	DINT	Populated in MC_MovePath for the StreamStruct if the Read_GCode_Stream function block is used. This pertains to the firmware level motion buffer and is identical to AxesGroup.Status.FreeMotionSegments.	MyPathStruct.Buffer.MotionAvailable
C	MotionPercent	REAL	Percentage of the firmware motion buffer that is currently used.	MyPathStruct.Buffer.MotionPercent

*	Element	Data Type	Description	Usage
	MyPathStruct.Buffer.	BufferStruct		
C	ExecutedSegment	INT	<p>This value indicates the array index of the PathData.Segment[] linked to the last motion segment providing motion. Because of the possibility of executed motion segments physically causing motion at some time later due to the firmware level motion buffer, there may be situations when the original data in PathData.Segment[] has been overwritten by the time the motion buffer executes the move. ExecutedSegment can be relied on for error recovery situations for small paths which are less than PathData.Buffer.Max and do not require overwriting the Segment data. When streaming PataData, ExecutedSegment is only useful if FillMaximum is manually decreased by the user. For streaming situations when error recovery is required, Yaskawa recommends handling path recovery on the front end GUI side using GCodeComm.DLL. Information is provided which links the last instruction physically providing motion to the byte offset of the G-Code data stream. Using this information, the front end GUI can cancel a path, causing all buffers to reset, and restart the data stream at an appropriate byte offset to allow the machine to resume.</p>	MyPathStruct.Buffer.ExecutedSegment



Data Type: GCodeStruct

GCodeStruct contains internal working information used by the G-Code Processor and is made available as a VAR_IN_OUT by [Read_GCode_File](#) and [Read_GCode_Stream](#) mainly for debugging purposes. There are four main sub structures:

1. G-Code Modal Modes
2. Block Update Flags
3. Registers
4. Variables

If using G-Code data with variables, connect the same structure to MC_MovePath. For example MyGCodeStruct.Variables.

Data Type Declaration

Much of this data is internally managed exclusively by the Group Toolbox and not necessary to interface with directly, but documented for debugging purposes.

*	Element	Data Type	Description	Usage
	MyGCodeData	GCodeStruct		
C	ProcessingMode	GCodeModalModes	Various modes of operation, such as ExactStopCheck, Tangent, Tool Compensation, etc.	MyGCodeData.ProcessingMode.Tangent
C	BlockUpdate	BlockUpdateFlags	Flags to indicate that a certain operation must be performed.	MyGCodeData.BlockUpdate.PathStruct
C	Register	GCodeRegister	All G-Code registers A through Z. This information contains the last received values from the G-Code source. This register list is initialized upon rising edge of either Read_GCode_File or Read_GCode_Stream .	MyGCodeData.Register._G
C	ScaledRegister	GCodeRegister	For registers that require having scaling applied depending if a variable is provided.	MyGCodeData.ScaledRegister._X
C/U	Variables	VariableArray	Only used if the G-Code registers being processed contain variables. In such applications, connect GCodeData.Variables to MC_MovePath	MyGCodeData.Variables[300]
U	UDP_SendInterval	REAL	Value in [ms]. Used in Read_GCode_Stream as the send interval of UDP packets. Default 48 ms if not set by user.	MyGCodeData.UDP_SendInterval
C	G_GroupStatus_1	DWORD	Reports the active G-Code based on pre defined categories.	MyGCodeData.G_GroupStatus_1
C	G_GroupStatus_2	DWORD	Reports the active G-Code based on pre defined categories.	MyGCodeData.G_GroupStatus_2
C	M_GroupStatus	DWORD	Reports the active M Code based on pre defined categories.	MyGCodeData.M_GroupStatus



Data Type: GridResultStruct

For use with the GridMeasurement, WriteGridFile, ReadGridFile, and GridLookup function blocks. The data in this structure is either copied from the GridSetupStruct or populated by the GridMeasurement function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	GridResults	GridResultsStruct		
C	Version	DINT	The structure version as a unique identifier so that the WriteGridFile and ReadGridFile functions can confirm the expected data format.	GridResults.Version
C	Point	Grid	A two dimensional array of GridData.	GridResults.Point[1][2].Measurement
C	MinCorner	YTB_LREAL2	The smallest coordinates of the grid to be measured. Typically XY position data. This data is copied from the GridSetupStruct by the GridMeasurement function block so it can be included in the output file written by the WriteGridFile function block.	GridResults.MinCorner[0]
C	MaxCorner	YTB_LREAL2	The largest coordinates of the grid to be measured. Typically XY position data. This data is copied from the GridSetupStruct by the GridMeasurement function block so it can be included in the output file written by the WriteGridFile function block.	GridResults.MaxCorner[0]
C	GridResolution	INT	Number of measurement points along the side of the grid. This data is copied from the GridSetupStruct by the GridMeasurement function block so it can be included in the output file written by the WriteGridFile function block.	GridResults.GridResolution

Grid sub structure

*	Element	Data Type	Description	Usage
	GridResults.Point	BufferStruct		
C	Grid	ARRAY [1..20] OF GridPoint	Two dimensional array of GridData .	GridResults.Point[1][2].Measurement

GridData sub structure

*	Element	Data Type	Description	Usage
---	---------	-----------	-------------	-------

	GridResults.Point	BufferStruct		
C	Measurement	LREAL	Recorded position when the measurement was taken.	GridResults.Point [1][2].Measurement
C	MeasuredOK	BOOL	Status flag to indicate that the measurement was recorded successfully at the coordinate location. This can be used for debugging if the measurements did not meet tolerance expectations, or other issues result when using the GridMeasuremetn function block.	GridResults.Point [1][2].Measurement

Data Type: GridSetupStruct



For use with the [GridMeasurement](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	GridSetup			
U	MeasureMethod	GTB_MeasurementType	Only Torque sensing is supported. In the future, select torque, digital sensor, or LVDT.	GridSetup.MeasureMethod
U	MeasuredPlane	YTB_STRING8	Provide the name as specified in the Hardware Configuration -> Group -> Label column for the plane to be measured.	GridSetup.MeasuredPlane
U	MinCorner	YTB_LREAL2	The smallest coordinates of the grid to be measured. Typically X Y position data. This data is copied from the GridSetupStruct by the GridMeasurement function block so it can be included in the output file written by the WriteGridFile function block.	GridResults.MinCorner[1]
U	MaxCorner	YTB_LREAL2	The largest coordinates of the grid to be measured. Typically X Y position data. This data is copied from the GridSetupStruct by the GridMeasurement function block so it can be included in the output file written by the WriteGridFile function block.	GridResults.MaxCorner[1]
U	Resolution	INT	Number of measurement points along the side of the grid. This data is copied from the GridSetupStruct by the GridMeasurement function block so it can be included in the output file written by the WriteGridFile function block.	GridResults.Resolution
U	TransferVelocity	LREAL	Velocity used when moving between measurement locations at the Transfer position.	GridResults.TransferVelocity
U	ApproachVelocity	LREAL	Velocity used when moving between the Transfer position and the Approach position.	GridResults.ApproachVelocity
U	MeasurementVelocity	LREAL	Velocity used when moving between the Approach position and the Measurement position.	GridResults.MeasurementVelocity
U	Acceleration	LREAL	Acceleration used with all moves in the measurement process.	GridResults.Acceleration
U	MaxDeviation	LREAL	For use as a monitoring check of all measured points. If MaxDeviation is non zero, the GridMeasurement function block will set its MaxDeviationWarning output after all measurements have been taken and the range of measurements exceeds MaxDeviation.	GridResults.MaxDeviation
U	TransferPosition	LREAL	The position away from the grid to be measured at which the TCP moves to the various measurement points.	GridResults.TransferPosition
U	ApproachPosition	LREAL	The position away from the grid to be measured at which the velocity may be reduced and the torque limited in the direction of measurement to prepare for contact with the surface.	GridResults.ApproachPosition
U	MeasurementPosition	LREAL	The average expected position where the grid will be measured.	GridResults.MeasurementPosition
U	Tolerance	LREAL	Each measurement must be within the Tolerance of the expected MeasurementPosition, or an Error will be generated. This is to ensure that valid measurements are being recorded and the MeasurementTorqueLimit provided is appropriate.	GridResults.Tolerance

*	Element	Data Type	Description	Usage
GridSetup				
U	MeasurementTorqueLimit	DINT	This value is written to ServoPack pn 402 or 403 depending on the direction of the measurements to be taken. Units are percentage of rated torque. It is important to select a torque limit which will not damage the equipment, but allow accurate position reading to be recorded. Ideally, this value is just above the level of torque required to move the axis under normal conditions at the MeasurementVelocity by 1.5 or 2 times.	GridResults.MeasurementTorqueLimit
U	ExternalPlane	GTB_MotorArray	If the group contains a virtual axis for the joint required to measure the grid, the GridMeasurement function block will search the ExternalPlane structure to determine the real servo(s) to set the torque limit and read the position. If the group does not contain a non servo axis on the joint in question, this structure is not used.	GridResults.ExternalPlane

Example Initialization

```

184 gridsetup.MeasureMethod:=GTB_MeasurementType#Torque;
185 gridsetup.MeasuredPlane:='2';
186 gridsetup.Tolerance:=LREAL#0.75;
187
188
189 gridsetup.Acceleration:=MyMachine.Frms.Acceleration;
190
191 gridsetup.Resolution:=INT#10;
192
193 gridsetup.MinCorner[0]:=LREAL#5.0;
194 gridsetup.MinCorner[1]:=LREAL#5.0;
195
196 gridsetup.MaxCorner[0]:=LREAL#230.0;
197 gridsetup.MaxCorner[1]:=LREAL#160.0;
198
199 gridsetup.TransferPosition:=LREAL#20.0;
200 gridsetup.TransferVelocity:=LREAL#200.0;
201
202 gridsetup.ApproachPosition:=LREAL#6.0;
203 gridsetup.ApproachVelocity:=LREAL#25.0;
204
205 gridsetup.MeasurementPosition:=LREAL#2.7;
206 gridsetup.MeasurementVelocity:=LREAL#1.0;
207 gridsetup.MeasurementTorqueLimit:=DINT#10;
208 GridSetup.ExternalPlane[0].AxisNum:=UINT#8;

```

(* Use torque limited touch mode *)
(* Measure the plane which the Z axis contacts *)
(* Measurement position must be within this value of the PlaneSetup.Point[n][3] for the plane axis *)

(* Make a total of 100 measurements in a 10 by 10 pattern *)

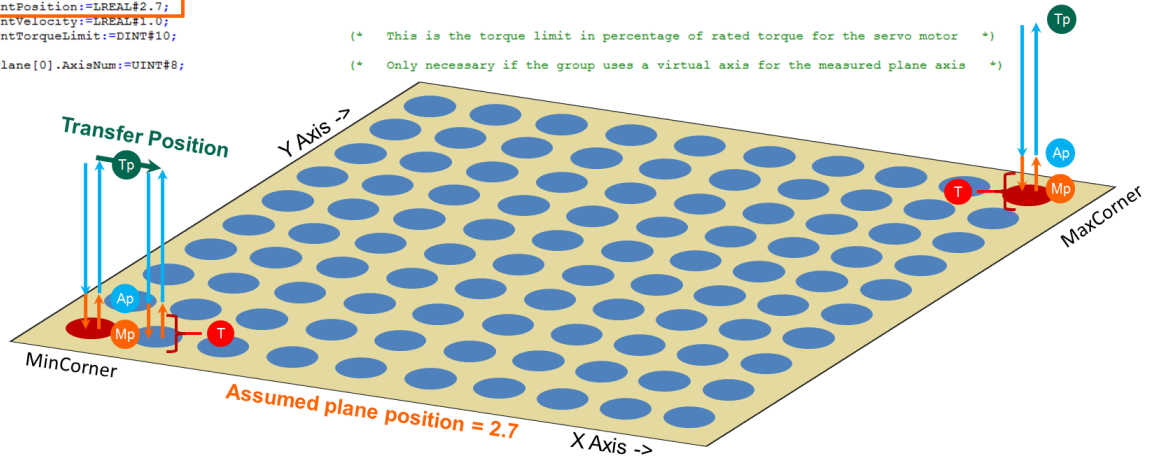
(* Coordinate for first measurement *)

(* Position above plane where the tool moves from one measurement location to the next location *)

(* Position at which the torque limit is applied in preparation for measuring the plane *)

(* This is the torque limit in percentage of rated torque for the servo motor *)

(* Only necessary if the group uses a virtual axis for the measured plane axis *)





Data Type: GroupHomeStruct

GroupHomeStruct contains all the setup information required for the [GroupToHome](#) function block to move the group to a home position in a specified sequence (if required.) This structure is also used with the [GroupCommManager](#) function block because it can execute GroupToHome.

HomeOptionsStruct - Data Type Declaration

This top level structure is used by [GroupCommManager](#) to support configuration of multiple homing sequences. For example, one sequence could home all axes, other sequences could home just a portion, such as X & Y, or simply individual axes.

*	Element	Data Type	Description	Usage
	Sequence	HomeSequenceArray		
U	Method	HomeOptionsArray	ARRAY [0..6] OF GroupHomeStruct	
U	LastMethod	INT	Specify the actual number of methods configured.	MyHomeData.Sequence [2].Acceleration

This is the main structure used with the [GroupToHome](#) function block.

GroupHomeStruct - Data Type Declaration

*	Element	Data Type	Description	Usage
	MyHomeData	GroupHomeStruct		
U	Sequence	HomeSequenceArray	Array of GroupHomeSeqData . This is an instruction list to specify the axes and the sequence in which they should home.	MyHomeData.Sequence [2].DOF[1]
U	DOF	HomeStructArray	ARRAY [1..6] OF GroupHomeData (for each degree of freedom).	MyHomeData.DOF[d].Position
U	PrimeAllowance	LREAL	See Getting Started with Secondary Axes to learn when this parameter applies.	MyHomeData.PrimeAllowance
U	LastSequence	INT	Identify the last Sequence.	MyHomeData.LastSequence

Supporting structure

GroupHomeSeqData Data Type Declaration

*	Element	Data Type	Description	Usage
	Sequence	HomeSequenceArray		
U	Velocity	LREAL	Velocity to be used during the home sequence.	MyHomeData.Sequence [2].Velocity
U	Acceleration	LREAL	Acceleration to be used during the sequence.	MyHomeData.Sequence [2].Acceleration
U	DOF	DOFNames	ARRAY[1..6] OF YTB_STRING32	MyHomeData.Sequence [2].DOF[1]
U	LastDOF	INT	Specify the last Degree of Freedom.	MyHomeData.Sequence [2].LastDOF

GroupHomeData Data Type Declaration

*	Element	Data Type	Description	Usage
	Sequence	HomeSequenceArray		
U	Position	LREAL	Absolute position to move to as the home (Origin) position in the CoordinateSystem specified to the GroupToHome function block.	MyHomeData.DOF[d].Position
U	Direction	INT	Reserved for future use.	MyHomeData.DOF[d].Direction

Example Initialization of GroupHomeStruct

This example homes a three axis gantry by first raising the Z upward to 12mm , then moving the X & Y axes to their home positions.

MyMachine is a MachineStruct, shown here for the initialization of the Origin, used by the HomeData.

HomeData is a GroupHomeStruct.

```

158 MyMachine.Origin[1]:=LREAL#0.0;          (* The X home position is 0 mm *)
159 MyMachine.Origin[2]:=LREAL#0.0;          (* The Y home position is 0 mm *)
160 MyMachine.Origin[3]:=LREAL#12.0;         (* The Z home position is 12 mm *)
161
162 FOR d:=INT#1 TO INT#3 DO
163     HomeData.DOF[d].Position:=MyMachine.Origin[d];
164 END_FOR;
165
166 HomeData.Sequence[1].DOF[1] :='Z';
167 HomeData.Sequence[1].Velocity:=LREAL#150.0;;
168 HomeData.Sequence[1].Acceleration:=LREAL#150.0;
169 HomeData.Sequence[1].LastDOF:=INT#1;      (* Only the Z axis should move in the first sequence *)
170
171 HomeData.Sequence[2].DOF[1] :='X';
172 HomeData.Sequence[2].DOF[2] :='Y';
173 HomeData.Sequence[2].Velocity:=LREAL#150.0;;
174 HomeData.Sequence[2].Acceleration:=LREAL#150.0;
175 HomeData.Sequence[2].LastDOF:=INT#2;      (* The XY axes move in the second sequence *)
176 HomeData.LastSequence:=INT#2;
177
178 HomeData.PrimeAllowance:=LREAL#2.0;       (* mm of deviation allowed if need to align prime axes *)

```

Example Initialization of HomeOptionsStruct

In this example, MyMachine is a MachineStruct, it's values are not shown, but MyMachine.Origin[d] can be any valid positions.

Lines 81 through 84 serve as constants, indicating that there are four home methods configured.

The primary method is "HomeAll" as seen in lines 86 through 102. The sequence is the same as the example for GroupHomeStruct.

The remainder of the initialization shows the specific variations required to create the other three methods, which are all based on the HomeAll method. The following initialization makes it possible for homing all axes or individual axes.

```
81 HomeAll:=INT#0;
82 HomeX:=INT#1;
83 HomeY:=INT#2;
84 HomeZ:=INT#3;
85
86 FOR d:=INT#1 TO INT#3 DO
87     HomeData.Method[HomeAll].DOF[d].Position:=MyMachine.Origin[d];
88 END_FOR;
89
90 HomeData.Method[HomeAll].Sequence[1].DOF[1] :='Z';
91 HomeData.Method[HomeAll].Sequence[1].Velocity:=LREAL#150.0;
92 HomeData.Method[HomeAll].Sequence[1].Acceleration:=LREAL#150.0;
93 HomeData.Method[HomeAll].Sequence[1].LastDOF:=INT#1;
94
95 HomeData.Method[HomeAll].Sequence[2].DOF[1] :='X';
96 HomeData.Method[HomeAll].Sequence[2].DOF[2] :='Y';
97 HomeData.Method[HomeAll].Sequence[2].Velocity:=LREAL#150.0;
98 HomeData.Method[HomeAll].Sequence[2].Acceleration:=LREAL#150.0;
99 HomeData.Method[HomeAll].Sequence[2].LastDOF:=INT#2;
100 HomeData.Method[HomeAll].LastSequence:=INT#2;
101
102 HomeData.Method[HomeAll].PrimeAllowance:=LREAL#2.0;          (* mm of deviation *)
103
104
105 (* Single joint home methods inherit same settings as HomeAll *)
106 HomeData.Method[HomeX].DOF[1]:=HomeData.Method[HomeAll].DOF[1];
107 HomeData.Method[HomeX].Sequence[1].DOF[1] :='X';
108 HomeData.Method[HomeX].Sequence[1].Velocity:=LREAL#150.0;
109 HomeData.Method[HomeX].Sequence[1].Acceleration:=LREAL#150.0;
110 HomeData.Method[HomeX].Sequence[1].LastDOF:=INT#1;
111 HomeData.Method[HomeX].LastSequence:=INT#1;
112
113 HomeData.Method[HomeY]:=HomeData.Method[HomeX];              (* Copy first to get same values, then change as necessary on next lines *)
114 HomeData.Method[HomeY].DOF[2]:=HomeData.Method[HomeAll].DOF[2];
115 HomeData.Method[HomeY].Sequence[1].DOF[1] :='Y';
116 HomeData.Method[HomeY].PrimeAllowance:=HomeData.Method[HomeAll].PrimeAllowance;
117
118 HomeData.Method[HomeZ]:=HomeData.Method[HomeX];              (* Copy first to get same values, then change as necessary on next lines *)
119 HomeData.Method[HomeZ].DOF[3]:=HomeData.Method[HomeAll].DOF[3];
120 HomeData.Method[HomeZ].Sequence[1].DOF[1] :='Z';
121
122 HomeData.LastMethod:=HomeZ;
```



Data Type: JointMap

The JointMap is populated by the [ReadJointMap](#) and [DetectPrimeAxes](#) function blocks. Other functions such as [GroupReAlignPrimeAxes](#) and [GroupToHome](#) require this data. JointMap provides a link between the Axis specific information such as AXIS_REF and AxisName and Group specific information such as Joint and Label. Functions using JointMap are able to determine if a group has more than one axis operating a Joint, and find the AXIS_REF for an axis operating one of the Degrees of Freedom labeled as a STRING such as 'X'.

Data Type Declaration

JointMap is an ARRAY [1..32] OF JointMapDetails:

*	Element	Data Type	Description	Usage
	JointMapDetails			
C	Label	STRING	The name (Label) of a degree of Freedom in World Space as listed in the Hardware Configuration for the group.	JointMap.Label
C	Joint	INT	The Index of the Joint that corresponds to the Label as listed in the Hardware Configuration for the group.	JointMap.Joint
C	AxisName	STRING	The AxisName as listed in the Hardware Configuration for the Axis, also listed as parameter 1809.	JointMap.AxisName
C	AxisRef	UINT	The AXIS_REF.AxisNum that corresponds to the AxisName.	JointMap.AxisRef

Example JointMap

Watch Window

Variable	Value	Type
JointMap		JointMap
[1]		JointMapDetail
Label	X	STRING
Joint	1	INT
AxisRef	5	UINT
[2]		JointMapDetail
Label	Y	STRING
Joint	2	INT
AxisRef	3	UINT
[3]		JointMapDetail
Label	Y_Prime	STRING
Joint	2	INT
AxisRef	4	UINT
[4]		JointMapDetail
Label	Z	STRING
Joint	3	INT
AxisRef	6	UINT
[5]		JointMapDetail
Label	ExternalTangent	STRING
Joint	0	INT
AxisRef	7	UINT
[6]		JointMapDetail
Label		STRING
Joint	0	INT
AxisRef	0	UINT



Data Type: LogicStruct

A sub structure of [MC_PATH_DATA_REF](#) which contains data for managing paths with looping or jumping events.

Data Type Declaration

LogicStruct is managed exclusively by the Group Toolbox, but documented here for debugging purposes.

*	Element	Data Type	Description	Usage
	MyPathPairs	PathPairs		
C	Event	LogicArray	An array [1..256] OF of LogicData	MyPath.Logic.Event[1].ReturnSegment
C	Stack	StackArray	An array of 16 INTs to manage nested Logic Events	MyPath.Logic.Event.Stack[3]
C	Events	INT	To determine which LogicData in LogicArray have already been allocated.	MyPath.Logic.Events
C	StackPointer	INT	Points to the next location in the stack to hold the next event encountered.	MyPath.Logic.StackPointer



Data Type: MachineStruct

NOTE: Starting in Group Toolbox v361, the communication method in GroupComm DLL was changed to provide more flexibility to avoid compatibility issues when features are changed or added. MachineStruct0120190110 was the final structure used with the original communication protocol as supported by the components shown in the chart below. Group Toolbox v361 still supports the previous communication method. The toolbox will convert the modern structure definition to the older version so that the GroupComm DLLs shown below can still communicate.

MachineStruct contains important configuration and other application specific data used by [MC_MovePath](#), [Read_GCode_File](#), [Read_GCode_Stream](#). This structure is shared with a PC application such as CNCWorks or Compass via the GroupComm DLL.

Version information

Version information is relevant if using [GroupCommManager](#) and GroupComm.DLL to operate the machine via PC software.

	Compatibility		
January 2022	1.0.7.20	n/a	v373d
July 2021	1.0.7.15	n/a	v372
December 2020	1.0.6.7	n/a	v371
April 2020	1.0.5.8	n/a	v370
February 2019	1.0.3.1	0120190110	v352
August 2018	1.0.2.1	0120180810	v350

Data Type Declaration

Note: In the chart below 'C' means the data is populated by a toolbox function block and should not or cannot be changed by the user. 'u' indicates that the User must in some way provide this information. In some cases the value can simply be initialized in a WARM_START POU, others can be updated via a PC GUI application software such as Compass.

*	Element	Data Type	Description	Usage
	MyMachine			
C	Version	UDINT	Internal value which identifies the structure format. GroupComm.DLL uses this value to confirm the contents and manage the data appropriately.	MyMachine.Version
U	MachineType	GTB_MachineType	Specify: GTB_MachineType#Milling, GTB_MachineType#Lathe, or GTB_MachineType#Printer to allow the G-Code processor to handle specific G-Codes appropriately based on the Machine Type. Default is Milling.	MyMachine.MachineType
U	Variant	GTB_Variant	Specify a machine variant to customize the implementation. Only applicable if MachineType=Printer. Supported Variants include: GTB_Variant#PrinterXD = 4 axis printer - XYZ and rotating base as C axis. GTB_Variant#PrinterRobot = 6 axis robot arm printer GTB_Variant#SimultaneousExtruder = Requires GTB_MachineType#Printer, and allows additional syntax for E registers such as N12 G01 X10 Y20 Z30 E1=11.1 E2=5 E3=15 E4=35 E5=-49.9 E6=20 F12000. Up to 6 extruders are supported.	MyMachine.Variant
U	Emulation	GTB_Emulation	Specify an emulation mode to be applied. See the Emulation mode support information for details.	MyMachine.Emulation
U	Feature	DWORD	To specify special operations, such as constant acceleration feedrate override, or the ability to set different jog speeds during incremental jog. See the chart below. If using Compass software, Compass writes this information when connecting to the MPiec controller. Features can be configured in the Options -> Feature tab.	MyMachine.Feature
U	Control	ControlData	Structure of items intended for PC application to control the machine via the GroupComm.DLL interface	MyMachine.Control.CycleStart
U	Origin	MC_CARTESIAN_REF	The home position of the machine. Used in conjunction with the GroupToHome function block and G-Code 28.	MyMachine.Origin[x]
U	Home	HomePresetStruct	For with MyMachine.Emulation:=GTB_Emulation#mode2. The Home Presets are referenced with G-Code commands such as G1 H4, where the H4 indicates to move the machine to Home Preset #4. Home Presets can be entered from the Compass Software.	MyMachine.Home[1]
U	LocalZeroOffset	VECTOR	Requires Mode2 Emulation mode. Stores additional offsets applied to MachineData.CoordinateSystem[] Offsets. LocalZeroOffsets are set via G92.	MyMachine.LocalZeroOffset[x]
U	LocalZeroOffsetInhibit	BOOL	Requires Mode2 Emulation mode. Referenced when executing G92.1 and G92.3.	MyMachine.LocalZeroOffsetInhibit

*	Element	Data Type	Description	Usage
	MyMachine			
U	G92RotationInhibit	BOOL	Original behavior of the G92 (G92RotationInhibit:=FALSE) allows the Work Offset to be rotated in three dimensions. When (G92RotationInhibit:TRUE) the system will generate an error if a A, B, or C register is provided with the G92 command. In this mode, use the G68 command to rotate the work offset.	MyMachine.G92RotationInhibit
U	CoordinateSystem	OffsetStruct	Holds up to 9 offsets, corresponding to G54 through G59.3. Important: If the offset values will be written to this structure from an HMI or other method within the IEC application program, they MUST be entered in the user units of the machine as defined in the Hardware Configuration, not the user units of a G-Code file. If a G 10 Code is processed, the G-Code processor will convert the offset values into the native units of the group when storing to the MachineStruct.	MyMachine.CoordinateSystem.Offset[x]
U	Prms	MotionParameters	User configurable parameters such as max velocity and max acceleration.	MyMachine.Prms.MaxVelocity
U	Printer	PrinterStruct	Special data for operating a 3D Printer. This data is typically collected from the G-Code stream and populated here for access by the IEC application program, which must be customized as required for the application.	MyMachine.Printer.Extruder.Axis.AxisNum
U	Tangent	TangentStruct	Data for operating a Tangent axis. Set Axis to a the AXIS_REF of the Tangent axis when using this mode. Some values will be populated by G-Code L30.1. The A register sets MaxAngle, P sets CutPosition, Z sets RaisedPosition. With Group Toolbox v374 and higher, MaxRotationalVelocity is referenced by the lookahead algorithm to slow the motion path if the programmed feedrate would force the Tangent axis to exceed this setting.	MyMachine.Tangent.MaxAngle
U	Laser	LaserStruct	Data for laser engraving applications using the M77 /M78 commands and the LaserControl function block.	MyMachine.Laser.Enable
U	Spindle	SpindleStruct	Contains data specific to operation of a Spindle axis which is externally operate (Not part of the AxesGroup)	MyMachine.Spindle.RPM
U	UserStatus	WORD	Reserved.	MyMachine.UserStatus
U/C	ToolChanger	ToolChangerStruct	Contains data specific to the operation of a tool changer mechanism which is operated externally to the MC_MovePath function block. This data acts as in interface to the application specific tool changer device, and allows the tool changer and the machine to synchronize their operations.	MyMachine.ToolChanger.RequestedTool
C	Shared	GCodeSharedStruct	Reserved for internal use by the Group Toolbox functions to share data between function blocks that don't have the same VAR_IN_OUT interface.	MyMachine.Shared.GroupHandle

Example Initialization

```

1  (*-----*)
2  (*-----*)
3  (*-----*)
4  (*-----*)
5  MyMachine.MachineType:=GTB_MachineTypeMilling; (* From Group Toolbox DataTypes: GTB_MachineTypes:(Milling, Lathe, Printer); *)
6  MyMachine.Variant:=GTB_Variant#n; (* From Group Toolbox DataTypes: GTB_MachineTypes:(Milling, Lathe, Printer); *)
7  MyMachine.Emulation:=GTB_Emulation#mode2;
8
9
10 MyMachine.Prms.MaxVelocity:=LREAL#500.0; (* In configured group units / sec *)
11 MyMachine.Prms.ZMaxVelocity:=LREAL#200.0; (* In configured group units / sec. Only used in MyMachine.Emulation:=GTB_Emulation#mode2; *)
12 MyMachine.Prms.MaxDirectVelocity:=LREAL#2000.0; (* In configured group units / sec. *)
13 MyMachine.Prms.RotationalVelocity := LREAL#3000.0; (* For Rz or Tangent axis when aligning to the next required tangent vector *)
14 MyMachine.Prms.MaxRotationalVelocity:=LREAL#15000.0; (* In configured group units / sec *)
15
16 MyMachine.Prms.Acceleration:=LREAL#9800.0; (* In configured group units / sec2 *)
17 MyMachine.Prms.DirectAcceleration:=LREAL#4000.0; (* In configured group units / sec2 *)
18 MyMachine.Prms.ZAcceleration:=LREAL#9800.0; (* In configured group units / sec2. Only used in MyMachine.Emulation:=GTB_Emulation#mode2; *)
19 MyMachine.Prms.RotationalAcceleration := LREAL#7000.0; (* For Rz or Tangent axis when aligning to the next required tangent vector *)
20 MyMachine.Prms.Deceleration:=LREAL#9800.0; (* In configured group units / sec2 *)
21 MyMachine.Prms.DirectDeceleration:=LREAL#4000.0; (* In configured group units / sec2 *)
22 MyMachine.Prms.MaxAcceleration:=LREAL#9800.0 * LREAL#2.0; (* In configured group units / sec2 *)
23 MyMachine.Prms.ZMaxAcceleration:=LREAL#9800.0 * LREAL#2.0; (* In configured group units / sec2. Only used in MyMachine.Emulation:=GTB_Emulation#mode2; *)
24 MyMachine.Prms.MaxDeceleration:=LREAL#9800.0 * LREAL#2.0; (* In configured group units / sec2 *)
25
26 MyMachine.Prms.TransitionMode:=INT#0; (* Blending Mode - Corner Radius *)
27 MyMachine.Prms.TransitionParameter[3]:=LREAL#0.25; (* 0.25 mm radius corner blending *)
28
29 MyMachine.Prms.MaxSegmentsPerScan:=INT#2;
30
31 MyMachine.Control.VelocityScaler:=REAL#200.0; (* Override value to permit the machine to run faster than the 'F' feedrates given in the G code data. *)
32
33 MyMachine.Printer.Extruder[0].ScaleFactor:=LREAL#1.0; (* Set default Extruder scaling *)
34 MyMachine.Printer.Extruder[1].ScaleFactor:=LREAL#1.0;
35 MyMachine.Printer.Extruder[2].ScaleFactor:=LREAL#1.0;

```

MyMachine.Feature

Bit	Feature	Description
0	Constant Acceleration During Feedrate Override	<ul style="list-style-type: none"> The default implementation of Feedrate Override uses a firmware function block Y_SetGroupOverride. This mode scales the time of the original move so that accel, decel and velocity are all changed. In addition, the range of the override has a maximum of 100% (could not go faster than the original move). This technique may cause undesirable motion since doubling the speed while keeping the same profile would result in 4 times more aggressive acceleration. With this feature for Constant Acceleration During Feedrate Override enabled, the accel and decel will remain constant and the override is applied only to the velocity. Note that certain move profiles could therefore be accel-limited. When this mode is active, the tool offset commands G41, G42 are NOT supported. Alarm '11082 - Combination Error' will be returned.
1	Include active work offset in Home Preset movement	Used with Emulation mode 2 Home Preset moves such as G1 H3.
2	Jog Increment uses current JogSpeed rather than defaulting to the highest JogSpeed	The speed used for Jog Increment defaults to UserData.Config.JogSpeed[3]. But when this bit is set, the currently selected JogSpeed is used.
3	Variable processing	<ul style="list-style-type: none"> When set to 0, scenario A (default) late processing for simple paths, XYZ & F registers can be variables. Looping through commands using and IF statement is supported. Lookahead, Constant accel feedrate override, and tool compensation G41, G42 are not supported. When set to 1, scenario B, early variable processing is performed. Lookahead, Constant accel feedrate override, and tool compensation G41, G42 are supported, but looping though commands using an IF statement is not.
4	Jog Mode Switching	0 = original behavior 1 = Switch Jog mode to Infinite when changing jog speed Requested by CAMaster esp. for use with Hotkeys note: This bit is not currently used by the toolbox and is for internal Compass reference.
5	Reduced feedrate override reduces Rapid / G0 velocities	0 = (default) G0 / Rapids use MyMachine.Prms.MaxDirectVelocity, feed rate override is not applied. 1 = G0 / Rapids use MyMachine.Prms.MaxDirectVelocity, and if the override is less than 100%, rapid moves will take place at reduced velocity.

The remaining bits in MyMachine.Feature are reserved for future use.



Data Type: MC_PATH_DATA_REF

Data structure used with the [Read_GCode_File](#), [Read_GCode_Stream](#), CP_PathGenerator, and [MC_MovePath](#) function blocks .

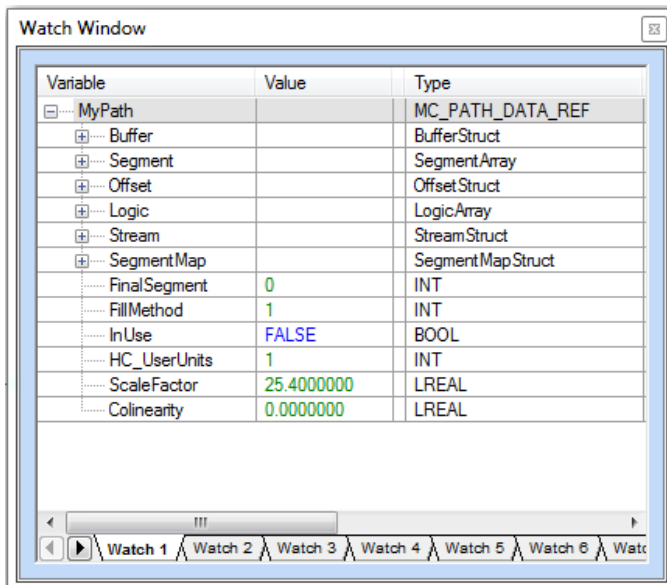
Data Type Declaration

The column that indicates whether the 'U'ser OR the 'C'ontroller (or Toolbox function block) write the data can be misleading for this data type, as there are function blocks available such as [Read_GCode_File](#) which act on behalf of the user to populate many of these values. In all cases, 'C' indicates that the user should not write to the value, as the [MC_MovePath](#) function block will update these elements. 'U' indicates that the User is in some way responsible for populating the data. This is especially important when creating a custom function block to load Segment data.

*	Element	Data Type	Description	Usage
	MyPath	MC_PATH_DATA_REF		
C	Buffer	BufferStruct	Sub structure containing information for managing the circular buffer of Segment data. SegmentArray is defined as having 250 elements by default. Paths requiring an infinite number of segments can be processed due to the circular buffer technique. The core function blocks Read_GCode_File, Read_GCode_Stream and MC_MovePath manage the data in this sub structure. If writing a custom path generator function, it must set MyPathStruct.Buffer.StorePointer as it creates the path.	MyPath.Buffer.StorePointer
U	Segment	SegmentArray	Contains all details required for each of the Segments in the path.	MyPath.Segment[0].X
U	Logic	LogicStruct	For support of SegmentType 'LoopDecision'. Up to 32 loops or jumps can be configured in PathData. <u>Limitation</u> : The entire path data must fit within the defined size of SegmentArray[]. Circular buffering and LoopDecision SegmentTypes can not be supported simultaneously. MC_MovePath monitors MyPathStruct.Buffer.Overwritten and will generate an error.	MyPath.Logic[n].DecisionIndex
C	SegmentMap	SegmentMapStruct	For internal use by MC_MovePath for tracking the relationship between MyPathStruct.Segment[n] and the unique motion segment ID assigned by the Motion Engine. Group parameters 2201 and 2202 are referenced to manage this mapping. This data is managed by MC_MovePath and the user does not need to reference it under normal circumstances.	MyPath.SegmentMap.Map[0][12].PathIndex
C	StreamStatus	StreamStruct	Populated by the Read_GCode_Stream and MC_MovePath function blocks. This structure supplies information required by a source application such as CNCWorks / Compass GCodeComm.DLL which sends path data via Ethernet socket. May also provide useful debugging information when viewed in the MotionWorks IEC watch window.	MyPath.StreamStatus.TCPPacketCount
U	FinalSegment	INT	This value indicates the very last Segment in the path. For example, if the path is loaded 'by hand' in the Initialize POU, set this value. In this case, the StorePointer and UsePointer are not necessary. When Read_GCode_File encounters the end of file, it will set the value to the proper Segment. When an M30 command is detected, this value will be set to the proper Segment.	MyPath.FinalSegment
U	FillMethod	INT	Specify whether the data comes from a File or a Stream. This will affect how MC_MovePath determines when it has reached the end of the data. If the data is being hard coded withing the IEC application, this value must be set to n/a or zero.	MyPath.FillMethod
C	InUse	BOOL	Written by MC_MovePath when the block is executing. This flag is monitored by functions such as Read_GCode_File to provide data integrity. Read_GCode_File will generate an error if upon the rising edge it detects that MC_MovePath is still using the data.	MyPath.InUse

*	Element	Data Type	Description	Usage
	MyPath	MC_PATH_DATA_REF		
U	HC_UserUnits	INT	<p>Stores the value /code of the user units set in the Hardware Configuration. See the Example MotionWorks IEC code below for a solution to read the user units of the X / Y axes of a gantry configuration.</p> <p>0 = inches 1 = millimeters 2 = microns</p> <p>If using another type of Group where it is not possible to read the user unit configuration, simply hard code the proper value into MyPath.HC_UserUnits. This is important if the machine will process G-Code files which contain G20 or G21 commands.</p>	MyPath.HC_UserUnits
U	ScaleFactor	LREAL	<p>If data may come from a source providing user unit positions that differ from that of the Hardware Configuration, this ScaleFactor will convert the positions. See the example below for reading the User Units set in the Hardware Configuration. Read_GCode_File and Read_GCode_Stream can automatically set this value based on HC_UserUnits.</p> <p>If ScaleFactor is zero when either of the G-Code processing blocks are started, it will be initialized to 1.</p> <p>This value can be preset to any ScaleFactor if necessary as long as the G-Code data does not contain G20 or G21.</p>	MyPath.ScaleFactor
U	Colinearity	LREAL	<p>Specify the angle between segments to invoke an 'ExactStopCheck'. For example, when shape cutting, if multiple segments are nearly colinear (less than 5 degrees) and should be blended by inserting a small corner radius, but large angles such as when the tool moves vertically out of the material (90 degrees), a colinearity value of say 65 degrees can be detected, causing motion to stop before raising the tool.</p>	MyPath.Colinearity

Watch Window Views





Data Type: MotionParameters

Sub struct of MachineStruct which holds configuration data used in [MC_MovePath](#), [Read_GCode_File](#), and [Read_GCode_Stream](#).

If the FeedRate is not specified for a PathData.Segment, the MC_MovePath function block will default to using the appropriate maximum velocity from the MachineStruct.Prms.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MachineStruct.Prms	MotionParameter		
U	MaxVelocity	LREAL	<p>Max velocity in Cartesian units/sec for a linear (G1) / Cartesian move. If the feed rate provided in the G-Code data exceeds this value, it will be limited to this value.</p> <p>You must set this value to the maximum safe velocity that the mechanism can tolerate.</p> <p>G Code users: Units are those of the group as configured in the Hardware Configuration!</p> <p>MC_MovePath will never set a velocity higher than this value, even if the MC_MovePath.ScaleFactor input would result in excessive velocity.</p>	MachineStruct.Prms.MaxVelocity
U	MaxDirectVelocity	LREAL	<p>Max velocity in Cartesian units/sec for a direct (G0) / Cartesian move. If the feed rate provided in the G-Code data exceeds this value, it will be limited to this value.</p> <p>You must set this value to the maximum safe velocity that the mechanism can tolerate.</p> <p>G-Code users: Units are those of the group as configured in the Hardware Configuration!</p> <p>MC_MovePath will never set a velocity higher than this value, even if the MC_MovePath.ScaleFactor input would result in excessive velocity.</p>	MachineStruct.Prms.MaxDirectVelocity
U	MaxRotationalVelocity	LREAL	<p>Max velocity in rotational units/sec when a motion segment contains a rotational change but no Cartesian motion.</p> <p>You must set this value to the maximum safe velocity that the mechanism can tolerate.</p> <p>G-Code users: Units are those of the group as configured in the Hardware Configuration!</p> <p>MC_MovePath will never set a velocity higher than this value, even if the MC_MovePath.ScaleFactor input would result in excessive velocity.</p>	MachineStruct.Prms.MaxRotationalVelocity
U	MaxAcceleration	LREAL	<p>Acceleration in Cartesian units/sec² applied to linear (G1) and direct Cartesian motion.</p> <p>You must set this value to the maximum safe acceleration that the mechanism can tolerate.</p> <p>G Code users: Units are those of the group as configured in the Hardware Configuration!</p> <p>MC_MovePath will never set an acceleration higher than this value, even if the MC_MovePath.ScaleFactor input would result in excessive acceleration.</p>	MachineStruct.Prms.Acceleration

*	Element	Data Type	Description	Usage
	MachineStruct.Prms	MotionParameter		
U	MaxDeceleration	LREAL	Deceleration in Cartesian units/sec2 applied to linear (G1) and direct Cartesian motion. You must set this value to the maximum safe acceleration that the mechanism can tolerate. G Code users: Units are those of the group as configured in the Hardware Configuration! MC_MovePath will never set an acceleration higher than this value, even if the MC_MovePath.ScaleFactor input would result in excessive deceleration.	MachineStruct.Prms.Deceleration
U	Acceleration	LREAL	Acceleration in Cartesian units/sec2 applied to linear (G1) and direct Cartesian motion. Typically this can be set to the same value as the MaxAcceleration, however, if MC_MovePath.VelocityScaler is used, the acceleration is also scaled by the square of the velocity, and could become limited to MaxAcceleration.	MachineStruct.Prms.Acceleration
U	Deceleration	LREAL	Deceleration in Cartesian units/sec2 applied to linear (G1) and direct Cartesian motion. Typically this can be set to the same value as the MaxAcceleration, however, if MC_MovePath.VelocityScaler is used, the deceleration is also scaled by the square of the velocity, and could become limited to MaxDeceleration.	MachineStruct.Prms.Deceleration
U	Jerk	LREAL	Supported for remote groups only (MLX and MotomanSync). (mm/sec3)	
U	DirectAcceleration	LREAL	Acceleration in Cartesian units/sec2 applied direct (G0) Cartesian motion. Typically this can be set to the same value as the MaxAcceleration, however, if MC_MovePath.VelocityScaler is used, the acceleration is also scaled by the square of the velocity, and could become limited to MaxAcceleration.	MachineStruct.Prms.DirectAcceleration
U	DirectDeceleration	LREAL	Deceleration in Cartesian units/sec2 applied to direct (G0) Cartesian motion. Typically this can be set to the same value as the MaxAcceleration, however, if MC_MovePath.VelocityScaler is used, the acceleration is also scaled by the square of the velocity, and could become limited to MaxDeceleration.	MachineStruct.Prms.DirectDeceleration
U	DirectJerk	LREAL	Not Supported.	
U	RotationalVelocity	LREAL	Velocity for purely rotational moves on Rx, Ry, or Rz, and when pre aligning a Tangent axis in the units configured for the external tangent axis.	MachineStruct.Prms.RotationalVelocity
U	RotationalAcceleration	LREAL	Acceleration used when pre aligning a Tangent axis in the units configured for the external tangent axis.	MachineStruct.Prms.RotationalAcceleration

*	Element	Data Type	Description	Usage
	MachineStruct.Prms	MotionParameter		
U	MaxSegmentsPerScan	INT	<p>The maximum number of motion functions that will be executed in the same scan. (Similar means Linear, Circular, and Direct.) For example, if MC_PATH_DATA_REF contains 100 linear motion segments with no other command type in between, and MaxSegmentsPerScan is set to 8, it would take 13 scans to load all the motion segments, provided the motion queue can accommodate all the segments, or that they are being consumed at a fast rate.</p> <p>For applications with fewer, longer segments, setting MaxSegmentsPerScan to 1 is adequate. 3 is considered high. Setting this value too high with MC_MovePath is a very fast task may result in a PLC watchdog.</p> <p>See the "Motion Queue Size" in the Hardware Configuration under the Group item in the configuration tree.</p>	MachineStruct.Prms.MaxSegmentsPerScan
U	MinMotionDistance	LREAL	Minimum distance commanded in user units to be recognized as a motion segment on an axis in the group. Default = 0.00001	MachineStruct.Prms.MinMotionDistance
U	TransitionMode	INT	This will be the default Transition mode used as the input to PLCopen Part 4 motion blocks. See Transition Mode in the PLCopen Motion Function Blocks help manual.	MachineStruct.Prms.TransitionMode
U	TransitionParameter	VECTOR	This will be the default TransitionParameter used as the input to PLCopen Part 4 motion blocks. See Transition Mode in the PLCopen Motion Function Blocks help manual.	MachineStruct.Prms.TransitionParameter

Special Notes for G-Code Applications:

All data in the MachineStruct are in the user units of the configured group. For example, if the G-Code data presented to the MPiec is in inches, but the group mechanism is configured for millimeters, the Work Coordinate Offsets must always be entered in millimeters.

Note that even if the same units are provided in the G-Code data, (say millimeters) there is still a difference to be accounted for. Native units on the MPiec controller are mm/sec for velocity, mm/sec² for acceleration. Native G-Code convention specifies feedrate in mm/minute. Take this difference into account when configuring the Maximums in the MachineStruct.

Example

This data is typically initialized in a Warm Start system task.

```
1
2  (*-----*)
3  (*-----*      Basic Machine Configuration and maximums      *-----*)
4  (*-----*)
5  MyMachine.MachineType:=GTB_MachineType#Milling;          (* From Group Toolbox DataTypes:  GTB_MachineTypes:(Milling, Lathe, Printer); *)
6  MyMachine.Variant:=GTB_Variant#na;                    (* From Group Toolbox DataTypes:  GTB_MachineTypes:(Milling, Lathe, Printer); *)
7  MyMachine.Emulation:=GTB_Emulation#mode2;
8
9
10 MyMachine.Prms.MaxVelocity:=LREAL#500.0;              (* In configured group units / sec *)
11 MyMachine.Prms.ZMaxVelocity:=LREAL#200.0;            (* In configured group units / sec.  Only used in MyMachine.Emulation:=GTB_Emulation#mode2; *)
12 MyMachine.Prms.MaxDirectVelocity:=LREAL#2000.0;      (* In configured group units / sec. *)
13 MyMachine.Prms.RotationalVelocity := LREAL#3000.0;   (* For Rz or Tangent axis when aligning to the next required tangent vector *)
14 MyMachine.Prms.MaxRotationalVelocity:=LREAL#15000.0; (* In configured group units / sec *)
15
16 MyMachine.Prms.Acceleration:=LREAL#9800.0;          (* In configured group units / sec2 *)
17 MyMachine.Prms.DirectAcceleration:=LREAL#4000.0;    (* In configured group units / sec2 *)
18 MyMachine.Prms.ZAcceleration:=LREAL#9800.0;        (* In configured group units / sec2.  Only used in MyMachine.Emulation:=GTB_Emulation#mode2; *)
19 MyMachine.Prms.RotationalAcceleration := LREAL#7000.0; (* For Rz or Tangent axis when aligning to the next required tangent vector *)
20 MyMachine.Prms.Deceleration:=LREAL#9800.0;         (* In configured group units / sec2 *)
21 MyMachine.Prms.DirectDeceleration:=LREAL#4000.0;   (* In configured group units / sec2 *)
22 MyMachine.Prms.MaxAcceleration:=LREAL#9800.0 * LREAL#2.0; (* In configured group units / sec2 *)
23 MyMachine.Prms.ZMaxAcceleration:=LREAL#9800.0 * LREAL#2.0; (* In configured group units / sec2.  Only used in MyMachine.Emulation:=GTB_Emulation#mode2; *)
24 MyMachine.Prms.MaxDeceleration:=LREAL#9800.0 * LREAL#2.0; (* In configured group units / sec2 *)
25
26 MyMachine.Prms.TransitionMode:=INT#0;              (* Blending Mode - Corner Radius *)
27 MyMachine.Prms.TransitionParameter[3]:=LREAL#0.25; (* 0.25 mm radius corner blending *)
28
29 MyMachine.Prms.MaxSegmentsPerScan:=INT#2;
30
31 MyMachine.Control.VelocityScaler:=REAL#200.0;      (* Override value to permit the machine to run faster than the 'F' feedrates given in the G code data. *)
32
33 MyMachine.Printer.Extruder[0].ScaleFactor:=LREAL#1.0; (* Set default Extruder scaling *)
34 MyMachine.Printer.Extruder[1].ScaleFactor:=LREAL#1.0;
35 MyMachine.Printer.Extruder[2].ScaleFactor:=LREAL#1.0;
```



Data Type: OffsetStruct

Sub structure of the [MachineStruct](#) which holds data related to the work coordinate offsets. When a G10 command is processed, the [Read_GCode_File](#) or [Read_GCode_Stream](#) function block will populate the appropriate Offset with the data received. Alternatively, Offset data can be loaded from any other source, such as a host PC or HMI. When a G-Code G54 through G59.3 is received, the subsequent position commands will be offset accordingly.

Important: All data in the MachineStruct are in the user units of the configured group. For example, if the G-Code data sent to the MPiec is in inches (G20), but the group mechanism is configured for millimeters, the Work Coordinate Offsets **MUST** be entered in millimeters. When G-Code G10 is received the G-Code processor will convert the offset values to the proper units for the MachineStruct if necessary.

Data Type Declaration

*	Element	Data Type	Description	Usage
	CoordinateSystem	OffsetStruct		
U	Offset	OffsetArray	ARRAY[1..9] OF MC_CARTESIAN_REF.	MyMachine.CoordinateSystem.Offset[2].Rx
U	Selected	INT	The Coordinate System Offset currently used.	MyMachine.CoordinateSystem.Selected



Data Type: PendantDataStruct

For use with the [Pendant_Driver](#) and [GroupCommManager](#) function blocks. The major structure 'PendantDataStruct' contains two sub structures which each contain two sub structures to categorize the data:

1. From the Pendant or PC.
2. To the Pendant or PC.
3. Coils (BOOL)
4. Tags (Non BOOL) values.

Data Type Declaration

*	Element	Data Type	Description	Usage
	PendantData	PendantDataStruct		
C	From	FromPendantStruct	Data received from the pendant.	PendantData.From.Coil.ConfirmHardwarePosition
C	ToP	ToPendantStruct	Data transmitted to the pendant.	PendantData.ToP.Tag.OperationMode

Sub Data Type Declaration - FromPendantStruct

*	Element	Data Type	Description	Usage
	PendantData.From			
C	Coil	PendantCoilStruct	BOOL data received from the pendant.	PendantData.From.Coil.ConfirmHardwarePosition
C	Tag	PendantHoldingRegisterStruct	All other datatypes received from the pendant.	PendantData.From.Tag.OperationMode

Sub Data Type Declaration - ToPendantStruct

*	Element	Data Type	Description	Usage
	PendantData.ToP			
C	Coil	PendantDiscreteInputStruct	BOOL data transmitted to the pendant.	PendantData.ToP.Coil.ConfirmHardwarePosition
C	Tag	PendantInputRegisterStruct	All other datatypes transmitted to the pendant.	PendantData.ToP.Tag.GroupErrorID

Sub Data Type Declaration - PendantCoilStruct (FROM Pendant TO MPiec)

*	Element	Data Type	Byte Offset	Modbus Address	Description
	PendantCoilStruct				
C	ConfirmHardwarePosition	BOOL	800	4X:7901.0	If power down position of the group is different from power up position, user needs to verify position.
C	RestartPLC	BOOL		4X:7901.1	Not supported
C	UserProgramStart	BOOL		4X:7901.2	For use in Play Mode screen
C	BrakeRelease	BOOL		4X:7901.3	Not Supported
C	calculateCurrentTCP	BOOL		4X:7901.4	Not Supported
C	CubicIZCopyTCPToCenterPoint	BOOL		4X:7901.5	Not Supported
C	CubicIZCopyTCPToCorner1	BOOL		4X:7901.6	Not Supported
C	CubicIZCopyTCPToCorner2	BOOL		4X:7901.7	Not Supported
C	CubicIZNameUpdate	BOOL		4X:7901.8	Not Supported
C	CubicIZNumberChanged	BOOL		4X:7901.9	Not Supported
C	CubicIZSaveCenterPointData	BOOL		4X:7901.10	Not Supported
C	CubicIZSaveTwoCornersData	BOOL		4X:7901.11	Not Supported
C	CubicIZSetCenterPoint	BOOL		4X:7901.12	Not Supported
C	CubicIZSetTwoCorners	BOOL		4X:7901.13	Not Supported
C	DecrementTeachPoint	BOOL		4X:7901.14	Used to decrement the index of the teach position
C	getQueuedError	BOOL	4X:7901.15	Not supported	
C	HomeOffsetsGet	BOOL	802	4X:7902.0	Add support for Y_GroupReadVectorParam[2500] for v350
C	HomeOffsetsGetCurrent	BOOL		4X:7902.1	Add support for Y_GroupReadVectorParam[2501] for v350
C	HomeOffsetsSet	BOOL		4X:7902.2	Add support for MC_GroupSetPosition for v350
C	IncrementTeachPoint	BOOL		4X:7902.3	Used to increment the index of the taught position
C	JobNameUpdate	BOOL		4X:7902.4	Not supported
C	JobNumberChanged	BOOL		4X:7902.5	Not supported
C	JogFlags.Axis[1].Positive	BOOL		4X:7902.6	Jog Joint 1 or TCP X in the positive direction
C	JogFlags.Axis[1].Negative	BOOL		4X:7902.7	Jog Joint 1 or TCP X in the negative direction
C	JogFlags.Axis[2].Positive	BOOL		4X:7902.8	Jog Joint 2 or TCP Y in the positive direction
C	JogFlags.Axis[2].Negative	BOOL		4X:7902.9	Jog Joint 2 or TCP Y in the negative direction
C	JogFlags.Axis[3].Positive	BOOL		4X:7902.10	Jog Joint 3 or TCP Z in the positive direction
C	JogFlags.Axis[3].Negative	BOOL		4X:7902.11	Jog Joint 3 or TCP Z in the negative direction
C	JogFlags.Axis[4].Positive	BOOL		4X:7902.12	Jog Joint 4 or TCP Rx in the positive direction
C	JogFlags.Axis[4].Negative	BOOL		4X:7902.13	Jog Joint 4 or TCP Rx in the negative direction
C	JogFlags.Axis[5].Positive	BOOL		4X:7902.14	Jog Joint 5 or TCP Ry in the positive direction
C	JogFlags.Axis[5].Negative	BOOL	4X:7902.15	Jog Joint 5 or TCP Ry in the negative direction	

*	Element	Data Type	Byte Offset	Modbus Address	Description
	PendantCoilStruct				
C	JogFlags.Axis[6].Positive	BOOL	804	4X:7903.0	Jog Joint 6 or TCP Rz in the positive direction
C	JogFlags.Axis[6].Negative	BOOL		4X:7903.1	Jog Joint 6 or TCP Rz in the negative direction
C	JogFlags.Axis[7].Positive	BOOL		4X:7903.2	Jog Joint 7 in the positive direction
C	JogFlags.Axis[7].Negative	BOOL		4X:7903.3	Jog Joint 7 in the negative direction
C	JogToHome	BOOL		4X:7903.4	Used to Jog to the robot's home position (Joint zeros)
C	JogToPoint	BOOL		4X:7903.5	Used to jog to the currently selected taught position
C	JogToUFOOrigin	BOOL		4X:7903.6	Jog to Part frame origin
C	JogToUFXX	BOOL		4X:7903.7	Jog to part frame XX
C	JogToUFXY	BOOL		4X:7903.8	Jog to part frame XY
C	ResetSafetyRelay	BOOL		4X:7903.9	Not supported
C	GroupNumberChanged	BOOL		4X:7903.10	Not supported
C	SendAbort	BOOL		4X:7903.11	Not supported
C	SetTeachPointToolAndPartFrame	BOOL		4X:7903.12	Not supported
C	TeachPointChanged	BOOL		4X:7903.13	Not supported
C	TeachPointNameEntered	BOOL		4X:7903.14	Not supported
C	ToolClear	BOOL	4X:7903.15	Used to deactivate the current tool parameters	
C	ToolNameUpdate	BOOL	806	4X:7904.0	Not supported
C	ToolNumberChanged	BOOL		4X:7904.1	Not supported
C	ToolSave	BOOL		4X:7904.2	Used to save the parameters of a tool to the UserApplicationData structure
C	ToolSet	BOOL		4X:7904.3	Used to activate a tool
C	triggerAbort	BOOL		4X:7904.4	Used to abort group motion
C	triggerPower	BOOL		4X:7904.5	Used to power on the group
C	triggerHold	BOOL		4X:7904.6	MC_GroupInterrupt
C	triggerReset	BOOL		4X:7904.7	Used to reset group alarms. Buffered motions get cleared from memory
C	triggerResetAndHold	BOOL		4X:7904.8	Not Supported
C	triggerRestart	BOOL		4X:7904.9	MC_GroupContinue
C	UndoLastTeachPoint	BOOL		4X:7904.10	Used to invalidate the last selected taught point
C	UpdateSpeedScale	BOOL		4X:7904.11	Not Supported
C	UpdateTeachPointTCPData	BOOL		4X:7904.12	Not Supported
C	PartFrameClear	BOOL		4X:7904.13	Used to deactivate the currently active part frame
C	PartFrameCopyOrigin	BOOL		4X:7904.14	Used to copy the current TCP to Part frame origin variable
C	PartFrameCopyXX	BOOL	4X:7904.15	Used to copy the current TCP to part frame XX variable	

*	Element	Data Type	Byte Offset	Modbus Address	Description
	PendantCoilStruct				
C	PartFrameCopyXY	BOOL	808	4X:7905.0	Used to copy the current TCP to part frame XY variable
C	PartFrameNameUpdate	BOOL		4X:7905.1	Not supported
C	PartFrameNumberChanged	BOOL		4X:7905.2	Not supported
C	PartFrameSave	BOOL		4X:7905.3	Used to write part frame data to the UserApplicationData
C	PartFrameSet	BOOL		4X:7905.4	Used to activate a part frame
C	JoggingMode	BOOL		4X:7905.5	Used to set the group in manual control mode when the group is in simulation operation mode
C	SignalsDisableLimits	BOOL		4X:7905.6	Not supported
C	queuedAlarmDisplayMode	BOOL		4X:7905.7	Not supported
C	R0BrakeReleaseBit0	BOOL		4X:7905.8	Used to release/engage break for joint 0
C	R0BrakeReleaseBit1	BOOL		4X:7905.9	Used to release/engage break for joint 1
C	R0BrakeReleaseBit2	BOOL		4X:7905.10	Used to release/engage break for joint 2
C	R0BrakeReleaseBit3	BOOL		4X:7905.11	Used to release/engage break for joint 3
C	R0BrakeReleaseBit4	BOOL		4X:7905.12	Used to release/engage break for joint 4
C	R0BrakeReleaseBit5	BOOL		4X:7905.13	Used to release/engage break for joint 5
C	R0BrakeReleaseBit6	BOOL		4X:7905.14	Used to release/engage break for joint 6
C	R0BrakeReleaseBit7	BOOL	4X:7905.15	Used to release/engage break for joint 7	
C	R1BrakeReleaseBit0	BOOL	810	4X:7906.0	Not supported
C	R1BrakeReleaseBit1	BOOL		4X:7906.1	Not supported
C	R1BrakeReleaseBit2	BOOL		4X:7906.2	Not supported
C	R1BrakeReleaseBit3	BOOL		4X:7906.3	Not supported
C	R1BrakeReleaseBit4	BOOL		4X:7906.4	Not supported
C	R1BrakeReleaseBit5	BOOL		4X:7906.5	Not supported
C	R1BrakeReleaseBit6	BOOL		4X:7906.6	Not supported
C	R1BrakeReleaseBit7	BOOL		4X:7906.7	Not supported
C	R2BrakeReleaseBit0	BOOL		4X:7906.8	Not supported
C	R2BrakeReleaseBit1	BOOL		4X:7906.9	Not supported
C	R2BrakeReleaseBit2	BOOL		4X:7906.10	Not supported
C	R2BrakeReleaseBit3	BOOL		4X:7906.11	Not supported
C	R2BrakeReleaseBit4	BOOL		4X:7906.12	Not supported
C	R2BrakeReleaseBit5	BOOL		4X:7906.13	Not supported
C	R2BrakeReleaseBit6	BOOL		4X:7906.14	Not supported
C	R2BrakeReleaseBit7	BOOL	4X:7906.15	Not supported	

*	Element	Data Type	Byte Offset	Modbus Address	Description
	PendantCoilStruct				
C	R3BrakeReleaseBit0	BOOL	812	4X:7907.0	Not supported
C	R3BrakeReleaseBit1	BOOL		4X:7907.1	Not supported
C	R3BrakeReleaseBit2	BOOL		4X:7907.2	Not supported
C	R3BrakeReleaseBit3	BOOL		4X:7907.3	Not supported
C	R3BrakeReleaseBit4	BOOL		4X:7907.4	Not supported
C	R3BrakeReleaseBit5	BOOL		4X:7907.5	Not supported
C	R3BrakeReleaseBit6	BOOL		4X:7907.6	Not supported
C	R3BrakeReleaseBit7	BOOL		4X:7907.7	Not supported
C	StepContinuePressed	BOOL		4X:7907.8	Not supported
C	StepPausePressed	BOOL		4X:7907.9	Not supported
C	TeachPoint	BOOL		4X:7907.10	Used to load a point into the UserApplicationData structure
C	UndoVisible	BOOL		4X:7907.11	Not supported
C	InvalidatePartFrame	BOOL		4X:7907.12	De-activate the current tool parameters
C	InvalidateTool	BOOL		4X:7907.13	De-activate the current part frame
C	ReadJobFile	BOOL		4X:7907.14	Read the defined job file from the Mpiec controller's flash memory
C	WriteJobFile	BOOL	4X:7907.15	Write the defined job file to the Mpiec controller's flash memory	
C	ReadPartFrameFile	BOOL	814	4X:7908.0	Read the defined part frame file from the controller's memory
C	WritePartFrameFile	BOOL		4X:7908.1	Write the defined part frame file to the controller's memory
C	ReadToolFile	BOOL		4X:7908.2	Read the defined tool file from the controller's memory
C	WriteToolFile	BOOL		4X:7908.3	Write the defined tool file to the controller's memory
C	AckDisconnect	BOOL		4X:7908.4	Used to acknowledge that a disconnect between the pendant and the Mpiec controller happened
C	AckDisconnectRequired	BOOL		4X:7908.5	Used to let the user know that a disconnect between the pendant and the Mpiec controller happened. The user will have to acknowledge for the pendant to restart communications.
C	GroupEnable	BOOL		4X:7908.6	Used to enable the group
C	DisableControlModeInput	BOOL		4X:7908.7	Not supported
C	ResetFunctionError	BOOL		4X:7908.8	Used to reset certain functionality errors in the pendant driver.
C	GroupDisable	BOOL		4X:7908.9	Used to disable the group
C	ControlMode	BOOL		4X:7908.10	For GUI to have complete control over Auto/Manual, for applications other than robots, which typically have a keyswitch for this.
C				4X:7908.11	
C				4X:7908.12	
C				4X:7908.13	
C				4X:7908.14	
C			4X:7908.15		

*	Element	Data Type	Byte Offset	Modbus Address	Description
	PendantCoilStruct				
C	Move To Origin Request	BOOL	816	4X:7909.0	Execute GroupToHome FB.
C		BOOL		4X:7909.1	
C		BOOL		4X:7909.2	
C		BOOL		4X:7909.3	
C		BOOL		4X:7909.4	
C		BOOL		4X:7909.5	
C		BOOL		4X:7909.6	
C		BOOL		4X:7909.7	
C		BOOL		4X:7909.8	
C		BOOL		4X:7909.9	
C		BOOL		4X:7909.10	
C		BOOL		4X:7909.11	
C		BOOL		4X:7909.12	
C		BOOL		4X:7909.13	
C				4X:7909.14	
C				4X:7909.15	
C					

Sub Data Type Declaration - PendantHoldingRegisterStruct FROM Pendant TO MPiec (FC16)

*	Element	Data Type	Byte Offset	Modbus Address	Description
	PendantDiscreteInputStruct				
C	TeachPoints	DINT	0	DWS:7501	Number of supported TeachPoints
C	PartFrames	DINT	4	DWS:7503	Number of supported Part frames
C	CubicIZDataCoordFrame	DINT	8	DWS:7505	Not Supported
C	CubicIZDataCubicIZType	DINT	12	DWS:7507	Not Supported
C	CubicIZDataZoneAction	DINT	16	DWS:7509	Not Supported
C	CubicIZNumber	DINT	20	DWS:7511	Not Supported
C	Reserved[0]	REAL	24	DWS:7513	Not Supported
C	Reserved[1]	REAL	28	DWS:7515	
C	Reserved[2]	REAL	32	DWS:7517	
C	Reserved[3]	REAL	36	DWS:7519	
C	Reserved[4]	REAL	40	DWS:7521	
C	Reserved[5]	REAL	44	DWS:7523	
C	Reserved[6]	REAL	48	DWS:7525	
C	JobNumber	DINT	52	DWS:7527	Current Job index
C	JoggingCoordinateMode	DINT	56	DWS:7529	Currently selected coordinate frame
C	JogSpeedIndex	DINT	60	DWS:7531	Currently chosen Jog speed Index
C	HomeMode	DINT	64	DWS:7533	Mode of setting home in MC_GroupSetPosition (was group number)
C	TeachPointNumber	DINT	68	DWS:7535	Current teach point index
C	ToolNumber	DINT	72	DWS:7537	Current tool index
C	PartFrameOptionalParameters	DINT	76	DWS:7539	Not Supported
C	PartFrameNumber	DINT	80	DWS:7541	Current Part frame index
C	InternalData_GlobalSpeedScale	REAL	84	FPS:7543	Not Supported
C	CubicIZDataCenterPoint0	REAL	88	FPS:7545	Not Supported
C	CubicIZDataCenterPoint1	REAL	92	FPS:7547	Not Supported
C	CubicIZDataCenterPoint2	REAL	96	FPS:7549	Not Supported
C	CubicIZDataCenterPoint3	REAL	100	FPS:7551	Not Supported
C	CubicIZDataCenterPoint4	REAL	104	FPS:7553	Not Supported
C	CubicIZDataCenterPoint5	REAL	108	FPS:7555	Not Supported
C	CubicIZDataCorner1_0	REAL	112	FPS:7557	Not Supported
C	CubicIZDataCorner1_1	REAL	116	FPS:7559	Not Supported
C	CubicIZDataCorner1_2	REAL	120	FPS:7561	Not Supported
C	CubicIZDataCorner2_0	REAL	124	FPS:7563	Not Supported
C	CubicIZDataCorner2_1	REAL	128	FPS:7565	Not Supported
C	CubicIZDataCorner2_2	REAL	132	FPS:7567	Not Supported
C	CubicIZDataDimensions0	REAL	136	FPS:7569	Not Supported
C	CubicIZDataDimensions1	REAL	140	FPS:7571	Not Supported
C	CubicIZDataDimensions2	REAL	144	FPS:7573	Not Supported
C	KeyswitchStatus	UINT	148	FPS:7575	Added 4/3/18. keyswitch status if it is from modbus - 0 = play, 1 = teach
C	Reserved[0]	INT	150	FPS:7576	Not Supported
C	Reserved[0]	REAL	152	FPS:7577	Not Supported
C	Reserved[0]	REAL	156	FPS:7579	Not Supported
C	Reserved[0]	REAL	160	FPS:7581	Not Supported
C	Reserved[0]	REAL	164	FPS:7583	Not Supported

*	Element	Data Type	Byte Offset	Modbus Address	Description
	PendantDiscreteInputStruct				
C	Reserved[0]	REAL	168	FPS:7585	Not Supported
C	JoggingSpeeds0	REAL	172	FPS:7587	Not Supported
C	JoggingSpeeds1	REAL	176	FPS:7589	Not Supported
C	JoggingSpeeds2	REAL	180	FPS:7591	Not Supported
C	JoggingSpeeds3	REAL	184	FPS:7593	Not Supported
C	BasePose[0]	REAL	188	FPS:7595	Not Supported
C	BasePose[0]	REAL	192	FPS:7597	Not Supported
C	BasePose[0]	REAL	196	FPS:7599	Not Supported
C	BasePose[0]	REAL	200	FPS:7601	Not Supported
C	BasePose[0]	REAL	204	FPS:7603	Not Supported
C	BasePose[0]	REAL	208	FPS:7605	Not Supported
C	Axes0_ConfigData_MaxLimit	REAL	212	FPS:7607	Not Supported
C	Axes0_ConfigData_MinLimit	REAL	216	FPS:7609	Not Supported
C	Axes1_ConfigData_MaxLimit	REAL	220	FPS:7611	Not Supported
C	Axes1_ConfigData_MinLimit	REAL	224	FPS:7613	Not Supported
C	Axes2_ConfigData_MaxLimit	REAL	228	FPS:7615	Not Supported
C	Axes2_ConfigData_MinLimit	REAL	232	FPS:7617	Not Supported
C	Axes3_ConfigData_MaxLimit	REAL	236	FPS:7619	Not Supported
C	Axes3_ConfigData_MinLimit	REAL	240	FPS:7621	Not Supported
C	Axes4_ConfigData_MaxLimit	REAL	244	FPS:7623	Not Supported
C	Axes4_ConfigData_MinLimit	REAL	248	FPS:7625	Not Supported
C	Axes5_ConfigData_MaxLimit	REAL	252	FPS:7627	Not Supported
C	Axes5_ConfigData_MinLimit	REAL	256	FPS:7629	Not Supported
C	Axes6_ConfigData_MaxLimit	REAL	260	FPS:7631	Not Supported
C	Axes6_ConfigData_MinLimit	REAL	264	FPS:7633	Not Supported
C	ToolDataCenterOfGravity_0	REAL	268	FPS:7635	Tool Center of gravity (Xg) mm
C	ToolDataCenterOfGravity_1	REAL	272	FPS:7637	Tool Center of gravity (Yg) mm
C	ToolDataCenterOfGravity_2	REAL	276	FPS:7639	Tool Center of gravity (Zg) mm
C	ToolDataMass	REAL	280	FPS:7641	Tool Mass kg
C	ToolDataMomentOfInertia_0	REAL	284	FPS:7643	Tool moment of Inertia (Ix) kgm2
C	ToolDataMomentOfInertia_1	REAL	288	FPS:7645	Tool moment of Inertia (Iy) kgm2
C	ToolDataMomentOfInertia_2	REAL	292	FPS:7647	Tool moment of Inertia (Iz) kgm2
C	ToolOffset[0]	REAL	296	FPS:7649	Tool Offset in X direction
C	ToolOffset[1]	REAL	300	FPS:7651	Tool Offset in Y direction
C	ToolOffset[2]	REAL	304	FPS:7653	Tool Offset in Z direction
C	ToolOffset[3]	REAL	308	FPS:7655	Tool Offset in Rx direction
C	ToolOffset[4]	REAL	312	FPS:7657	Tool Offset in Ry direction
C	ToolOffset[5]	REAL	316	FPS:7659	Tool Offset in Rz direction
C	PF_Origin0	REAL	320	FPS:7661	Part frame origin X Coordinate
C	PF_Origin1	REAL	324	FPS:7663	Part frame origin Y Coordinate
C	PF_Origin2	REAL	328	FPS:7665	Part frame origin Z Coordinate
C	PF_XX0	REAL	332	FPS:7667	Part frame XX directional point's X Coordinate
C	PF_XX1	REAL	336	FPS:7669	Part frame XX directional point's Y Coordinate
C	PF_XX2	REAL	340	FPS:7671	Part frame XX directional point's Z Coordinate
C	PF_XY0	REAL	344	FPS:7673	Part frame XY directional point's X Coordinate
C	PF_XY1	REAL	348	FPS:7675	Part frame XY directional point's Y Coordinate
C	PF_XY2	REAL	352	FPS:7677	Part frame XY directional point's Z Coordinate

*	Element	Data Type	Byte Offset	Modbus Address	Description
	PendantDiscreteInputStruct				
C	CubicIZ0_RobotActiveStatus0	DINT	356	DWS:7679	Not Supported
C	CubicIZ0_RobotActiveStatus1	DINT	360	DWS:7681	Not Supported
C	TeachPoint[0]	REAL	364	FPS:7683	Taught Point's X Coordinate
C	TeachPoint[0]	REAL	368	FPS:7685	Taught Point's Y Coordinate
C	TeachPoint[0]	REAL	372	FPS:7687	Taught Point's Z Coordinate
C	TeachPoint[0]	REAL	376	FPS:7689	Taught Point's Rx Coordinate
C	TeachPoint[0]	REAL	380	FPS:7691	Taught Point's Ry Coordinate
C	TeachPoint[0]	REAL	384	FPS:7693	Taught Point's Rz Coordinate
C	HomeSequence	UINT	388	FPS:7695	Used with GroupToHome to select which homing sequence to use.
C	NameStringArray[1]	STRING32	400	ST:7701:32	Group name
C	NameStringArray[2]	STRING32	432	ST:7717:32	Not Supported
C	NameStringArray[3]	STRING32	464	ST:7733:32	Job Name
C	NameStringArray[4]	STRING32	496	ST:7749:32	Teach Point Name
C	NameStringArray[5]	STRING32	528	ST:7765:32	Tool Name
C	NameStringArray[6]	STRING32	560	ST:7781:32	Part Frame Name
C	NameStringArray[7]	STRING32	592	ST:7797:32	Name of Part frame set (used to read /write part frame file to the controller's flash memory.
C	NameStringArray[8]	STRING32	624	ST:7813:32	Name of tool set (used to read /write tool file to the controller's flash memory.
C	HeartBeat	DINT	700	DWS:7851	Pendant heartbeat
C	PendantType	DINT	704	DWS:7853	Code to identify the Pendant type (0=default, unused, 1=GroupComm DLL, 2=indusoft, 3=global pendant, others?)
C	VelFactor	REAL	708	DWS:7855	VelFactor for MC_GroupSetOverride 0.0 to 1.0
C	HomeOffset[0]	LREAL	712	DWS:7857	Add support Y_GroupSetPosition and Y_GroupReadVectorParameter. Use for remote groups
C	HomeOffset[1]	LREAL	720	DWS:7861	
C	HomeOffset[2]	LREAL	728	DWS:7865	
C	HomeOffset[3]	LREAL	736	DWS:7869	
C	HomeOffset[4]	LREAL	744	DWS:7873	
C	HomeOffset[5]	LREAL	752	DWS:7877	
C	HomeOffset[6]	LREAL	760	DWS:7881	

Sub Data Type Declaration - PendantDiscreteInputStruct (FROM MPiec TO Pendant)

*	Element	Data Type	Byte Offset	Modbus Address	Description
	PendantDiscreteInputStruct				
C	IDEITriggerInstructionError	BOOL	400	3X:7901.0	Not Supported
C	ServoOnReadyStatus	BOOL		3X:7901.1	Added 4/3/18. FALSE = not servo on ready, TRUE = servo on ready
C	SignalsAllDrivesPowered	BOOL		3X:7901.2	TRUE if all drives in the group are energized
C	SignalsDeadmanDisengaged	BOOL		3X:7901.3	TRUE if the Deadman/Liveman switch on the pendant is preventing the group from getting energized/moving
C	SignalsEStop1Pressed	BOOL		3X:7901.4	TRUE if an E-Stop is preventing the group from getting energized/moving
C	SignalsInitializationComplete	BOOL		3X:7901.5	Not Supported
C	ManualMode	BOOL		3X:7901.6	TRUE = manual, FALSE=auto control mode
C	SignalsGroupActive	BOOL		3X:7901.7	TRUE if the group is enabled (Group.Status.Active = TRUE)
C	PendantConnected	BOOL		3X:7901.8	TRUE if the pendant heartbeat is healthy
C	SignalsSafetyCircuitOK	BOOL		3X:7901.9	TRUE if the Safety Circuit for the group is preventing the group from getting energized /moving.
C	UserProgramStepContinue	BOOL		3X:7901.10	Not Supported
C	AtHomePositionIndicator	BOOL		3X:7901.11	Not Supported
C	AtTaughtPositionIndicator	BOOL		3X:7901.12	Not Supported
C	DisplayStartButton	BOOL		3X:7901.13	Not Supported
C	RobotJogging	BOOL		3X:7901.14	Not Supported
C	TCPJogMode	BOOL		3X:7901.15	Not Supported
C	TeachPointValid	BOOL	402	3X:7902.0	TRUE if the current teach point index in a job contains valid data
C	PartFrameValid	BOOL		3X:7902.1	TRUE if the current part frame index contains valid data
C	ToolValid	BOOL		3X:7902.2	TRUE if the current tool index contains valid data
C	RemoteGroup	BOOL		3X:7902.3	TRUE if the group is a remotely hosted group. Eg Groups controlled by the MLX are considered remote groups.
C	SignalsGuardCircuitOpen	BOOL		3X:7902.4	TRUE if the guard circuit is preventing the group from getting energized/moving
C	ControlInputsValid	BOOL		3X:7902.5	TRUE if the control inputs of the group are valid.
C	GroupSelected	BOOL		3X:7902.6	TRUE if the group name entered in the pendant is valid and matches with the AXES_GROUP_REF on the PendantDriver function block.

Sub Data Type Declaration - PendantInputRegisterStruct (FROM MPiec TO Pendant)

*	Element	Data Type	Byte Offset	Modbus Address	Description
	FromPendantStruct				
C	CubicIZs	DINT	0	3X:7501	Number of cubic Interference Zones supported
C	TeachPoints	DINT	4	3X:7503	Number of Teach Points supported
C	Tools	DINT	8	3X:7505	Number of Tools supported
C	PLCControllerType	INT	12	3X:7507	any controller = 0, MPiec = 1, std = 02 / reserved / reserved / reserved
C	RemoteControllerType	INT	14	3X:7508	local = 0, MLX = 1, MotomanSync = 02 / reserved / reserved / reserved
C	MemoryMapVersion	DWORD	16	3X:7509	Each byte as HEX = Major / Minor / Build / Build - example [03 05 00 01]
C	PendantDriverVersion	DWORD	20	3X:7511	Each byte as HEX = Major / Minor / Build / Build - example [03 05 00 03]
C	available	DINT	24	3X:7513	available, unused
C	NumberOfQueuedErrors	DINT	28	3X:7515	Not Supported
C	Robots	DINT	32	3X:7517	Not Supported
C	OperationMode	DINT	36	3X:7519	Hardware mode if OperationMode = 1. Simulation Mode if Operation Mode = 0
C	ControllerAlarmID	DINT	40	3X:7521	added 2/9/18
C	AxisErrorClass	UINT	44	3X:7523	added 2/9/18, reduced to 16 bit 2/16/18
C	AxisErrorID	UINT	46	3X:7524	added 2/9/18, reduced to 16 bit 2/16/18
C	GroupErrorSubCode	UDINT	48	3X:7525	ErrorSubCode from remote groups
C	FunctionErrorID	UINT	52	3X:7527	Error from functions like Jog, set part frame, etc in the pendant driver function block. 2/9/18 moved from 3X:7671.
C	AxisWarningID	UINT	54	3X:7528	added 2/9/18
C	SystemErrorCode	UINT	56	3X:7529	GroupErrorID
C	GroupErrorClass	UINT	58	3X:7530	GroupErrorClass
C	SystemState	DINT	60	3X:7531	Group's System State

*	Element	Data Type	Byte Offset	Modbus Address	Description
	FromPendantStruct				
C	SystemStatus	DWORD	64	3X:7533	Reflection of MC_GroupReadStatus (bit coding reflects VAR_OUTPUT order) 0 = GroupMoving 1 = GroupHoming 2 = GroupErrorStop 3 = GroupStandby 4 = GroupStopping 5 = GroupDisabled 6 = ConstantVelocity 7 = Accelerating 8 = Decelerating 9 = InPosition 10 = StandStill 11 = Interrupted 12 = SynchronizedMotion 13 = DiscreteMotion 14 = ContinuousMotion
C	IZStatus	DINT	68	3X:7535	Not Supported
C	queuedErrorDetailerrorNumber	DINT	72	3X:7537	Not Supported
C	RobotNumberOfAxes	DINT	76	3X:7539	added to pendant driver 2/13/18
C	Position[0]	REAL	80	FP3:7541	Axis 0 Joint Position
C	Position[1]	REAL	84	FP3:7543	Axis 1 Joint Position
C	Position[2]	REAL	88	FP3:7545	Axis 2 Joint Position
C	Position[3]	REAL	92	FP3:7547	Axis 3 Joint Position
C	Position[4]	REAL	96	FP3:7549	Axis 4 Joint Position
C	Position[5]	REAL	100	FP3:7551	Axis 5 Joint Position
C	Position[6]	REAL	104	FP3:7553	Axis 6 Joint Position
C	currentTCPInJF[0]	REAL	108	FP3:7555	TCP X Coordinate in currently selected frame
C	currentTCPInJF[1]	REAL	112	FP3:7557	TCP Y Coordinate in currently selected frame
C	currentTCPInJF[2]	REAL	116	FP3:7559	TCP Z Coordinate in currently selected frame
C	currentTCPInJF[3]	REAL	120	FP3:7561	TCP Rx Coordinate in currently selected frame
C	currentTCPInJF[4]	REAL	124	FP3:7563	TCP Ry Coordinate in currently selected frame
C	currentTCPInJF[5]	REAL	128	FP3:7565	TCP Rz Coordinate in currently selected frame
C	currentTCP[0]	REAL	132	FP3:7567	Not Supported
C	currentTCP[1]	REAL	136	FP3:7569	Not Supported
C	currentTCP[2]	REAL	140	FP3:7571	Not Supported
C	currentTCP[3]	REAL	144	FP3:7573	Not Supported
C	currentTCP[4]	REAL	148	FP3:7575	Not Supported
C	currentTCP[5]	REAL	152	FP3:7577	Not Supported

*	Element	Data Type	Byte Offset	Modbus Address	Description
	FromPendantStruct				
C	MaxAngularAccel	REAL	156	FP3:7579	Group's Maximum Angular Acceleration read from the AXES_GROUP_REF structure
C	MaxAngularJerk	REAL	160	FP3:7581	Group's Maximum Angular Jerk read from the AXES_GROUP_REF structure
C	MaxAngularSpeed	REAL	164	FP3:7583	Group's Maximum Angular Speed read from the AXES_GROUP_REF structure
C	MaxLinearAccel	REAL	168	FP3:7585	Group's Maximum Linear Acceleration read from the AXES_GROUP_REF structure
C	MaxLinearJerk	REAL	172	FP3:7587	Group's Maximum Linear Jerk read from the AXES_GROUP_REF structure
C	MaxLinearSpeed	REAL	176	FP3:7589	Group's Maximum Linear Speed read from the AXES_GROUP_REF structure
C	ReservedTeachPoint[0]	REAL	180	FP3:7591	Not Supported
C	ReservedTeachPoint[1]	REAL	184	FP3:7593	Not Supported
C	ReservedTeachPoint[2]	REAL	188	FP3:7595	Not Supported
C	ReservedTeachPoint[3]	REAL	192	FP3:7597	Not Supported
C	ReservedTeachPoint[4]	REAL	196	FP3:7599	Not Supported
C	ReservedTeachPoint[5]	REAL	200	FP3:7601	Not Supported
C	ActiveTool	INT	204	3X:7603	Currently active tool index
C	ActivePartFrame	INT	206	3X:7604	Currently active part frame index
C	errorNumber	INT	208	3X:7605	Not Supported
C	queuedErrorNumber	INT	210	3X:7606	Not Supported
C	TeachPointTool	INT	212	3X:7607	Tool Number corresponding to the current taught point
C	TeachPointPF	INT	214	3X:7608	Part frame corresponding to the current taught point
C	CubicIZ1_GroupActiveStatus0	INT	216	3X:7609	Not Supported
C	CubicIZ1_GroupActiveStatus1	INT	218	3X:7610	Not Supported
C	CubicIZ2_GroupActiveStatus0	INT	220	3X:7611	Not Supported
C	CubicIZ2_GroupActiveStatus1	INT	222	3X:7612	Not Supported
C	CubicIZ3_GroupActiveStatus0	INT	224	3X:7613	Not Supported
C	CubicIZ3_GroupActiveStatus1	INT	226	3X:7614	Not Supported
C	CubicIZ4_GroupActiveStatus0	INT	228	3X:7615	Not Supported
C	CubicIZ4_GroupActiveStatus1	INT	230	3X:7616	Not Supported

*	Element	Data Type	Byte Offset	Modbus Address	Description
	FromPendantStruct				
C	CubicI25_GroupActiveStatus0	INT	232	3X:7617	Not Supported
C	CubicI25_GroupActiveStatus1	INT	234	3X:7618	Not Supported
C	CubicI26_GroupActiveStatus0	INT	236	3X:7619	Not Supported
C	CubicI26_GroupActiveStatus1	INT	238	3X:7620	Not Supported
C	CubicI27_GroupActiveStatus0	INT	240	3X:7621	Not Supported
C	CubicI27_GroupActiveStatus1	INT	242	3X:7622	Not Supported
C	CubicI28_GroupActiveStatus0	INT	244	3X:7623	Not Supported
C	CubicI28_GroupActiveStatus1	INT	246	3X:7624	Not Supported
C	CubicI29_GroupActiveStatus0	INT	248	3X:7625	Not Supported
C	CubicI29_GroupActiveStatus1	INT	250	3X:7626	Not Supported
C	CubicI210_GroupActiveStatus0	INT	252	3X:7627	Not Supported
C	CubicI210_GroupActiveStatus1	INT	254	3X:7628	Not Supported
C	CubicI211_GroupActiveStatus0	INT	256	3X:7629	Not Supported
C	CubicI211_GroupActiveStatus1	INT	258	3X:7630	Not Supported
C	CubicI212_GroupActiveStatus0	INT	260	3X:7631	Not Supported
C	CubicI212_GroupActiveStatus1	INT	262	3X:7632	Not Supported
C	CubicI213_GroupActiveStatus0	INT	264	3X:7633	Not Supported
C	CubicI213_GroupActiveStatus1	INT	266	3X:7634	Not Supported
C	CubicI214_GroupActiveStatus0	INT	268	3X:7635	Not Supported
C	CubicI214_GroupActiveStatus1	INT	270	3X:7636	Not Supported
C	CubicI215_GroupActiveStatus0	INT	272	3X:7637	Not Supported
C	CubicI215_GroupActiveStatus1	INT	274	3X:7638	Not Supported
C	CubicI216_GroupActiveStatus0	INT	276	3X:7639	Not Supported
C	CubicI216_GroupActiveStatus1	INT	278	3X:7640	Not Supported
C	CubicI217_GroupActiveStatus0	INT	280	3X:7641	Not Supported
C	CubicI217_GroupActiveStatus1	INT	282	3X:7642	Not Supported
C	CubicI218_GroupActiveStatus0	INT	284	3X:7643	Not Supported
C	CubicI218_GroupActiveStatus1	INT	286	3X:7644	Not Supported
C	CubicI219_GroupActiveStatus0	INT	288	3X:7645	Not Supported
C	CubicI219_GroupActiveStatus1	INT	290	3X:7646	Not Supported
C	CubicI220_GroupActiveStatus0	INT	292	3X:7647	Not Supported
C	CubicI220_GroupActiveStatus1	INT	294	3X:7648	Not Supported
C	CubicI221_GroupActiveStatus0	INT	296	3X:7649	Not Supported
C	CubicI221_GroupActiveStatus1	INT	298	3X:7650	Not Supported
C	CubicI222_GroupActiveStatus0	INT	300	3X:7651	Not Supported
C	CubicI222_GroupActiveStatus1	INT	302	3X:7652	Not Supported
C	CubicI223_GroupActiveStatus0	INT	304	3X:7653	Not Supported
C	CubicI223_GroupActiveStatus1	INT	306	3X:7654	Not Supported
C	CubicI224_GroupActiveStatus0	INT	308	3X:7655	Not Supported
C	CubicI224_GroupActiveStatus1	INT	310	3X:7656	Not Supported
C	CubicI225_GroupActiveStatus0	INT	312	3X:7657	Not Supported
C	CubicI225_GroupActiveStatus1	INT	314	3X:7658	Not Supported
C	CubicI226_GroupActiveStatus0	INT	316	3X:7659	Not Supported
C	CubicI226_GroupActiveStatus1	INT	318	3X:7660	Not Supported
C	CubicI227_GroupActiveStatus0	INT	320	3X:7661	Not Supported
C	CubicI227_GroupActiveStatus1	INT	322	3X:7662	Not Supported
C	CubicI228_GroupActiveStatus0	INT	324	3X:7663	Not Supported

*	Element	Data Type	Byte Offset	Modbus Address	Description
	FromPendantStruct				
C	CubicIZ28_GroupActiveStatus1	INT	326	3X:7664	Not Supported
C	CubicIZ29_GroupActiveStatus0	INT	328	3X:7665	Not Supported
C	CubicIZ29_GroupActiveStatus1	INT	330	3X:7666	Not Supported
C	CubicIZ30_GroupActiveStatus0	INT	332	3X:7667	Not Supported
C	CubicIZ30_GroupActiveStatus1	INT	334	3X:7668	Not Supported
C	CubicIZ31_GroupActiveStatus0	INT	336	3X:7669	Not Supported
C	CubicIZ31_GroupActiveStatus1	INT	338	3X:7670	Not Supported
C	UserStatus	WORD	340	3X:7671	User customizable status bits to pass from their IEC application to a host App



Data Type: PlaneResultStruct

For use with the PlaneMeasurement function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	GridResults	GridResultsStruct		
C	Offset	MC_CARTESIAN_REF	Offset which can be used directly with Y_GroupSetFrameOffset.	PlaneResults.Offset
C	Measurement	PlaneResultArray	ARRAY[0..2] OF MeasurementResultStruct	PlaneResults.Measurement[1].Position[1]

MeasurementResultStruct sub structure

*	Element	Data Type	Description	Usage
	GridResults.Point	BufferStruct		
C	Position	MC_CARTESIAN_REF		PlaneResults.Measurement.Position
C	MeasuredOK	BOOL	Status flag which indicates if the measurement was successful.	PlaneResults.Measurement.MeasuredOK

Data Type: PlaneSetupStruct



For use with the [PlaneMeasurement](#) function block.

Data Type Declaration

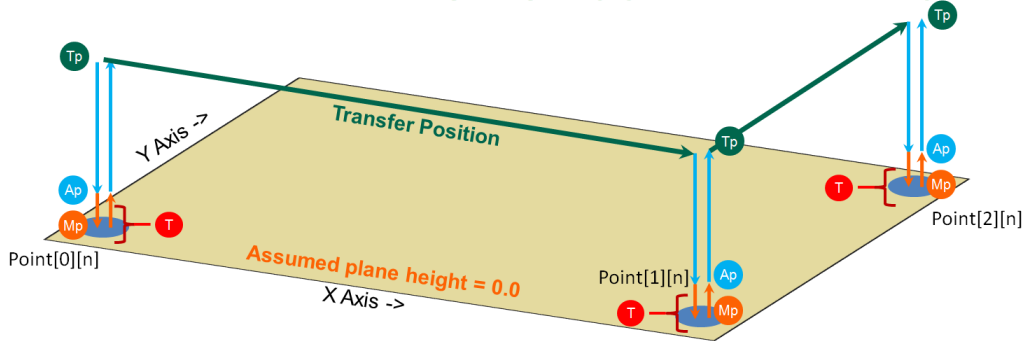
*	Element	Data Type	Description	Usage
	GridSetup			
U	MeasureMethod	GTB_Meas- urementType	Only Torque sensing is supported. In the future, select torque, digital sensor, or LVDT.	PlaneSetup.MeasureMethod
U	MeasuredPlane	YTB_STRING8	Provide the name as specified in the Hardware Configuration -> Group -> Label column for the plane to be measured.	PlaneSetup.MeasuredPlane
U	Point	TouchPointArray	ARRAY [0..2] OF VECTOR. Specify the XY locations of the three locations on the plane to be measured.	PlaneResults.Point[1]
U	TransferVelocity	LREAL	Velocity used when moving between measurement locations at the Transfer position.	PlaneResults.TransferVelocity
U	ApproachVelocity	LREAL	Velocity used when moving between the Transfer position and the Approach position.	PlaneResults.ApproachVelocity
U	MeasurementVelocity	LREAL	Velocity used when moving between the Approach position and the Measurement position.	PlaneResults.MeasurementVelocity
U	Acceleration	LREAL	Acceleration used with all moves in the measurement process.	PlaneResults.Acceleration
U	TransferPosition	LREAL	The position away from the grid to be measured at which the TCP moves to the various measurement points.	PlaneResults.TransferPosition
U	ApproachPosition	LREAL	The position away from the grid to be measured at which the velocity may be reduced and the torque limited in the direction of measurement to prepare for contact with the surface.	PlaneResults.ApproachPosition
U	Tolerance	LREAL	Each measurement must be within the Tolerance of the expected MeasurementPosition, or an Error will be generated. This is to ensure that valid measurements are being recorded and the MeasurementTorqueLimit provided is appropriate.	PlaneResults.Tolerance
U	MeasurementTorqueLimit	DINT	This value is written to ServoPack pn 402 or 403 depending on the direction of the measurements to be taken. Units are percentage of rated torque. It is important to select a torque limit which will not damage the equipment, but allow accurate position reading to be recorded. Ideally, this value is just above the level of torque required to move the axis under normal conditions at the MeasurementVelocity by 1.5 or 2 times.	PlaneResults.MeasurementTorqueLimit
U	ExternalPlane	GTB_MotorArray	If the group contains a virtual axis for the joint required to measure the grid, the GridMeasurement function block will search the ExternalPlane structure to determine the real servo(s) to set the torque limit and read the position. If the group does not contain a non servo axis on the joint in question, this structure is not used.	PlaneResults.ExternalPlane

Example Initialization

```

214 PlaneSetup.MeasureMethod:=GTB_MeasurementType#Torque; (* Use torque limited touch mode *)
215 PlaneSetup.MeasuredPlane:=1; (* Measure the plane which the 2 axis contacts *)
216 PlaneSetup.Tolerance:=LREAL#0.5; (* Measurement position must be within this value of the PlaneSetup.Point[n][3] for the plane axis *)
217
218 PlaneSetup.Acceleration:=MyMachine.Prms.Acceleration;
219
220 PlaneSetup.Point[0][1]:=LREAL#0.0; (* Coordinate for first measurement *)
221 PlaneSetup.Point[0][2]:=LREAL#0.0;
222 PlaneSetup.Point[0][3]:=LREAL#0.0;
223
224 PlaneSetup.Point[1][1]:=LREAL#0.0; (* Coordinate for second measurement *)
225 PlaneSetup.Point[1][2]:=LREAL#60.0;
226 PlaneSetup.Point[1][3]:=LREAL#0.0;
227
228 PlaneSetup.Point[2][1]:=LREAL#30.0; (* Coordinate for third measurement *)
229 PlaneSetup.Point[2][2]:=LREAL#60.0;
230 PlaneSetup.Point[2][3]:=LREAL#0.0;
231
232 PlaneSetup.TransferPosition:=LREAL#20.0; (* Position above plane where the tool moves from one measurement location to the next location *)
233 PlaneSetup.TransferVelocity:=LREAL#200.0;
234
235 PlaneSetup.ApproachPosition:=LREAL#4.0; (* Position at which the torque limit is applied in preparation for measuring the plane *)
236 PlaneSetup.ApproachVelocity:=LREAL#25.0;
237
238 PlaneSetup.MeasurementVelocity:=LREAL#1.0;
239 PlaneSetup.MeasurementTorqueLimit:=DINT#10; (* This is the torque limit in percentage of rated torque for the servo motor *)
240
241 PlaneSetup.ExternalPlane[0].AxisNum:=UINT#8; (* Only necessary if the group uses a virtual axis for the measured plane axis *)

```





Data Type: PrinterStruct

Contains information for 3D printing applications, including externally operated devices, such as heaters, and fans. The functions in the Group Toolbox do not directly manage temperature and fan control features. Support for these features must be added to the MotionWorks IEC application based on the machines specific hardware. The [Read_GCode_File](#) and [Read_GCode_Stream](#) function block will recognize various M Codes for temperature and fan and copy the parameter values into the PrinterStruct for the MotionWorks IEC project to reference.

Data Type Declaration (0120190110)

*	Element	Data Type	Description	Usage
	PrinterStruct			
U	BedTempActual	YTB_RealArray32	The user application can write the actual bed temperatures back to this structure if desired, but they are not used by any function in the Group Toolbox. These values along with the entire MachineStruct are sent to the host application such as CNCWorks once per second for display purposes.	MyMachine.Printer.BedTempActual[x]
U	BedTempSetting	YTB_RealArray32	When receiving an M140 command, bed temperatures are stored in this structure for use by the MotionWorks IEC application program.	MyMachine.Printer.BedTempSetting[x]
U	BedTempTolerance	REAL	The PC software such as CNCWorks, the MotionWorks IEC Application Program, or M190 code can set the minimum temperature to wait for when using M190.	MyMachine.Printer.BedTempMinTarget
U	EnclosureTempActual	REAL	The MotionWorks IEC project must be customized by connecting the actual temperature of the Enclosure to this variable, which will be communicated to the GroupComm DLL and the PC software.	MyMachine.Printer.EnclosureTempActual
U	EnclosureTempSetting	REAL	When receiving an M190 command, the S register value will be stored in this variable. The MotionWorks IEC project must be customized to use this data to control the Enclosure temperature.	MyMachine.Printer.EnclosureTempSetting
U	EnclosureTempTolerance	REAL	For future use.	MyMachine.Printer.EnclosureTempMinTarget
U	BedTempReady	BOOL	When receiving an M140 command, bed temperatures are stored in this structure for use by the MotionWorks IEC application program.	MyMachine.Printer.BedTempReady
U	EnclosureTempReady	BOOL	When receiving an M140 command, bed temperatures are stored in this structure for use by the MotionWorks IEC application program.	MyMachine.Printer.EnclosureTempReady
U	Fan	BOOL	Flag which can be used by the MotionWorks IEC application program to control the Fan. This is set by M106 and M107 commands.	MyMachine.Printer.Fan

U	FanSpeed	BYTE	The fan speed as received from an M106 command.	MyMachine.Printer.FanSpeed
U	Extruder	ExtruderArray	Array of three ExtruderStruct containing information for up to three extruders. (See below.)	MyMachine.Printer.Extruder[0].CmdPos
C	ExtruderScalerEffective	BOOL	This variable is managed by MC_MovePath. The purpose is to allow the PC software such as CNCWorks to report when a change to the Extruder scaler is live on the machine. When the scaler is changed, the correction values are applied to new motion segments, which get placed into a queue, and may not actually cause motion for several seconds. Each time the scaler is changed, this flag is set FALSE and the byte offset of the stream noted. When MC_MovePath determines that the motion instruction including the changed scaler is live (causing motion), the variable is set back to TRUE.	MyMachine.Printer.ExtruderScalerEffective
C	ExtruderOffsetValid	BOOL	For internal use by MC_MovePath. The extruder offset must be valid before a G1 command can command the extruder. When extruders are changed via the T0 ~ T2 command, the G92 command typically defines the start point for the extruder.	MyMachine.Printer.ExtruderOffsetValid
C	ActiveExtruder	INT	Reports the Active Extruder for systems which are configured to operate more than one device. Extruders are switched using the T0 ~ T2 command.	MyMachine.Printer.ActiveExtruder

ExtruderData

Starting in v350, there is an array of ExtruderData to support machines with multiple extruder heads.

In the following chart, the left column indicating 'U' ser or 'C' ontroller - Controller indicates that the [MC_MovePath](#) function block manages the value.

*	Element	Data Type	Description	Usage
	ExtruderData			
U	Axis	AXIS_REF	Reserved for future use.	MyMachine.Printer.Extruder [0].Axis.AxisNum
U	XOffset	LREAL	If the machine is equipped with multiple extruders, these offsets define the physical relationship between the primary and secondary extruders. Can also be used to adjust the general X offset of any extruder.	MyMachine.Printer.Extruder [0].XOffset
U	YOffset	LREAL	If the machine is equipped with multiple extruders, these offsets define the physical relationship between the primary and secondary extruders. Can also be used to adjust the general X offset of any extruder.	MyMachine.Printer.Extruder [0].YOffset
U	ZOffset	LREAL	If the machine is equipped with multiple extruders, these offsets define the physical relationship between the primary and secondary extruders. Can also be used to adjust the general X offset of any extruder.	MyMachine.Printer.Extruder [0].ZOffset
C	EOffset	LREAL	For internal use by MC_MovePath.	MyMachine.Printer.Extruder [0].EOffset
U	ScaleFactor	REAL	The G-Code data specified for the extruder position can be overridden by this factor. The allowable range is limited between 0.5 to 1.5.	MyMachine.Printer.Extruder [0].ScaleFactor
C	CmdPos	LREAL	For internal use by MC_MovePath.	MyMachine.Printer.Extruder [0].CmdPos
C	CmdVel	REAL	For internal use by MC_MovePath.	MyMachine.Printer.Extruder [0].CmdVel
U	TempSetting	YTB_RealArray4	An array of extruder temperature settings which can be populated by M104 or M109 commands.	MyMachine.Printer.Extruder [0].TempSetting
U	MinWarmTemp	YTB_RealArray4	This value can be customized based on application needs. For example, set the value to a temperature that is too hot to touch without burning the skin, such as 45 C. If using CNCWorks, the background color of the Extruder setting area will be blue if TempActual is less than MinWarmTemp and Orange if TempActual is greater than MinWarmTemp but not within the required tolerance.	MyMachine.Printer.Extruder [0].MinWarmTemp
U	TempTolerance	YTB_RealArray4	Enter a tolerance as required before the M109 command will pass program flow to the next command. This applies for either heating to a specified temperature, or cooling to the new desired temperature.	MyMachine.Printer.Extruder [0].TempTolerance
U	TempActual	YTB_RealArray4	Connect the value from the real world temperate sensor in the units of your choice.	MyMachine.Printer.Extruder [0].TempActual
U	TempOffset	REAL	Enter a value as required to override the temperature settings specified in the G ode file.	MyMachine.Printer.Extruder [0].TempOffset
U	RetractLength	LREAL	Reserved for future use.	MyMachine.Printer.Extruder [0].RetractLength
U	UnRetractLength	LREAL	Reserved for future use.	MyMachine.Printer.Extruder [0].UnRetractLength
U	RetractFeedRate	LREAL	Reserved for future use.	MyMachine.Printer.Extruder [0].RetractFeedRate
U	ZLift	LREAL	Reserved for future use.	MyMachine.Printer.Extruder [0].ZLift
U	AirPressure	BOOL	User customizable to report to the PC software. Not used by any Toolbox function block.	MyMachine.Printer.Extruder [0].AirPressure
U	VacuumPressure	BOOL	User customizable to report to the PC software. Not used by any Toolbox function block.	MyMachine.Printer.Extruder [0].VacuumPressure
U	LowTorque	BOOL	User customizable to report to the PC software. Not used by any Toolbox function block.	MyMachine.Printer.Extruder [0].LowTorque

U	HighTorque	BOOL	User customizable to report to the PC software. Not used by any Toolbox function block.	MyMachine.Printer.Extruder [0].HighTorque
U	TempReady	BOOL	User customizable to report to the PC software. Not used by any Toolbox function block.	MyMachine.Printer.Extruder [0].TempReady



Data Type: SegmentDetails

A supporting structure for [MC_PATH_DATA_REF](#). This is an array of SegmentDetails, each of which contains the necessary information for the Segment based on the SegmentType. This list of elements in this structure is comprehensive; note that elements are only used for specific SegmentTypes as detailed below.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyPathStruct	MC_PATH_DATA_REF		
	Segment	SegmentDetails	Data structure used with the MC_MovePath function block.	
U	SegmentType	INT	See GTB_SegmentType .	MyPathStruct.Segment[n].SegmentType
C	Index	INT	Identifier of the Segment Index of this item within MC_PATH_DATA_REF. For use by functions such as CheckColinearity and ToolComp. Typically function blocks which parse source data and load the MC_PATH_DATA_REF will populate this with data from the source for debugging purposes. For example, if a G-Code file contains lines numbers with the code N344, the value 344 will be copied to this element.	MyPathStruct.Segment[n].Index
U	LineNumberRef	DINT	Typically function blocks which parse source data and load the MC_PATH_DATA_REF will populate this with data from the source for debugging purposes. For example, if a G-Code file contains lines numbers with the code N344, the value 344 will be copied to this element.	MyPathStruct.Segment[n].LineNumberRef
U	Label	YTB_STRING16	For debugging purposes, short descriptive labels can be added. When MC_MovePath is executing, it will display the value as a STRING (16) at the VAR_OUTPUT 'SegmentLabel.'	MyPathStruct.Segment[n].Label
U	X	LREAL	If SegmentType is StraightLine, Arc, or Direct, this is the absolute or relative coordinate [based on the AbsoluteMode element] of the X axis within the CoordSystem specified. For moves in ACS, this value corresponds to Joint #1. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].X
U	Y	LREAL	If SegmentType is StraightLine, Arc, or Direct, this is the absolute or relative coordinate [based on the AbsoluteMode element] of the Y axis within the CoordSystem specified. For moves in ACS, this value corresponds to Joint #2. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].Y
U	Z	LREAL	If SegmentType is StraightLine, Arc, or Direct, this is the absolute or relative coordinate [based on the AbsoluteMode element] of the Z axis within the CoordSystem specified. For moves in ACS, this value corresponds to Joint #3. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].Z
U	Rx	LREAL	If SegmentType is StraightLine, Arc, or Direct, this is the absolute or relative coordinate [based on the AbsoluteMode element] of the Rx axis within the CoordSystem specified. For moves in ACS, this value corresponds to Joint #4. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].Rx
U	Ry	LREAL	If SegmentType is StraightLine, Arc, or Direct, this is the absolute or relative coordinate [based on the AbsoluteMode element] of the Ry axis within the CoordSystem specified. For moves in ACS, this value corresponds to Joint #5. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].Ry
U	Rz	LREAL	If SegmentType is StraightLine, Arc, or Direct, this is the absolute or relative coordinate [based on the AbsoluteMode element] of the Rz axis within the CoordSystem specified. For moves in ACS, this value corresponds to Joint #6. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].Rz

*	Element	Data Type	Description	Usage
	MyPathStruct	MC_PATH_DATA_REF		
	Segment	SegmentDetails	Data structure used with the MC_MovePath function block.	
U	E	LREAL	For 3D printing applications. The MachineStruct must be configured with an Extruder axis. See these instructions for Configuring an Extruder axis .	MyPathStruct.Segment[n].E
U	AxesFlags	WORD	These flags identify the axes for which the position data is specifically set for in this segment. This is required for some G-Code commands which behave differently depending on whether a value has been supplied or not. If manually entering SegmentDetails or creating a custom path generator, you must set the appropriate flags or no motion will occur . X=bit 0, Y=bit 1, Z = bit 2, Rx = bit 3, Ry = bit 4, Rz = bit 5.	MyPathStruct.Segment[n].AxesFlags
U	VarFlags	DWORD	For use with G-Code applications. Each bit in the DWORD corresponds to a register letter, bit 0 for A, bit 1 for B, etc. . For example, if the data is specified as G1 X#3 Y#4, the values in MyPath.Segment[n].X and MyPath.Segment[n].Y are not the actual position to be used, but are pointers to variables. The values point to the position values to be referenced from the VarData struct connected to MC_MovePath.	MyPathStruct.Segment[n].VarFlags
U	AbsoluteMode	BOOL	If SegmentType is StraightLine, Arc, or Direct, this specifies whether the coordinates in this Segment are Absolute or Relative within the specified CoordSystem. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].AbsoluteMode
U	CoordSystem	MC_CoordinateSystem	If SegmentType is Linear, Direct, or Circular, this specifies the Coordinate System and will be copied to the corresponding PLCopen Part 4 function block. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].CoordSystem
U	FeedRate	REAL	If SegmentType is StraightLine, Arc, or Direct, this value is scaled by the VelocityOverride input to MC_MovePath and copied to the Input of the corresponding PLCopen Part 4 function block. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].FeedRate
U	ExactStopCheck	BOOL	If SegmentType is Linear, Direct, or Circular and there are several contiguous motion Segments, ExactStopCheck will cause MC_MovePath to wait for motion to come to a stop when reaching the position specified in this Segment before executing additional motion segments.	MyPathStruct.Segment[n].ExactStopCheck
U	TransitionMode	INT	If SegmentType is StraightLine, Arc, or Direct, this value is mapped to motion blocks 'TransitionMode' VAR_INPUT. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].TransitionMode
U	TransitionPrm	REAL	If SegmentType is Linear, Direct, or Circular, this value is mapped to motion blocks 'TransitionParameter' VAR_INPUT. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].TransitionPrm
U	Radius	LREAL	If SegmentType is Arc, this value may be used by Path generating function blocks such as Read_GCode_File or Read_GCode_Stream to convert the necessary circle information for use the MC_MoveCircularAbsolute.	MyPathStruct.Segment[n].Radius

*	Element	Data Type	Description	Usage
	MyPathStruct	MC_PATH_DATA_REF		
	Segment	SegmentDetails	Data structure used with the MC_MovePath function block.	
C	StartAngle	LREAL	If SegmentType is Arc, this value is updated by Path generating function blocks such as Read_GCode_File or Read_GCode_Stream. For debugging purposes only.	MyPathStruct.Segment[n].StartAngle
C	TraversedAngle	REAL	If SegmentType is Arc, this value is updated by Path generating function blocks such as Read_GCode_File or Read_GCode_Stream. For debugging purposes only.	MyPathStruct.Segment[n].TraversedAngle
U	PathChoice	INT	If SegmentType is Arc, this value is mapped to MC_MoveCircularAbsolute's 'PathChoice' VAR_INPUT.	MyPathStruct.Segment[n].PathChoice
U	CircleMode	INT	If SegmentType is Arc, this value is mapped to MC_MoveCircularAbsolute's 'CircleMode' VAR_INPUT.	MyPathStruct.Segment[n].CircleMode
U	CenterXCoord	LREAL	If SegmentType is Arc, this value is mapped to MC_MoveCircularAbsolute's 'AuxPoint' VAR_INPUT and used based on PathChoice and CircleMode. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].CenterXCoord
U	CenterYCoord	LREAL	If SegmentType is Arc, this value is mapped to MC_MoveCircularAbsolute's 'AuxPoint' VAR_INPUT and used based on PathChoice and CircleMode. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].CenterYCoord
U	CenterZCoord	LREAL	If SegmentType is Arc, this value is mapped to MC_MoveCircularAbsolute's 'AuxPoint' VAR_INPUT and used based on PathChoice and CircleMode. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].CenterZCoord
U	AuxXCoord	LREAL	If SegmentType is Arc, this value is mapped to MC_MoveCircularAbsolute's 'AuxPoint2' VAR_INPUT and used based on PathChoice and CircleMode. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].AuxXCoord
U	AuxYCoord	LREAL	If SegmentType is Arc, this value is mapped to MC_MoveCircularAbsolute's 'AuxPoint2' VAR_INPUT and used based on PathChoice and CircleMode. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].AuxYCoord
U	AuxZCoord	LREAL	If SegmentType is Circular, this value is mapped to MC_MoveCircularAbsolute's 'AuxPoint2' VAR_INPUT and used based on PathChoice and CircleMode. For other SegmentTypes, this data is ignored.	MyPathStruct.Segment[n].AuxZCoord
U	GenericLREAL	GenericLREALArray	Used for 3D printer extruder positions, and can also be used by custom segment for hold any register/parameter data necessary. Three LREALs are available.	MyPathStruct.Segment[n].GenericLREAL[2]
U	InputConditions	DWORD	Used only if SegmentType is 'WaitForInputs', 'LoopDecision', 'BranchDecision', 'NonBlockingInputCheck', 'NonBlockingLoopDecision', or 'NonBlockingBranchDecision'. Set the InputConditions required to advance to the next Segment. Only the bits in InputConditionw which are set HIGH are used to determine which bits of MC_MovePath.InputFlags must be set high to continue operation. (Bits set low indicate that the bit is not checked as part of this segments requirement for advancing, it does not mean that the corresponding InputFlags' bit must be low.	MyPathStruct.Segment[n].InputConditions

*	Element	Data Type	Description	Usage
	MyPathStruct	MC_PATH_ DATA_REF		
	Segment	SegmentDetails	Data structure used with the MC_MovePath function block.	
U	InputTimeout	TIME	Used only if the SegmentType is 'WaitForInputs'. This is the TIME that MC_MovePath will wait for (InputConditions = InputFlags). If the machine must wait forever when the InputConditions are unsatisfied, then use 0 TIME. If the InputTimeout value is non zero, MC_MovePath will advance to the next Segment after the InputTimeout. Typically the next Segment would be 'LoopDecision' or 'BranchDecision' with the same InputConditions to permit the desired behavior in the event that the InputConditions have not been met. Specify TIME in standard IEC61131 syntax.	MyPathStruct.Segment[n].InputTimeout
U	StandStillDuration	TIME	Used only if the SegmentType is 'StandStill'. This is the TIME that MC_MovePath will do nothing during this segment. Specify TIME in standard IEC61131 syntax.	MyPathStruct.Segment[n].StandStillDuration
U	TangentActive	BOOL	<p>This flag designates segments for which an axis operating tangent to the vector path must be synced. MC_MovePath will execute the functions Y_SyncTangentAxisToGroup and MC_Stop using the AXIS_REF supplied in the MachineStruct.</p> <p>The MachineStruct must be configured to operate a tangent axis outside of the group configuration. If the tangent axis is not aligned with the upcoming vector path, the PathData can optionally pre align the tangent axis using the Rz value with the proper pre alignment angle. The MC_MovePath can determine that Rz values are for the external tangent by referencing the MachineStruct information and use the MC_MoveAbsolute function block to pre align the tangent axis using rotational velocity and acceleration as specified in the MachineStruct.</p>	MyPathStruct.Segment[n].TangentActive
U	OutputFlags	DWORD	The MC_MovePath function block's VAR_OUTPUT 'OutputFlags' will be set to this value when this Segment is active. If motion is ongoing, the OutputFlags corresponding to this segment is verified by using Group Parameters 2201 and 2202 to account for the processing of buffered motion segments. When there is no motion, such as during SegmentType 'StandStill' or 'WaitForInputs', the OutputFlags are set immediately.	MyPathStruct.Segment[n].OutputFlags
U	LogicEvent	INT	Used only if the SegmentType is 'LoopDecision.' This is a pointer (array index) in the LogicArray information corresponding to this Segment.	MyPathStruct.Segment[n].LogicIndex
U	SelectedTool	INT	<p>If MachineStruct.MachineType = GTB_Machine#TypeMilling, this indicates the required Tool from the ToolStruct.</p> <p>If MachineStruct.MachineType = GTB_MachineType#Printer, this indicates the extruder to be used as set by T0 ~ T2..</p>	MyPathStruct.Segment[n].SelectedTool
C	G_GroupStatus1	DWORD	Holds the G-Code group status information relevant to this segment. The data is reported to StreamStatus for GCodeComm.DLL to provide the information to a PC application.	MyPathStruct.Segment[n].G_GroupStatus1

*	Element	Data Type	Description	Usage
	MyPathStruct	MC_PATH_ DATA_REF		
	Segment	SegmentDetails	Data structure used with the MC_ MovePath function block.	
C	G_GroupStatus2	DWORD	Holds the G-Code group status information relevant to this segment. The data is reported to StreamStatus for GCodeComm.DLL to provide the information to a PC application.	MyPathStruct.Segment[n].G_GroupStatus2
C	M_GroupStatus	DWORD	Holds the M Code group status information relevant to this segment. The data is reported to StreamStatus for GCodeComm.DLL to provide the information to a PC application.	MyPathStruct.Segment[n].M_GroupStatus

Not all elements are used for each SegmentType. The SegmentType element dictates the data which will be referenced for a given Segment[]. The following chart outlines the relationship between SegmentType and its relevant information.

Example 1

If designing a custom path generating algorithm, it is useful to know which elements of the PathData.Segment[] are applicable based on the SegmentType specified. Refer to the following chart for usage by [MC_MovePath](#).

Attribute	Straight Line	Arc	Stand Still	Wait For Inputs	Set Tangent	Direct	Jump Decision	Branch Decision	Non Blocking	Input Look Ahead	Move To Origin	Set Position	Non Blocking	Jump Decision	Non Blocking Branch Decision
Line Number Ref	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Label	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
X	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Y	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Z	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Rx	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Ry	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Rz	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
E	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Axes Flags	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Absolute Mode	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Coordinate System	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Feed Rate	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Exact Stop Check	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Transition Mode	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Transition Prm	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Radius	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Start Angle	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Traversed Angle	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Path Choice	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Circle Mode	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Center X Coord	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Center Y Coord	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Center Z Coord	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Aux X Coord	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Aux Y Coord	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Aux Z Coord	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Input Conditions	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Input TimeOut	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Stand Still Duration	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Tangent Active	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Output Flags	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Logic Event	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green



Data Type: SegmentMapStruct

This structure is used by [MC_MovePath](#) to track buffered motion segments. It is necessary to control the OutputFlags in synchronization with the motion specified. Motion Segments may be executed some time before they are actually causing motion based on the number of motion blocks buffered.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MySegmentMap	SegmentMapStruct		
C	Map	MapArray	Stores the relationship between a PachStruct.Segment and the ID of the motion function executed / queued by the motion engine.	MySegmentMap.Map[0][134].PathIndex
C	Buffer	SegmentBuffer	AXIS_REF of axis to be used for tangent motion.	MySegmentMap.Buffer[1].StorePointer
C	ActiveStoreMap	INT	Flag [0 or 1] which indicates the MapArray to which data is being stored.	MySegmentMap.ActiveStoreMap
C	ActiveUseMap	INT	Flag [0 or 1] which indicates the MapArray from which data is being read.	MySegmentMap.ActiveUseMap



Data Type:SpindleStruct

Contains information for use by externally operated Spindles (not configured as part of the AxesGroup).

Data Type Declaration

Certain G & M codes will write data into MachineData.Spindle[]. This data is available for the user to access and customize the project. The only built in spindle features include:

- MC_MovePath waits until MachineData.Spindle[].SpeedOK is TRUE after executing a M3 / M4 / M5 command.
- Set MachineData.Spindle[].RPM to zero and restore it for ProgramStop events (M0, M1) and Feed Hold.
- In G97 mode, MC_MovePath Calculates MachineData.Spindle[].RPM.

*	Element	Data Type	Description	Usage
	MyTools	ToolStruct		
U	RPM	REAL	When G96 or G97 is received, the S register is used to populate this value.	MyMachine.Spindle.RPM
U	Override	REAL	For user customization.	MyMachine.Spindle.Override
U	Accel	REAL	For user customization.	MyMachine.Spindle.Accel
U	MaxPRM	REAL	Only used in Constant Surface Speed Mode.	MyMachine.Spindle.MaxRPM
U	SurfaceSpeed	REAL	When G96 is received, UnitCode is set to 0 to indicate 'Surface Speed' Mode When G97 is received, UnitCode is set to 1 to indicate 'Feet Per Minute' Mode.	MyMachine.Spindle.SurfaceSpeed
U	UnitCode	INT	When G96 is received, UnitCode is set to 0 to indicate 'Surface Speed' Mode When G97 is received, UnitCode is set to 1 to indicate 'Feet Per Minute' Mode.	MyMachine.Spindle.UnitCode
U	Direction	INT	Same as M03 (Clockwise Rotation), M04 (CounterClockwise Rotation).	MyMachine.Spindle.Direction
U	Axis	AXIS_REF	For user customization.	MyMachine.Spindle.Axis.AxisRef
U	OrientationPosition	LREAL	Could come from an M19, or other method to move the spindle to a specific angle.	MyMachine.Spindle.OrientationPosition
U	OrientationExe	LREAL	Set by MC_MovePath when an M19 is encountered.	MyMachine.Spindle.OrientExe
U	OrientationDone	BOOL	To be populated by the the template layer project code if the user wants to show this information on a GUI via the MachineStruct.	MyMachine.Spindle.OrientationDone
U	OrientationBusy	BOOL	To be populated by the the template layer project code if the user wants to show this information on a GUI via the MachineStruct.	MyMachine.Spindle.OrientationBusy
U	OrientationAborted	BOOL	To be populated by the the template layer project code if the user wants to show this information on a GUI via the MachineStruct.	MyMachine.Spindle.OrientationAborted
U	OrientationError	BOOL	To be populated by the the template layer project code if the user wants to show this information on a GUI via the MachineStruct.	MyMachine.Spindle.OrientationError
U	OrientationErrorID	UINT	To be populated by the the template layer project code if the user wants to show this information on a GUI via the MachineStruct.	MyMachine.Spindle.OrientationErrorID
U	ToolPresent	BOOL	Connect a BOOL to this structure item if there is a sensor mounted to detect a tool in the chuck.	MyMachine.Spindle.ToolPresent
U	AtSpeed	BOOL	Indicates when the spindle motor is within a target speed so MC_MovePath can pause and wait for the spindle to get up to speed. For a basic example, see this spindle support section.	MyMachine.Spindle.AtSpeed
U	FeedbackSpeed	REAL	To be populated by the the template layer project code if the user wants to show this information on a GUI via the MachineStruct.	MyMachine.Spindle.FeedbackSpeed
U	FeedbackTorque	REAL	To be populated by the the template layer project code if the user wants to show this information on a GUI via the MachineStruct.	MyMachine.Spindle.FeedbackTorque



Data Type: StreamStruct

This is a substructure within [MC_PATH_DATA_REF](#). It holds data for reporting to a host application, such as Compass, via Ethernet.

Version information

Release Date	Compatibility		
	GCodeComm.DLL	UDP Packet	Group Tool-box
January 2020	3.4.2.0	20190127	v361 to v374b
February 2019	3.4.2.0	20190127	v352
November 2018	1.0.1.1	20180103	v350
August 2018	3.4.1.0		
January 2018			v340

Data Type Declaration (20190127)

*	Element	Data Type	Description	Usage
	MyPath.StreamStatus	StreamStruct		
C	Version	UDINT	Identifier for packet structure content. The GCodeComm DLL uses this information to confirm the data contents.	MyPath.StreamStatus.Version
C	TCPPacketCount	UDINT	Total packets received since a connection was made to the Read_GCode_Stream function block.	MyPath.StreamStatus.TCPPacketCount
C	MCSPosition	MC_CARTESIAN_REF	Array of 6 world space positions in User Units as declared in the Hardware Configuration.	MyPath.StreamStatus.MCSPosition[3]
C	PCSPosition	MC_CARTESIAN_REF	Array of 6 world space positions in User Units as declared in the Hardware Configuration.	MyPath.StreamStatus.PCSPosition[3]
C	AuxPosition	YTB_LrealArray8	Array of 8 positions for any auxiliary axes added to the group in Hardware Configuration.	MyPath.StreamStatus.AuxPosition[2]
C	ToGoDistance	MC_CARTESIAN_REF	The remaining distance for each axis in the active coordinate system until it achieves the position of the most immediate move instruction. If many instructions are queued, ToGoDistance only reflects the live motion instruction, but will continue to update as each new instruction becomes live.	MyPath.StreamStatus.ToGoDistance[3]
C	TCPVelocity	REAL	Velocity of the TCP in User Units as declared in the Hardware Configuration.	MyPath.StreamStatus.TCPVelocity
C	Velocity	LREALArray6	Array of 6 world space velocities in User Units as declared in the Hardware Configuration.	MyPath.StreamStatus.Velocity[1]
C	Torque	YTB_LrealArray16	Contains the joint torques for each motor configured in the group, which includes prime axes and auxiliary axes.	MyPath.StreamStatus.Torque[6]
C	Path	PathStatusStruct	Data pertaining to the path such as segments processed, current path segment.	See below
C	Buffer	BufferStatusStruct	Data pertaining to the three buffers involved with processing data: ByteBuffer, PathBuffer, and MotionBufer.	See below
C	FB	FBStatusStruct	Provides the Read_GCode_Stream and MC_MovePath function block outputs to a host application.	See below
C	G_GroupStatus1	DWORD	Reports the active G-Code for each of the predefined G-Code groups. The GCodeComm DLL can interpret the raw bit data in the DWORD and report meaningful information.	PathData.StreamStatus.G_GroupStatus1
C	G_GroupStatus2	DWORD	Reports the active G-Code for each of the predefined G-Code groups. The GCodeComm DLL can interpret the raw bit data in the DWORD and report meaningful information.	PathData.StreamStatus.G_GroupStatus1
C	M_GroupStatus	DWORD	Reports the active G-Code for each of the predefined G-Code groups. The GCodeComm DLL can interpret the raw bit data in the DWORD and report meaningful information.	PathData.StreamStatus.M_GroupStatus

PathStatusStruct

*	Element	Data Type	Description	Usage
C	MyPath.StreamStatus.Path InUse	PathStatusStruct BOOL	Set TRUE by the MC_MovePath function block any time it is Busy. This prevents a function that loads data into MC_PATH_DATA_REF from initializing the data while it is being accessed to provide motion.	MyPath.StreamStatus.Path.InUse
C	ByteOffset	UDINT	Provides a way to link back to the source data. ByteOffset refers to the first character of the G-Code which is currently providing motion (or live execution, such as M3). This can be used by PC software systems to provide motion recovery and resume a path already in progress.	MyPath.StreamStatus.Path.ByteOffset
C	ProcessingByteOffset	UDINT	Provides a way to link back to the source data. ProcessingByteOffset refers to the first character of the G-Code which is currently being converted from the data stream into MPiec data. This can be used by PC software applications to show the line which may have resulted in an error condition, such as an unsupported command or bad syntax.	MyPath.StreamStatus.Path.ProcessingByteOffset
C	ExecutingByteOffset	UDINT	Provides a way to link back to the source data. ExecutingByteOffset refers to the first character of the G-Code which is currently being sent to the firmware layer. The firmware may reject a motion command if for example circle parameters do not represent a valid arc. This can be used by PC software applications to show the line which may have resulted in an error condition.	MyPath.StreamStatus.Path.ExecutingByteOffset

BufferStatusStruct

*	Element	Data Type	Description	Usage
	MyPath.StreamStatus.Buffer	Buffer-StatusStruct		
C	BytePercent	REAL	Indicates the percentage (0.0 to 100.0%) of the CircularByteBuffer that is filled with data. This information can be used by a host application such as GCodeComm.DLL to throttle the data being streamed to the Read_GCode_Stream function block.	MyPath.StreamStatus.Buffer.BytePercent
C	BytesAvailable	DINT	Number of bytes available to be filled in the CircularByteBuffer (typically for streaming applications.)	MyPath.StreamStatus.Buffer.BytesAvailable
C	BytesUtilization	REAL	Not used.	MyPath.StreamStatus.Buffer.BytesUtilization
C	PathPercent	REAL	Indicates the percentage (0.0 to 100.0%) of the PathData.Segment[] array that is filled with data. Filled means new data has been written, but the MC_MovePath function block has not accessed it yet. This can be useful for debugging and performance analysis.	MyPath.StreamStatus.Buffer.PathPercent
C	PathAvailable	DINT	Number of Segments in MC_PATH_DATA_REF that are available to be filled.	MyPath.StreamStatus.Buffer.PathAvailable
C	UnderRunWarning	BOOL	This bit will be set if (Motion has started) AND (PathData.FinalSegment=Undetermined) AND (PathData.Buffer.StorePointer = PathData.Buffer.UsePointer) AND (AxesGroup.Status.NumMotionSegments=0) This situation should never occur, and means the system is starved for data, or the end of a G-Code file was not properly specified with an M30 command.	MyPath.StreamStatus.Buffer.UnderRunWarning
C	MotionPercent	REAL	Indicates the percentage (0.0 to 100.0%) of the firmware layer motion queue that has a function block waiting to provide motion.	MyPath.StreamStatus.Buffer.MotionPercent
C	MotionAvailable	DINT	Availability in the firmware layer motion queue. This is AxesGroup.Status.FreeMotionSegments.	MyPath.StreamStatus.Buffer.MotionAvailable

FBStatusStruct

*	Element	Data Type	Description	Usage
	MyPath.StreamStatus.FB.ReadStream	ReadGCodeStreamStatus		
C	ErrorID	UINT	The ErrorID reported by the ReadGCode_Stream function block.	MyPath.StreamStatus.FB.ErrorID
C	ErrorRow	UDINT	If applicable, the line number from the G-Code data which corresponds to the ErrorID.	MyPath.StreamStatus.FB.ErrorRow
C	ErrorString	YTB_STRING16	If applicable, the string value of the command in the G-Code data which caused the processing error. In some cases when there is more than one command per line, this string may be inaccurate.	MyPath.StreamStatus.FB.ErrorString
C	InstructionsProcessed	UDINT	The total number of instructions processed since processing began.	MyPath.StreamStatus.FB.InstructionsProcessed

	MyPath.StreamStatus.FBMovePath	MovePathStatusStruct		
C	InputFlagsRequired	DWORD	Reports the input conditions that MC_MovePath is checking to advance to the next segment	MyPath.StreamStatus.FB.InputFlagsRequired
C	OutputFlags	DWORD	Reports the outputs that are being set based on the currently executing segment.	MyPath.StreamStatus.FB.OutputFlags
C	ErrorID	UINT	The ErrorID reported by the MC_Movepath function block.	MyPath.StreamStatus.FB.ErrorID
C	ProcessedLabel	YTB_STRING16	If populated, reports PathData.Segment[p].Label, where [p] points to the segment information that is being referenced in order to execute a motion or other sequence related instruction.	MyPath.StreamStatus.FB.ProcessedLabel
C	ExecutedLabel	YTB_STRING16	If populated, reports PathData.Segment[e].Label, where [e] points to the segment that is currently providing motion or other sequence related instruction.	MyPath.StreamStatus.FB.ExecutedLabel
C	ProcessedTotal	UDINT	The total number of segments evaluated and processed.	MyPath.StreamStatus.FB.ProcessedTotal
C	ExecutedTotal	UDINT	The total number of segments completed (motion was satisfied or other sequence related activity was successful.)	MyPath.StreamStatus.FB.ExecutedTotal



Data Type: ToolDataStruct

Contains Tool information used in [Read_GCode_File](#) and [Read_GCode_Stream](#).

ToolData is typically populated by the application program via an HMI or other source. It contains tool data for use with any compensation (G41, G42, G43, G44) G-Code commands. ToolDataStruct contains an array of 100 Tools. When using Compass software, this data can be viewed and edited via the Tool Manager.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyTools	ToolDataStruct		
U	Tool	ToolDataArray	Structure containing an array of 100 Tools . See below.	MyTools.Tool[1].Radius
U	Selected	INT	Value to be used as array index into MyTools.Tool [].	MyTools.Selected
U	LastTool	INT	Set this value to the last defined tool (array index) in ToolDataArray.	MyTools.LastTool
U	Station	StationArray	Structure containing an array of 100 Stations . See below.	MyTools.Station[1].RotaryPosition
U	SelectedStation	INT	Value to be used as array index into MyTools.Station [].	MyTools.SelectedStation
U	LastStation	INT	Set this value to the last defined Station (array index) in StationArray.	MyTools.LastTool
U	Changer	ChangerArray	Structure containing an array of 6 Changers . See below.	MyTools.SelectedStation
U	LastChanger	INT	The actual number of tool changers defined in the application.	MyTools.LastTool

Tools

*	Element	Data Type	Description	Usage
		Tools		
U	Station	INT	To indicate where this tool is located, if there is more than one tool changer system on the machine. This is the array index of MyTools.Station[].	MyTools.Tool[2].Station
U	Description	STRING	For describing the tool, not used by the system,	MyTools.Tool[2].Description
U	Radius	LREAL	Radius of a tool in the groups Cartesian user units from the Hardware Configuration.	MyTools.Tool[2].Radius
U	NominalRadius	LREAL	Length of a tool in the groups Cartesian user units from the Hardware Configuration.	MyTools.Tool[2].NominalRadius
U	LengthOffset	LREAL	Not the tools actual length, but a measured length relative to some known point. See emulation mode2 G43.	MyTools.Tool[2].LengthOffset
U	Length	LREAL	Length of a tool in the groups Cartesian user units from the Hardware Configuration.	MyTools.Tool[2].Length
U	LengthNominal	LREAL	Radius of a tool in the groups Cartesian user units from the Hardware Configuration.	MyTools.Tool[2].LengthNominal
U	MaxRPM	INT	For reference only, not used by the system.	MyTools.Tool[2].MaxRPM
U	COffset	LREAL	Position Offset for Rz axis. Useful when operating an accessory tool rather than a traditional drilling or routing bit.	MyTools.Tool[2].COffset

Stations

*	Element	Data Type	Description	Usage
		Tools		
U	Changer	INT	The index of the changer in ChangerArray.	MyTools.Tool[2].Station
U	Description	MC_ CARTESIAN_REF	For describing the tool, not used by the system,	MyTools.Tool[2].Description
U	Radius	LREAL	Radius of a tool in the groups Cartesian user units from the Hardware Configuration.	MyTools.Tool[2].Radius

ChangerStruct

*	Element	Data Type	Description	Usage
		Tools		
U	ChangerType	INT	To indicate rotary or linear changer. 0=Rotary (one degree of offset); 1=Linear Grid (six degrees of offset)	MyTools.Changer[1].ChangerType
U	BaseLocation	MC_ CARTESIAN_REF	MCS position of the base of the changer. For rotary, this must be centered over the tool location.	MyTools.Changer[1].BaseLocation



Data Type: UserApplicationData

Contains information for use with robot and pendant applications, also used by Compass.

Data Type Declaration

*	Element	Data Type	Description	Usage
	UserApplicationData			
U	Config	ConfigStruct	Contains user settable configuration items for the application, such as JogSpeeds.	MyData.Config.JogSpeed.Linear[3]
U	PointList	TeachPointArray		MyData.PointLost[4].RecordedPoint[2].TCPCoordinate[5]
U	Tool	ToolStruct		
U	ToolValid	BOOL_ARRAY_64		
U	PartFrame	PartFrameArray		
U	CubicIZ	CubicIZArray		
U	MaxPointLists	DINT		MyData.MaxPointLists
U	MaxTools	DINT		MyData.MaxTools
U	MaxPartFrames	DINT		MyData.MaxPartFrames
U	MaxCubicIZs	DINT		MyData.MaxCubicIZs
U	MaxTeachPoints	DINT		MyData.MaxTeachPoints
U	PrimeAlignment	AlignPrimePrms	Only necessary if the group contains more than one servo operating an joint. See the Getting Started with Secondary Axes section.	MyData.PrimeAlignment.Allowance

ConfigStruct

*	Element	Data Type	Description	Usage
	ExtruderData			
U	JogSpeed	JogSpeedStruct	Contains 4 settable jog speeds for Joint, Linear, and orientation motion.	MyData.Config.JogSpeed.Linear[3]

JogSpeedStruct

*	Element	Data Type	Description	Usage
	ExtruderData			
U	Joint	GTB_LREALArray4	4 speeds for Joint jogs (ACS)	MyData.Config.JogSpeed.Joint[1]
U	Linear	GTB_LREALArray4	4 speeds for linear moves (MCS)	MyData.Config.JogSpeed.Linear[3]
U	Orientation	GTB_LREALArray4	4 speeds for Rx, Ry, Rz moves (MCS)	MyData.Config.JogSpeed.Orientation[0]

Example Configuration

```
150 AppData.Config.JogSpeed.Linear[0]:=LREAL#2.0; (* 2 mm/sec *)
151 AppData.Config.JogSpeed.Linear[1]:=LREAL#10.0; (* 10 mm/sec *)
152 AppData.Config.JogSpeed.Linear[2]:=LREAL#20.0; (* 20 mm/sec *)
153 AppData.Config.JogSpeed.Linear[3]:=LREAL#50.0; (* 50 mm/sec *)
154
155 AppData.PrimeAlignment.Tolerance:=LREAL#3.0; (* Allow up to 3 mm to align or generate an error *)
```



Enumerated Types for Group Toolbox

Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to zero-based integers (INT). Therefore, the first value equates to zero, the second to 1, etc. The format for enumerated types is as follows: ENUM:(0, 1, 2...) as displayed in the example below (MC_BufferMode#Aborting).

Enumerated Types Declaration

Enumerated Type	#INT Value	Enum Value	Description
GTB_SegmentType			ENUM for SegmentType

Enumerated Type	#INT Value	Enum Value	Description
	0	n/a	Not a valid SegmentType.

Enumerated Type	#INT Value	Enum Value	Description
	1	StraightLine	Straight line motion between two coordinate locations.

Enumerated Type	#INT Value	Enum Value	Description
	2	Arc	Arc or circular path.

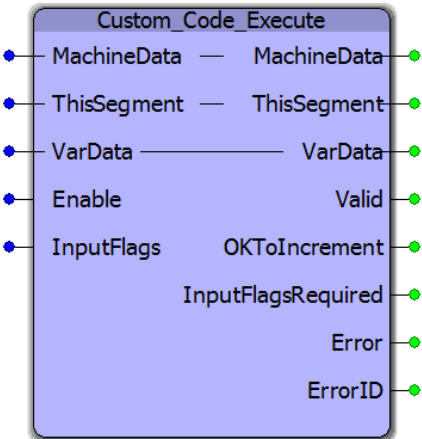
Enumerated Type	#INT Value	Enum Value	Description
	3	StandStill	Pause between segments for a specified delay time.

Enumerated Type	#INT Value	Enum Value	Description
	4	WaitForInputs	Path processing will wait until the specified input conditions are met. MC_MovePath will compare this segments InputConditions to the MC_MovePath.InputFlags.
	5	SetTangent	Preparation move for a tangent axis to track a tangent path relative to the XY vector path.
	6	Direct	Non linear motion between two coordinate locations.
	7	JumpDecision	Jump to another non sequential Segment (either forward or backward) based on InputConditions.
	8	BranchDecision	End the path early based on InputConditions. MC_MovePath.Done will come on, the application can check MC_MovePath.ProcessedLabel to determine if the path completed prematurely.
	9	NonBlockingInputCheck	Don't wait to process future Segments if the input conditions are already met. For example, consider Segment types (1,1,1,1,1,9,1,1,1,1,1.) StraightLine segments will be executed / buffered to the motion queue until a non motion segment type is encountered. When segment type 9 is encountered, the Segments InputConditions are compared to MC_MovePath.InputFlags. As soon as they are satisfied, processing of the segments after type 9 resumes. This differs from Segment type 4 because if the path sequence contains SegmentTypes (1,1,1,1,1,4,1,1,1,1,1) the inputs specified are not evaluated until after the first series of linear motions has been executed and the motion completed.
	10	InputLookAhead	Reserved for future use.
	11	MoveToOrigin	Reserved for future use.
	12	SetPosition	Set the position of an extruder axis. (3D Printing support)
	13	NonBlockingJumpDecision	Similar to SegmentType 7, but processing does not wait until any previously buffered motion has completed.
	14	NonBlockingBranchDecision	Similar to SegmentType 8, but processing does not wait until previously buffered motion has completed.
	15	SubCall	Designed for use by G-Code "IF" and "M98" commands, this segment can switch program flow to a non sequential segment. A companion sub structure PathData.Logic.Event, linked via PathData.Segment[p].LogicEvent, must be loaded with the proper data to satisfy the required logical operation of the Sub call.
	16	SubReturn	MC_MovePath refers to PathData.Logic.Event[p].ReturnSegment to set the next segment to run after the sub call is complete.
	17	SetFrameOffset	MC_MovePath executes Y_GroupSetFrameOffset to set the PCS relationship to the MCS. If ThisSegment.AxesFlags <> WORD#0, then the Segment X,Y,Z,Rx,Ry,Rz values are used, otherwise, the offsets are references from MachineData.CoordinateSystem.Offset. Use ThisSegment.CircleMode to act as the index pointer to the coordinate system.
	18	Expression	Designed for use by G-Code "IF" and variable assignments such as #3=100.
	19	SetOutput	Designed for use by G-Code G62 and G63 commands. If outputs must be set in between motion commands, this special Segment Type can be used. Normally, outputs are set to coincide with a motion segment, or remain in a certain state for other segments. If a G63 command exists on a line by itself (single instruction block) then no other segment type will be associated with the output setting. In these cases, the G-Code processor will automatically set this segment type, forcing the output to be set.
	20	SetFanSpeed	For 3D printing applications. Activated by the M106 command.
	21	SetExtruder	For 3D printing applications. Activated by the M104 command.
	22	SetExtruderWait	For 3D printing applications. Activated by the M109 command.
	23	SetWorkOffset	Changes the work coordinate system offset, typically from a G10 command. This only changes the offset stored in MachineStruct.CoordinateSystem[]. It does not change the offset in use by the machine. (Reference G54 ~ G59.)
	24	FinalSegment	Indicates the final segment of the path. If multiple M2 or M30 commands are present due to logical sequences, the first 'FinalSegment' encountered will end path execution.
	25	OptionalStop	From M0 and M1 commands which act as breakpoints requiring MC_MovePath.Execute to be triggered again to resume.
	26	SubCallHeader	Placeholder segment to indicate the start of a subroutine. Represents the location of the Oxxx sub program label.
	27	SetBed	Write the temperature data from M140 H & S registers to MachineStruct.Printer.BedTempSetting[]

Enumerated Type	#INT Value	Enum Value	Description
	28	Spindle	Writes the spindle data provided by G96 / G97 /M3 / M4 / M5 and the S register to MachineStruct.Spindle[.].
	29	SetBedWait	Write the temperature data from M140 H & S registers to MachineStruct.Printer.BedTempSetting[.] and wait.
	30	ShareToolInfo	Provides MC_MovePath with the Tool Length Offset when a T or G43 or G49 command is executed.



Custom_Code_Execute



This is a special function block which must be copied and pasted from the Group Toolbox to the Logical POU tree of the main project for customization. Its purpose is to execute custom G or M code actions required by the user application that are not supported by the Group Toolbox. Do not include this function in any main project POU. It will automatically be called by [MC_MovePath](#) as necessary when unsupported commands are encountered.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	MachineData	MachineStruct	Contains parameters such as maximum velocities and accelerations, and support for 3D printers and tangent axes if required. Also included are Origin (Home) and Part Coordinate System offsets.
V	ThisSegment	SegmentDetails	Structure of data that holds information for an individual Segment of information along the path.
V	VarData	VariableArray	An array of LREAL data which can be used with G-Code applications where registers contain variable data. The variable ThisSegment contains an element named VarFlags. VarFlags is a DWORD where each bit indicates if a variable was provided for a specific G-Code register. A=bit 0, B=bit 1, C=bit 2, etc. If the application will not require the use of variables, it is not important that this variable be Global, or linked with any other data or function blocks, but it must be connected to satisfy the compiler.
VAR_INPUT			Default

B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	InputFlags	DWORD	The custom G-Code can optionally reference the bits of this DWORD. The data is connected to the VAR_INPUT of MC_MovePath.	DWORD#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	OKToIncrement	BOOL	The customized code sets this to TRUE by default. Under each CASE statement, customize the code if required to conditionally inform MC_MovePath if the activity taking place in the custom G or M code event has passed the conditions required to advance along in the path. See examples below.	
V	InputFlagsRequired	DWORD	For debugging purposes. If a custom G or M code requires waiting for one of the 32 bits connected to MC_MovePath.InputFlags, Custom_Code_Execute can report the bits required to continue along the path. This is useful for debugging purposes. When reported at the output of MC_MovePath, this information is also sent to a PC application via the GCodeComm DLL.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

- See the recommended procedures and example in the help for [Custom Code Processor](#).

Controlling Program Flow

Custom_Code_Execute has the ability to control processing based on logical conditions. A VAR_OUTPUT named 'OKToIncrement' is referenced by [MC_MovePath](#) after the call to Custom_Code_Execute to determine when it should advance to the next Segment instruction. By default, OKToIncrement is TRUE, meaning the custom code will complete in one scan. See the examples in Custom_Code_Execute for programmatically setting OKToIncrement.

Once the data is copied into the user defined data structure, the MotionWorks IEC application has full access to the data and can be programmed to use the information with the relevant components, such as a SLIO remote I/O, LIO card, third party Ethernet/IP device, etc.

Adding an Application Specific Data Structure

In the examples shown, notice the variable "MyAppData" in several locations. Most applications will require a structure to hold the data required for access by the custom codes.

Procedure

- 1) Create a User Defined Data Type in the main project. This structure design is based completely on the needs and complexity required of the application.
- 2) Create a Global variable of this new User Defined Type. The variable can have any name desired.
- 3) Right click on the variables grid of Custom_Code_Execute and add this variable as VAR_EXTERNAL. This data structure is not required by Custom_Code_Processor.

It is now possible to transfer data to/from the G and M Codes to the Application specific data structure, where they can be further processed.

Error Description

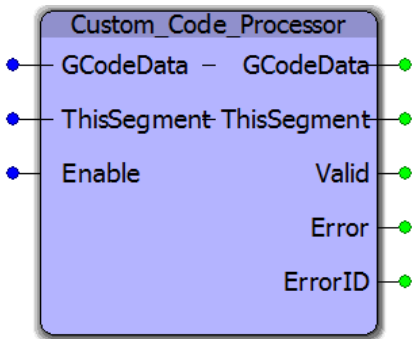
See the [Function Block ErrorID](#) list.

Examples

All examples for this function are listed in [Custom_Code_Processor](#) because the two blocks are related and customization must be performed together.



Custom_Code_Processor



This is a special function block which must be copied and pasted from the Group Toolbox to the Logical POU tree of the main project for customization. Its purpose is to process overrides or custom G or M codes required by the user application that are not supported by the Group Toolbox. It must have the exact name as shown: "Custom_Code_Processor."

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_IN_OUT				
V	GCodeData	GCodeStruct	Working data set used by the G-Code Processor. It is made available as a VAR_IN_OUT mainly for debugging purposes. For applications using variables for register values, this GCodeData.Variables sub structure must be connected to the VAR_IN_OUT "VarData" of MC_MovePath .	
V	ThisSegment	SegmentDetails	Structure of data that holds information for an individual Segment of information along the path. A sub structure within PathData .	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Procedure

1. Open the main project in MotionWorks IEC.
2. Open a second copy of MotionWorks IEC, and open the Group Toolbox, typically from the "C:\User-s\Public\Documents\MotionWorks IEC 3 Pro\Libraries" folder. It is also possible to locate the Group Toolbox referenced in your project by right clicking on the Group Toolbox / User Library and viewing its Properties.
3. In the Project Tree Window of the restricted version of MotionWorks IEC, locate the functions Custom_Code_Execute and Custom_Code_Processor.
4. Copy these functions by multi selecting and Copy. In the main project, right click on the Logical POU's section of the Project Tree Window and Paste.
5. Important - Even though this function is called internally by the G-Code processor, it is necessary to add it to a POU in the main project so the compiler knows which instance to reference. Attach appropriate VAR_IN_OUT variables to satisfy the compiler requirement. It is not necessary to populate relevant data in the connected variables as they will not be used.
6. The function blocks serve as templates for customization. Within the overall code architecture of the functions, add custom codes to the G or M code section of the CASE statements.

Motion Flow Considerations

Motion flow can be configured by using the custom segments "ExactStopCheck" BOOL flag as shown in the examples below.

Setting ExactStopCheck = FALSE (Default)

Any motion commands surrounding the custom code will be not be stopped. For example, an analog or digital output must be synchronized with motion. The custom code will be placed in a monitoring queue to be executed when the motion path (which is likely buffered) actually crosses the preceding coordinate position. In this scenario, Custom_Code_Execute (for the custom code in question) cannot be configured with **OKToIncrement** logic to control program flow. OKToIncrement will be ignored when ExactStopCheck=FALSE.

Up to 30 custom codes can be deferred for later execution.

Setting ExactStopCheck = TRUE

The custom code will wait for any preceding Motion commands to stop before executing. In this scenario, Custom_Code_Execute can be customized to allow the program flow (and motion) to continue only after any required conditions have been met. For example, if a heater or pressure reading must be within a required value to proceed. See the **OKToIncrement** customization below in Example #1, line 10.

Error Description

See the [Function Block ErrorID](#) list.

Example #1 - Setting a BOOL value to the MPiec Controller

Assume a custom M code command "M742" is required to set a BOOL value, and M743 to clear the same BOOL.

Custom_Code_Processor function block

Simply add the custom codes to the CASE statement as shown on line 28. This identifies the codes as recognized by system, and prevents the CASE ELSE condition from setting the "UnsupportedMCode" Error condition.

```

12 IF (ThisSegment.SegmentType >= INT#1000) AND (ThisSegment.SegmentType <= INT#1999) THEN
13   (* M Codes must be included here in the CASE statement to signify support, and to prevent the ELSE condition from throwing an error. *)
14   CASE GCodeData.Register._G OF
15     123: (* Wait until Oxygen level falls below a specified value *)
16         ThisSegment.GenericLREAL[1]:=INT_TO_LREAL(GCodeData.Register._O);
17     555:
18         ThisSegment.GenericLREAL[1]:=GCodeData.Register._P;
19         ThisSegment.ExactStopCheck:=FALSE; (* Set FALSE to force motion ahead and behind this instruction to continue without stopping *)
20
21   ELSE
22     UnsupportedGCode:=TRUE; (* Do not remove this line *)
23   END_CASE;
24
25 ELSE
26   (* M Codes must be included here in the CASE statement to signify support, and to prevent the ELSE condition from throwing an error. *)
27   CASE GCodeData.Register._M OF
28     742, 743: (* Example to set or clear a boolean output. *)
29       ;(* Nothing else in ThisSegment will be referenced by Custom_Code_Execute *)

```

Custom_Code_Execute function block

Simply add the code as shown on lines 21 and 22. The variable CustomMCode will have the value of 742 or 743 when the CASE statement is executed, and can be used in the logic as shown to set or clear a BOOL variable in a user defined structure referenced in this function as VAR EXTERNAL.

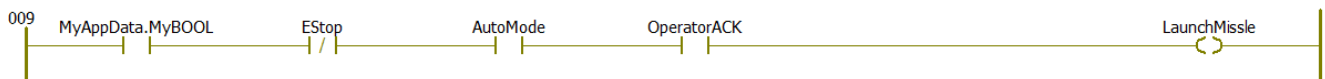
```

6 IF (ThisSegment.SegmentType >= INT#1000) AND (ThisSegment.SegmentType <= INT#1999) THEN
7   CustomGCode:=ThisSegment.SegmentType - INT#1000; (* Convert back to original custom M code for ease of programming *)
8   CASE CustomGCode OF
9     123: (* Wait until Oxygen level falls below a specified value *)
10         OKToIncrement:=MyAppData.OxygenLevel < ThisSegment.GenericLREAL[1];
11     555:
12         MyAppData.GasCoolingTemp:=ThisSegment.GenericLREAL[1];
13
14   ELSE
15     UnsupportedGCode:=TRUE; (* Do not remove this line *)
16   END_CASE;
17 ELSE
18   CustomMCode:=ThisSegment.SegmentType - INT#2000; (* Convert back to original custom M code for ease of programming *)
19
20   CASE CustomMCode OF
21     742, 743: (* Example to set or clear a Boolean output *)
22       MyAppData.MyBOOL := (CustomMCode = INT#742);

```

Additionally, choose a program location in the project to link MyAppData.MyBOOL to the relevant device. Consider other logic which may be required to safely operate the device under all machine conditions. Yaskawa recommends adding only the minimum code to link custom Codes to User structures within the Custom_Code_Execute function and adding additional interlock logic externally.

(*Linked G Code value*)



(*Required Additional Conditions*)

(*Connected to physical output*)

Example #2 - Write an analog or multiple analog values to a custom user structure on the MPiec Controller.

Assume a custom G-Code command "G392" is required to set several values, specified in the H, I, J, and Q registers.

G392 H12.5 I550.0 J-0.5 Q11.12

The variable 'ThisSegment' is one of the SegmentDetails in an array of SegmentDetails in the PathData structure. 'ThisSegment' is exclusive to this custom code, so any data element in 'ThisSegment' can be used without interfering with other processing in the Group Toolbox. This example shows usage of the Generic LREAL elements provided, and the FeedRate variable, even though the value transferred here may have nothing to do with FeedRate. When implementing custom codes, its only important to link the same data in both function blocks, and consider the datatype.

Custom_Code_Processor function block

Add 392 to the CASE statement as shown on lines 15 through 19.

```
12 IF (ThisSegment.SegmentType >= INT#1000) AND (ThisSegment.SegmentType <= INT#1999) THEN
13   (* M Codes must be included here in the CASE statement to signify support, and to prevent the ELSE condition from throwing an error. *)
14   CASE GCodeData.Register._G OF
15     392: (* // Set configuration parameters for the external widget *)
16         ThisSegment.GenericLREAL[0] := GCodeData.Register._H;
17         ThisSegment.GenericLREAL[1] := GCodeData.Register._I;
18         ThisSegment.GenericLREAL[2] := GCodeData.Register._J;
19         ThisSegment.FeedRate := LREAL_TO_REAL(GCodeData.Register._Q);
```

Custom_Code_Execute function block

Add 392 to the CASE statement as shown on lines 9 through 13.

```
6 IF (ThisSegment.SegmentType >= INT#1000) AND (ThisSegment.SegmentType <= INT#1999) THEN
7   CustomGCode:=ThisSegment.SegmentType - INT#1000; (* Convert back to original custom M code for ease of programming *)
8   CASE CustomGCode OF
9     392:
10       MyAppData.WidgetPrm1:=ThisSegment.GenericLREAL[0];
11       MyAppData.WidgetPrm2:=ThisSegment.GenericLREAL[1];
12       MyAppData.WidgetPrm3:=ThisSegment.GenericLREAL[2];
13       MyAppData.WidgetPrm4:=REAL_TO_DINT(ThisSegment.FeedRate);
```

Once the parameter values are loaded into the MyAppData, use the information in other parts of the project to write the values to the Widget as necessary. Yaskawa recommends keeping the customization code in this block to a minimum, and adding any other necessary logic externally.

Example #3 - Wait for an analog value to be greater than or equal to another value.

Assume that custom G-Code command M406 indicates the machine must wait until a temperature has reached or exceeded a set point before continuing. In this example, assume MyAppData.TempSetting has been previously set using another custom code or other method.

Custom_Code_Processor function block

Simply add the custom code to the CASE statement as shown on line 34. This identifies the codes as recognized by system, and prevents the CASE ELSE condition from setting the "UnsupportedMCode" Error condition. There is nothing else to do in the Custom_Code_Processor function. Optionally, the set point could be sent with the custom G-Code as a parameter. In that case, copy the TempSetting into ThisSegment.GenericLREAL[0] similar to the example shown on line 16.

```
12 IF (ThisSegment.SegmentType >= INT#1000) AND (ThisSegment.SegmentType <= INT#1999) THEN
13   (* M Codes must be included here in the CASE statement to signify support, and to prevent the ELSE condition from throwing an error. *)
14   CASE GCodeData.Register._G OF
15     392: (* // Set configuration parameters for the external widget *)
16         ThisSegment.GenericLREAL[0] := GCodeData.Register._H;
17         ThisSegment.GenericLREAL[1] := GCodeData.Register._I;
18         ThisSegment.GenericLREAL[2] := GCodeData.Register._J;
19         ThisSegment.FeedRate := LREAL_TO_REAL(GCodeData.Register._Q);
20
21     123: (* Wait until Oxygen level falls below a specified value *)
22         ThisSegment.GenericLREAL[1]:=INT_TO_LREAL(GCodeData.Register._O);
23
24     555:
25         ThisSegment.GenericLREAL[1]:=GCodeData.Register._P;
26         ThisSegment.ExactStopCheck:=FALSE; (* Set FALSE to force motion ahead and behind this instruction to continue without stopping *)
27
28   ELSE
29     UnsupportedGCode:=TRUE; (* Do not remove this line *)
30   END_CASE;
31 ELSE
32   (* M Codes must be included here in the CASE statement to signify support, and to prevent the ELSE condition from throwing an error. *)
33   CASE GCodeData.Register._M OF
34     406: ;(* Wait until specified temperature is reached *)
35
```

Custom_Code_Execute function block

Add the code as shown in lines 27 and 28. Use the OKToIncrement BOOL variable which is returned to MC_MovePath. Setting this variable according to the comparison of the two values allows MC_MovePath to know when it's OK to proceed.

```
6 IF (ThisSegment.SegmentType >= INT#1000) AND (ThisSegment.SegmentType <= INT#1999) THEN
7   CustomGCode:=ThisSegment.SegmentType - INT#1000; (* Convert back to original custom M code for ease of programming *)
8   CASE CustomGCode OF
9     392:
10        MyAppData.WidgetPrm1:=ThisSegment.GenericLREAL[0];
11        MyAppData.WidgetPrm2:=ThisSegment.GenericLREAL[1];
12        MyAppData.WidgetPrm3:=ThisSegment.GenericLREAL[2];
13        MyAppData.WidgetPrm4:=REAL_TO_DINT(ThisSegment.FeedRate);
14
15        123: (* Wait until Oxygen level falls below a specified value *)
16            OKToIncrement:=MyAppData.OxygenLevel < ThisSegment.GenericLREAL[1];
17
18        555:
19            MyAppData.GasCoolingTemp:=ThisSegment.GenericLREAL[1];
20
21   ELSE
22     UnsupportedGCode:=TRUE; (* Do not remove this line *)
23   END_CASE;
24 ELSE
25   CustomMCode:=ThisSegment.SegmentType - INT#2000; (* Convert back to original custom M code for ease of programming *)
26   CASE CustomMCode OF
27     406: (* Wait until specified temperature is reached *)
28         OKToIncrement:=(MyAppData.TempActual >= MyAppData.TempSetting);
```

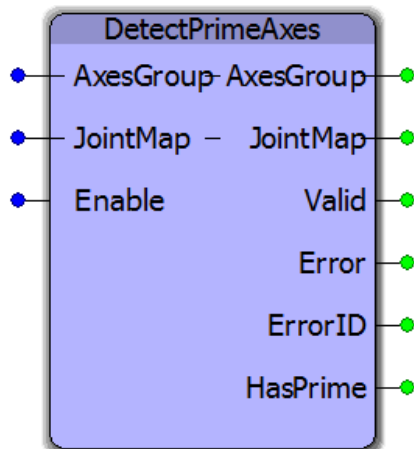
Example #4 Overrides

The ID assigned in the OverrideList function block is added to the base SegmentType of 3000 indicating an override. The original ID can be obtained by subtracting 3000 for the ease of programming (See line 45.) Very little code is required in Custom_Code_Processor, sometimes no code other than to list the ID in the CASE statement to prevent the ELSE condition from returning an Error.

```
41 ELSE
42 (*****
43 (***** Code Overrides / Macros *****)
44 (*****
45 OverrideCode:=ThisSegment.SegmentType - INT#3000; (* Convert back to original Override ID for ease of programming *)
46 CASE OverrideCode OF
47 3: ThisSegment.ExactStopCheck:=TRUE; (* M3: Spindle ON *)
48 5: ThisSegment.ExactStopCheck:=TRUE; (* M5: Spindle OFF *)
49 40: ThisSegment.ExactStopCheck:=TRUE; (* M40: OEM Feature A *)
50 41: ThisSegment.ExactStopCheck:=TRUE; (* M41: OEM Feature B *)
51 50: ThisSegment.ExactStopCheck:=TRUE; (* M50: OEM Feature C *)
52 51: ThisSegment.ExactStopCheck:=TRUE; (* M51: OEM Feature D *)
53 52: ThisSegment.ExactStopCheck:=TRUE; (* M52: OEM Feature E *)
54 ELSE
55 UnPreparedOverride:=TRUE; (* Do not remove this line *)
56 END_CASE;
57 END_IF;
```




DetectPrimeAxes



This function block searches the AxesGroup structure and the JointMap to determine if any joints are configured with additional axes. It is useful for operating other functions in Group Toolbox such as [GroupToHome](#) and [GroupReAlignPrimeAxes](#).

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.	
V	JointMap	JointMap	Structure of data that holds information for Joint Index, Label, AxisName and AXIS_REF.	
VAR_INPUT			Default	
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	HasPrime	BOOL	Set TRUE if the Group contains at least one joint configured with more than one servo.	

Notes

- This block only supports Mechatrolink groups, not remote hosted robots via MotomanSync.

Error Description

This function block will not generate any errors, even if the AxesGroup is invalid.

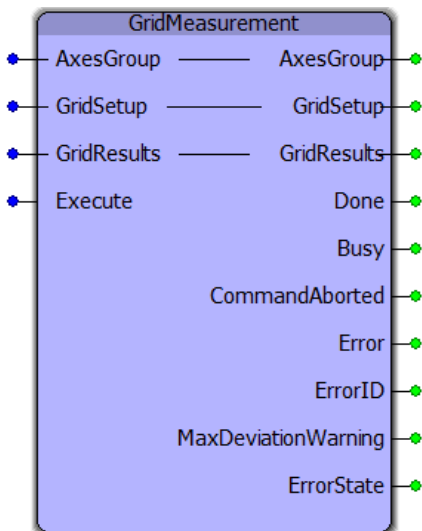
Example

This example shows how GetPrimeAxes is used inside the [Pendant_Driver](#) function block.

```
47 (* For locally hosted robots, read joint map and check if there are prime axes in the mechanism *)
48 (*-----*)
49 IF (AxesGroup.Host_ID < INI#1) THEN (* Only Mechatrolink groups checked for prime axes *)
50     IF NOT (Init2Complete) THEN
51         ReadJointMap_1(AxesGroup:=AxesGroup, JointMap:=JointMap, Execute:=R_TRIG_Enable.Q);
52         AxesGroup:=ReadJointMap_1.AxesGroup;
53         JointMap:=ReadJointMap_1.JointMap;
54         IF ReadJointMap_1.Done THEN
55             DetectPrimeAxes_1(AxesGroup:=AxesGroup, JointMap:=JointMap, Enable:=TRUE);
56             AxesGroup:=DetectPrimeAxes_1.AxesGroup;
57             JointMap:=DetectPrimeAxes_1.JointMap;
58             IF DetectPrimeAxes_1.Valid THEN
59                 HasPrime:=DetectPrimeAxes_1.HasPrime;
60                 Init2Complete:=TRUE;
61             ELSIF DetectPrimeAxes_1.Error THEN
62                 ConfigError:=DetectPrimeAxes_1.Error;
63                 ConfigErrorID:=DetectPrimeAxes_1.ErrorID;
64             END_IF;
65         ELSIF ReadJointMap_1.Error THEN
66             ConfigError:=TRUE;
67             ConfigErrorID:=ReadJointMap_1.ErrorID;
68         END_IF;
69     END_IF;
70 ELSE
71     Init2Complete:=TRUE;
72 END_IF;
```




GridMeasurement



This function block moves a group to measure a grid / plane surface by touching the grid surface at an array of locations. The recorded GridResults can then be written to a file using [WriteGridFile](#) / [ReadGridFile](#), and ultimately used to provide on the fly offset compensation using GridLookup in conjunction with Y_DirectControl.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.
V	GridSetup	GridSetupStruct	Structure of data that provides information for measuring the specified plane.
V	GridResults	GridResultsStruct	Contains the measurement information which can be written to a file and referenced with the GridLookup function block.
VAR_INPUT			Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
			FALSE

VAR_OUTPUT			
B	Done	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
E	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	MaxDeviationWarning	BOOL	Reports if any of the measured points deviate from another by more than GridSetup.MaxDeviation. Can be used as a test for measurement validity.
V	ErrorState	INT	If an Error is reported, this is the code from the internal state machine which may be helpful when debugging.

Notes

- This block will detect if joint has a secondary servo(s), and monitor the torque of all servos configured when moving the joint to the touch positions. The first motor to indicate a torque limit will cause motion to stop and the measurement to be taken. A maximum of three servos operating the joint (as secondary axes) are supported.
- The only mode supported is "measurement by torque limited contact with the surface." The other modes are reserved for future enhancement.
- See related function blocks: [WriteGridFile](#), [ReadGridFile](#), GridLookup.

Error Description

See the [Function Block ErrorID](#) list.

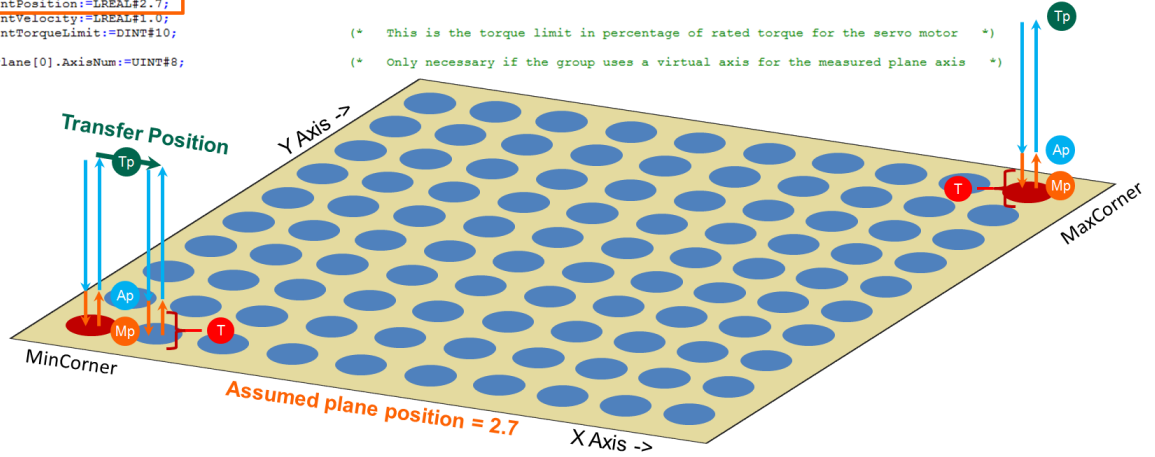
Example

Initialization:

```

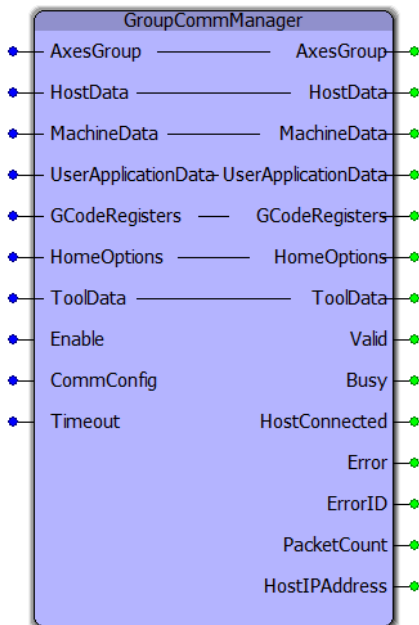
184 gridsetup.MeasureMethod:=GTB_MeasurementType#Torque; (* Use torque limited touch mode *)
185 gridsetup.MeasuredPlane:='Z'; (* Measure the plane which the Z axis contacts *)
186 gridsetup.Tolerance:=LREAL#0.75; (* Measurement position must be within this value of the PlaneSetup.Point[n][3] for the plane axis *)
187
188 gridsetup.Acceleration:=MyMachine.Prms.Acceleration;
189
190 gridsetup.Resolution:=INT#10; (* Make a total of 100 measurements in a 10 by 10 pattern *)
191
192 gridsetup.MinCorner[0]:=LREAL#5.0; (* Coordinate for first measurement *)
193 gridsetup.MinCorner[1]:=LREAL#5.0;
194
195 gridsetup.MaxCorner[0]:=LREAL#230.0;
196 gridsetup.MaxCorner[1]:=LREAL#160.0;
197
198 gridsetup.TransferPosition:=LREAL#20.0; (* Position above plane where the tool moves from one measurement location to the next location *)
199 gridsetup.TransferVelocity:=LREAL#200.0;
200
201 gridsetup.ApproachPosition:=LREAL#6.0; (* Position at which the torque limit is applied in preparation for measuring the plane *)
202 gridsetup.ApproachVelocity:=LREAL#25.0;
203
204 gridsetup.MeasurementPosition:=LREAL#2.7;
205 gridsetup.MeasurementVelocity:=LREAL#1.0;
206 gridsetup.MeasurementTorqueLimit:=DINT#10; (* This is the torque limit in percentage of rated torque for the servo motor *)
207 (* Only necessary if the group uses a virtual axis for the measured plane axis *)
208 GridSetup.ExternalPlane[0].AxisNum:=UINT#8;

```





GroupCommManager



This function block provides a communication interface to GroupComm.DLL to provide support for group operations to PC software such as Compass. It contains the [Pendant_Driver](#) function block, and supports data transfer of other structures such as [MachineStruct](#) and GCodeRegisters.

Library

Group Toolbox

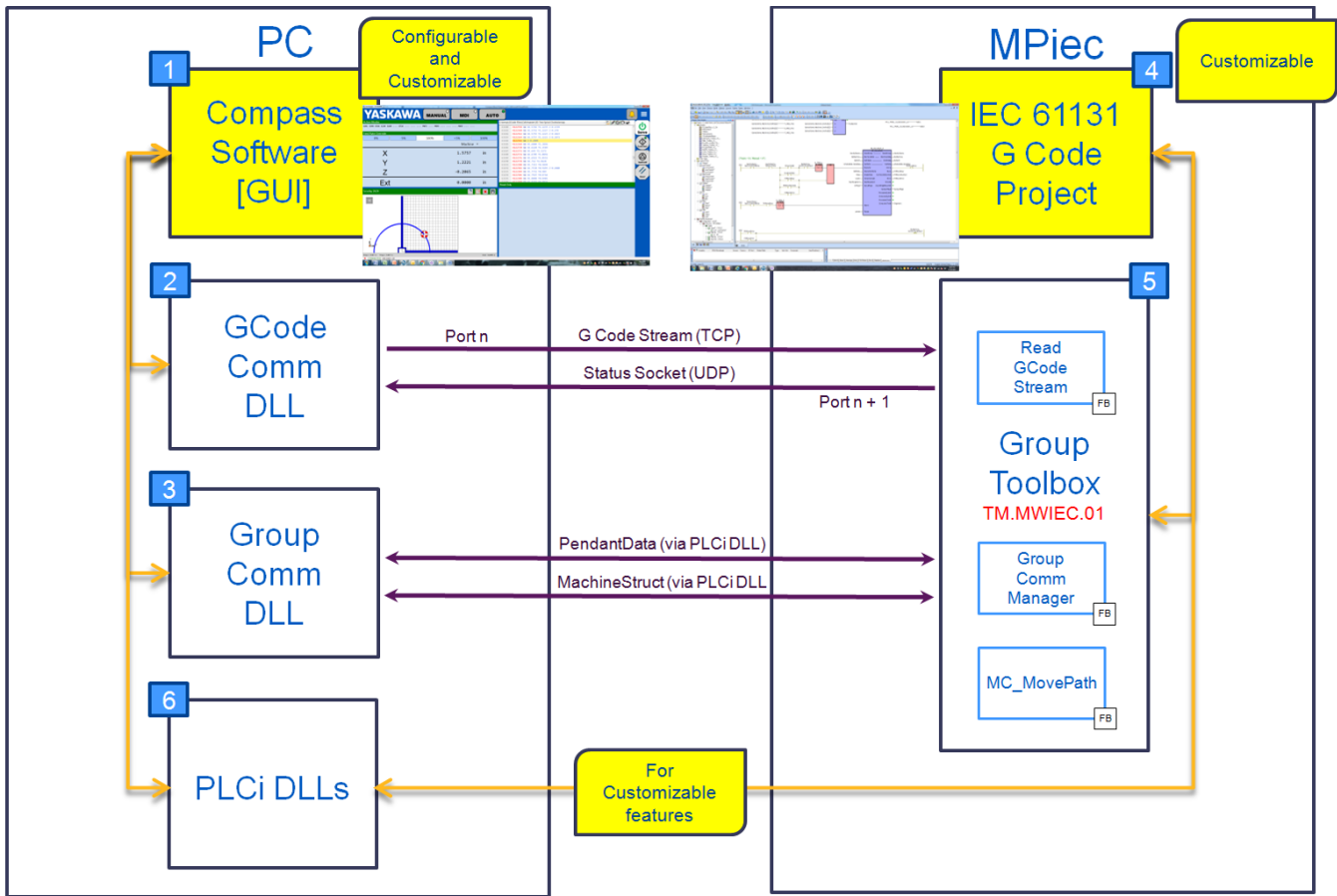
Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.
V	HostData	PendantDataStruct	Contains the data which is shared with the host via GroupComm.DLL.
V	MachineData	MachineStruct	Contains parameters such as maximum velocities and accelerations, and support for 3D printers and tangent axes if required. Also included are Origin (Home) and Part Coordinate System offsets.
V	UserApplicationData	UserApplicationData	Structure which contains configuration, teach point and tool data.

V	GCodeRegisters	GCodeRegister	Structure containing the last value provided by the host (file or stream) for each register (letter of the alphabet.)
V	HomeOptions	HomeOptionsStruct	Data structure containing information for various homing methods and sequences.
V	ToolData	ToolDataStruct	Contains radius and length data for tools that may be selected.
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	CommConfig	CommStruct	Structure which configures this function block to communicate with the GCodeComm.DLL on a PC.
V	TimeOut	TIME	Reserved for future use.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
V	HostConnected	BOOL	Confirms that a host has successfully initiated a connection.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	PacketCount	UDINT	Reports the number of packets received on port
V	HostIPAddress	STRING	The IP address of the device which initiated the connection request to the MPiec controller.

Notes

- This block was designed for use with a PC software interface such as Compass using GroupComm.DLL.
- Yaskawa recommends executing this function block in a 50 mSec task. This has been determined as an ideal update interval for adequate response to jog buttons and other user controls.
- See the [Getting Started with Secondary Axes](#) section.
- This block diagram shows how the GroupCommManager fits into a complete solution:

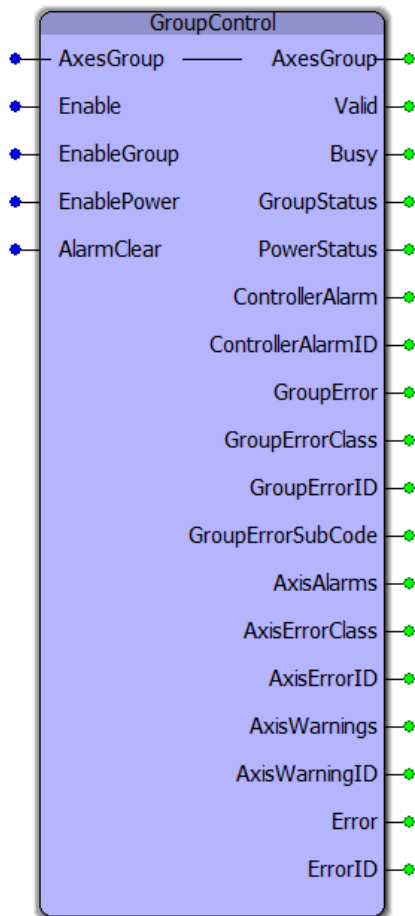


Error Description

See the [Function Block ErrorID](#) list.



GroupControl



This function block operates and monitors several PLCopen Part 4 group related functions including MC_GroupEnable, MC_GroupDisable, MC_GroupReadError, MC_GroupReset, Y_GroupPower and other functions such as Y_ReadAlarms, Y_ClearAlarm.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
---	-----------	-----------	-------------

VAR_IN_OUT				
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional sub-structures pertaining to the group.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	EnableGroup	BOOL	If TRUE, the MC_GroupEnable function block will be executed. If FALSE, the MC_GroupDisable function block will be executed.	FALSE
V	EnablePower	BOOL	If TRUE, all the axes belonging to the AxesGroup will be powered. All axes are first checked for alarms, and if any alarms are present, no axes are powered. If FALSE, all of the axes belonging to the group will be powered down (servo off). NOTE: This level sensitive input will retry to achieve the requested condition every 2 seconds if there is an error preventing its success.	FALSE
V	AlarmClear	BOOL	Executes MC_GroupReset, MC_Reset, and Y_ClearAlarms based on the current alarms related to the AxesGroup.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
V	GroupStatus	BOOL	Indicates if the group is enabled.	
V	PowerStatus	BOOL	Indicates that all axis in the group are powered.	
V	ControllerAlarm	BOOL	Indicates a controller side axis alarm.	
V	ControllerAlarmID	UDINT	Indicates the controller alarm ID number, such as 3302 0018. (shown in hex.) Refer to the Controller AlarmID list in the PLCopenPlus manual for troubleshooting.	
V	GroupError	BOOL	Indicates a group alarm.	
V	GroupErrorClass	UINT	The error class indicates the source of the error. For more information, refer to the Controller Alarm ID List .	
V	GroupErrorID	UINT	Indicates the group alarm ID number. Refer to the Controller Alarm ID list .	
V	GroupErrorSubCode	UDINT	If using a remote hosted Robot Controller via MotomanSync, this output reports additional error detail.	
V	AxisAlarms	BOOL	Indicates an axis alarm.	
V	AxisErrorClass	UINT	Indicates the axis alarm ID number. See notes for more information.	
V	AxisErrorID	UINT	Indicates the axis alarm ID number. See MC_ReadAxisError in the PLCopenPlus help manual for more information.	
V	AxisWarnings	BOOL	Indicates an axis warning.	
V	AxisWarningID	UINT	Indicates the axis warning ID number. See notes for more information.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes:

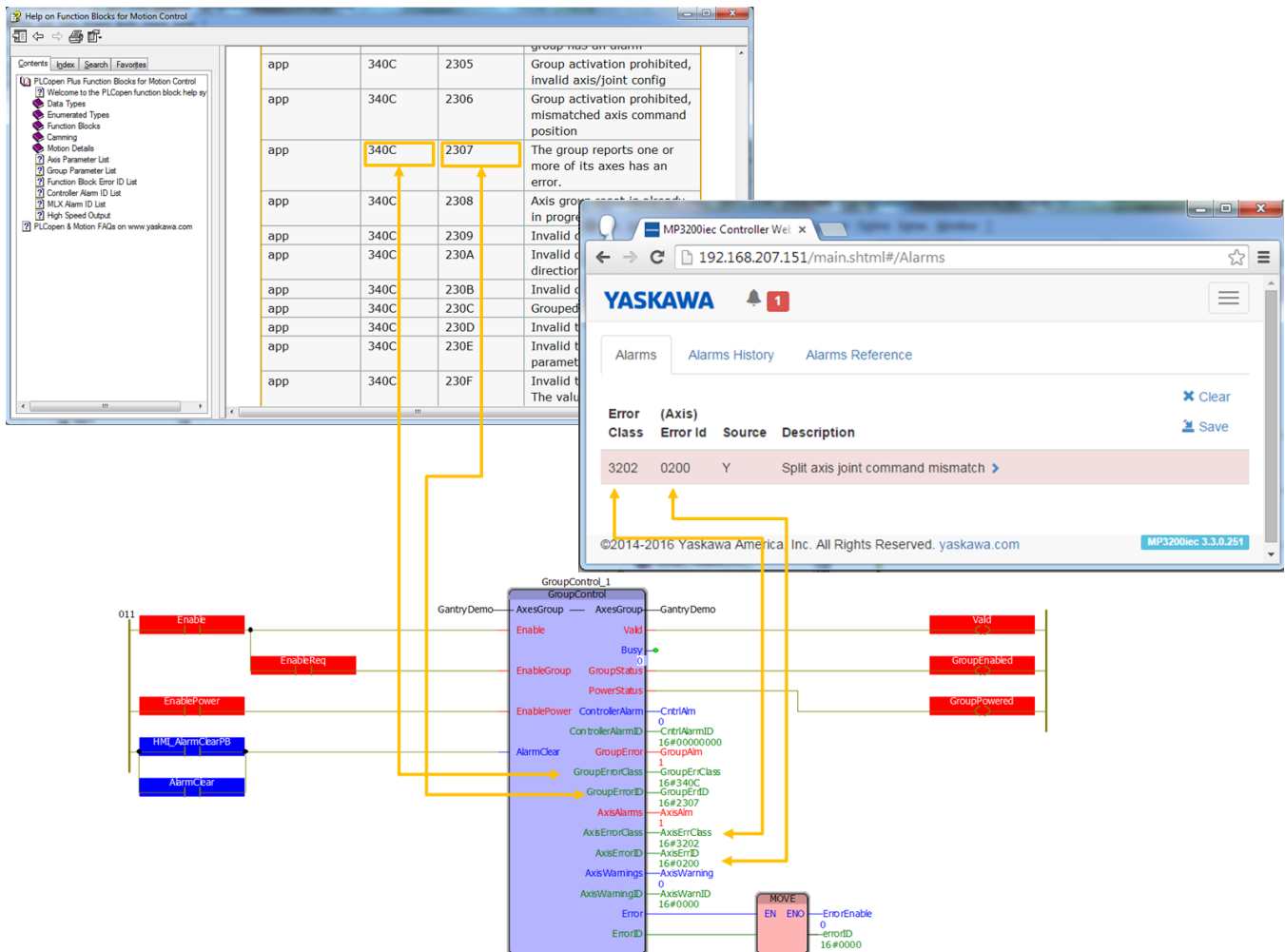
- If ErrorClass = 16#3504 (13572 decimal) then the source of the error is the MLX200. Refer to the [MLX AlarmID List](#) for MLX200 GroupErrorIDs.
- If ErrorClass = 16#3302, 16#3303, 16#4302, or 16#4403, then the source of the error (alarm) is the ServoPack. Sigma alarms are documented in the Sigma Series user manuals. Please refer to the following manuals for details regarding servo amplifier errors to look up the alarm code shown at AxisErrorID output:
- Sigma-7 Mechatrolink-III: [SIEPS8000128](#), see Section 12.2
- Sigma-5 Mechatrolink-III with rotary motor: [SIEPS8000064](#), see Section 9.1
- Sigma-5 Mechatrolink-III with linear motor: [SIEPS8000065](#), see Section 8.1
- Sigma-5 Mechatrolink-II with rotary motor: [SIEPS8000046](#), see Section 9.1
- Sigma-5 Mechatrolink-II with linear motor: [SIEPS8000048](#), see Section 8.1
- If ErrorClass is some value other than 16#3302, 16#3303, 16#3504, 16#4302, or 16#4403, the source of the error is on the MPiec controller side. Refer to the [Controller Alarm ID List](#).
- There is no distinction between Alarms and Errors; they have the same meaning.

Error Description

See the [Function Block ErrorID](#) list.

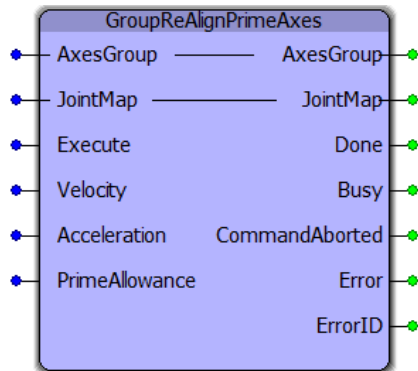
Example

This example shows an error condition when the group has two axes which operate the same joint (Y axis) on a gantry. The commanded position of both axes must be identical before the group can be enabled, otherwise, the alarm condition shown below will result.





GroupReAlignPrimeAxes



This function block searches the AxesGroup structure to determine if any joints are configured with additional prime axes. If all the motors operating a joint are already within the PrimeAllowance, the prime motors are moved to the same commanded position as the main motor. It is useful to prepare a group for MC_GroupEnable and prevent that function from generating the ErrorID 8966.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional sub-structures pertaining to the group.	
V	JointMap	JointMap	Structure of data that holds information for Joint Index, Label, AxisName and AXIS_REF.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
B	Velocity	LREAL	Absolute value of the velocity in user unit-s/second.	LREAL#0.0
B	Acceleration	LREAL	Value of the acceleration in user unit-s/second^2 (acceleration is applicable with same sign of torque and velocity)	LREAL#0.0

V	PrimeAllowance	LREAL	Specify the maximum amount of difference between the main and prime servo commanded positions to permit safe alignment. See the Getting Started with Secondary Axes section.	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
E	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

- This block only supports Mechatrolink groups, not remote hosted robots via MotomanSync.
- The Group's servo power must be applied before executing this block. (Y_GroupPower, or [GroupControl.EnablePower](#))
- **PrimeAllowance must not be set to a large value**, which could cause damage to the machine! It is assumed that the motors were previously aligned and positions were set properly. This function block is only to be used to perform minor realignment motion such that all motors on each joint are at the exact same commanded position in preparation for execution of MC_GroupEnable.

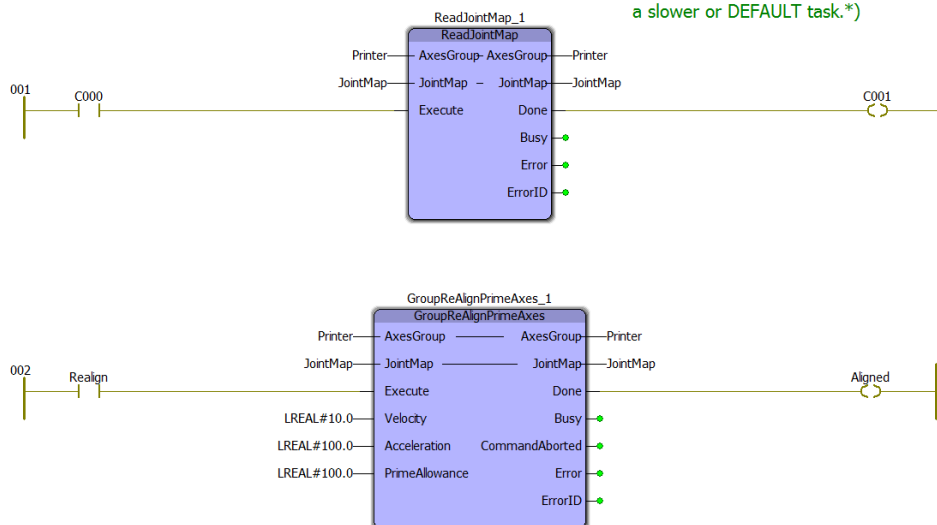
Error Description

See the [Function Block ErrorID](#) list.

Example #1

(*JointMap must be populated before Executing GroupReAlignPrimeAxes*)

(*Only need to Execute ReadJointMap once, it takes some time. Could be executed in a slower or DEFAULT task.*)



Example #2

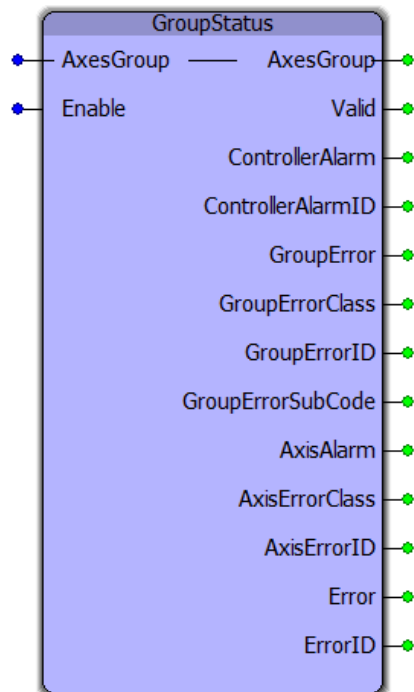
This example is taken from the [Pendant_Driver](#) function block. It's part of a CASE statement which enables the servos and enables the group. The CASE statement operates as follows:

- 1) MC_GroupDisable
- 2) Y_GroupPower
- 3) GroupReAlignPrimeAxes (HasPrime BOOL was set by executing [DetectPrimeAxes](#). See its example.
- 4) MC_GroupEnable (Should now be successful because all secondary axes have same commanded position as main.)

```
600 3: IF HasPrime THEN
601   (* ToDo: Need way for user to specify the values for Velocity, Acceleration, Tolerance *)
602   IF GroupReAlignPrimeAxes_1.Busy THEN
603     ;
604   ELSIF GroupReAlignPrimeAxes_1.Done THEN
605     GroupPowerState:=GroupPowerState + INT#1;
606   ELSIF GroupReAlignPrimeAxes_1.CommandAborted THEN
607     Output[GroupPowerState].CommandAborted:=TRUE;
608     GroupPowerState:=INT#-1;
609   ELSIF GroupReAlignPrimeAxes_1.Error THEN
610     Output[GroupPowerState].Error:=TRUE;
611     Output[GroupPowerState].ErrorID:=GroupReAlignPrimeAxes_1.ErrorID;
612     GroupPowerState:=INT#-1;
613   END_IF;
614   GroupReAlignPrimeAxes_1.Velocity:=UserApplicationData.PrimeAlignment.Velocity;
615   GroupReAlignPrimeAxes_1.Acceleration:=UserApplicationData.PrimeAlignment.Acceleration ;
616   GroupReAlignPrimeAxes_1.PrimeAllowance:=UserApplicationData.PrimeAlignment.Tolerance;
617   GroupReAlignPrimeAxes_1.AxesGroup:=AxesGroup;
618   GroupReAlignPrimeAxes_1.JointMap:=JointMap;
619   GroupReAlignPrimeAxes_1(Execute:=iActive AND (GroupPowerState=INT#3));
620   AxesGroup:=GroupReAlignPrimeAxes_1.AxesGroup;
621   JointMap:=GroupReAlignPrimeAxes_1.JointMap;
622 ELSE
623   GroupPowerState:=GroupPowerState + INT#1;
624 END_IF;
```




GroupStatus



This function block monitors all alarms from various areas of the system which could affect group operation. It is used within the [GroupControl](#) function block.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional sub-structures pertaining to the group.
VAR_INPUT			Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors. FALSE

VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	ControllerAlarm	BOOL	Indicates a controller side axis alarm.
V	ControllerAlarmID	BOOL	Indicates the controller alarm ID number, such as 3302 0018. (shown in hex.) Refer to the Controller AlarmID list in the PLCopenPlus manual for troubleshooting.
V	GroupError	BOOL	Indicates a group alarm.
V	GroupErrorClass	BOOL	The error class indicates the source of the error. For more information, refer to the Controller Alarm ID List .
V	GroupErrorID	BOOL	Indicates the group alarm ID number.
V	GroupErrorSubCode	UDINT	If using a remote hosted Robot Controller via MotomanSync, this output reports additional error detail.
V	AxisErrorClass	BOOL	Indicates an axis alarm.
V	AxisErrorID	BOOL	Indicates an axis warning.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes:

- If ErrorClass = 16#3504 (13572 decimal) then the source of the error is the MLX200. Refer to the [MLX AlarmID List](#) for MLX200 GroupErrorIDs.
- If ErrorClass = 16#3302, 16#3303, 16#4302, or 16#4403, then the source of the error (alarm) is the ServoPack. Sigma alarms are documented in the Sigma Series user manuals. Please refer to the following manuals for details regarding servo amplifier errors to look up the alarm code shown at AxisErrorID output:
 - Sigma-7 Mechatrolink-III: [SIEPS8000128](#), see Section 12.2
 - Sigma-5 Mechatrolink-III with rotary motor: [SIEPS8000064](#), see Section 9.1
 - Sigma-5 Mechatrolink-III with linear motor: [SIEPS8000065](#), see Section 8.1
 - Sigma-5 Mechatrolink-II with rotary motor: [SIEPS8000046](#), see Section 9.1
 - Sigma-5 Mechatrolink-II with linear motor: [SIEPS8000048](#), see Section 8.1
- If ErrorClass is some value other than 16#3302, 16#3303, 16#3504, 16#4302, or 16#4403, the source of the error is on the MPiec controller side. Refer to the [Controller Alarm ID List](#).
- There is no distinction between Alarms and Errors; they have the same meaning.

Error Description

See the [Function Block ErrorID](#) list.



GroupToHome



This function block moves a group to its home location in the Coordinate system specified in a sequence specified in GroupHomeData. Prepare GroupHomeData to indicate the required sequence of operation for safe motion to the home position to avoid obstacles, if required, as well as the home position. This function does not locate home switches or overtravels, it assumes that the system has been previously calibrated or homed, and returns to the positions configured in GroupHomeData. It is intended for use with Absolute Encoders. It can also be used with systems that home to sensors, but that operation is not included here, and must be previously executed before GroupToHome can be used.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.	
V	GroupHomeData	GroupHomeStruct	Structure of data that holds information for how the group will be moved to its home (Origin).	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE

E	CoordSystem	MC_CoordinateSystem	Enumeration with the following values: 0 = ACS, 1 = MCS, 2 = PCS, 3 = TCS, 4 = WCS, 5 = TPCS. MC_CoordinateSystem#ACS MC_CoordinateSystem#MCS MC_CoordinateSystem#PCS MC_CoordinateSystem#TCS MC_CoordinateSystem#WCS MC_CoordinateSystem#TPCS	MC_CoordinateSystem#ACS
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
C	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
C	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

- This block will detect joints that have secondary (prime) servos, and will pre align the servos sharing a load prior to starting the home sequence. If pre alignment must be performed, this block will execute MC_GroupDisable, [GroupRealignPrimeAxes](#), and MC_GroupEnable prior to starting the home sequence.
- See the [Getting Started with Secondary Axes](#) section.
- There will be a slight delay until motion starts after executing this function as it determines if there are any joints with prime axes to realign prior to Executing MC_GroupEnable.

Error Description

See the [Function Block ErrorID](#) list.

Example

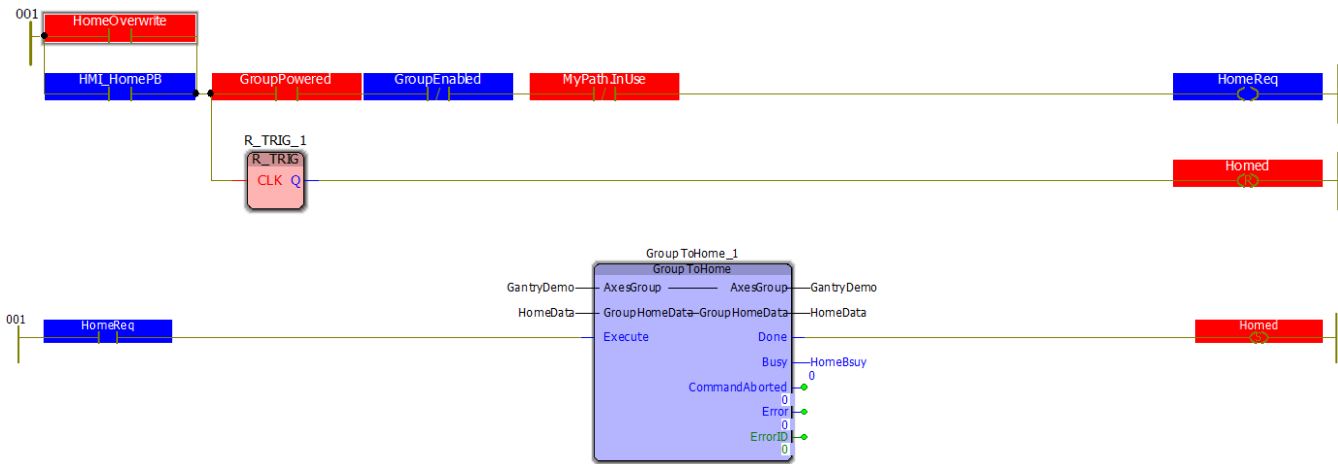
This example homes a three axis gantry by first raising the Z upward to 12mm , then moving the X & Y axes to their home positions.

```

158 MyMachine.Origin[1]:=LREAL#0.0;          (* The X home position is 0 mm *)
159 MyMachine.Origin[2]:=LREAL#0.0;          (* The Y home position is 0 mm *)
160 MyMachine.Origin[3]:=LREAL#12.0;        (* The Z home position is 12 mm *)
161
162 FOR d:=INT#1 TO INT#3 DO
163     HomeData.DOF[d].Position:=MyMachine.Origin[d];
164 END_FOR;
165
166 HomeData.Sequence[1].DOF[1] :='Z';
167 HomeData.Sequence[1].Velocity:=LREAL#150.0;;
168 HomeData.Sequence[1].Acceleration:=LREAL#150.0;
169 HomeData.Sequence[1].LastDOF:=INT#1;      (* Only the Z axis should move in the first sequence *)
170
171 HomeData.Sequence[2].DOF[1] :='X';
172 HomeData.Sequence[2].DOF[2] :='Y';
173 HomeData.Sequence[2].Velocity:=LREAL#150.0;;
174 HomeData.Sequence[2].Acceleration:=LREAL#150.0;
175 HomeData.Sequence[2].LastDOF:=INT#2;      (* The XY axes move in the second sequence *)
176 HomeData.LastSequence:=INT#2;
177
178 HomeData.PrimeAllowance:=LREAL#2.0;      (* mm of deviation allowed if need to align prime axes *)

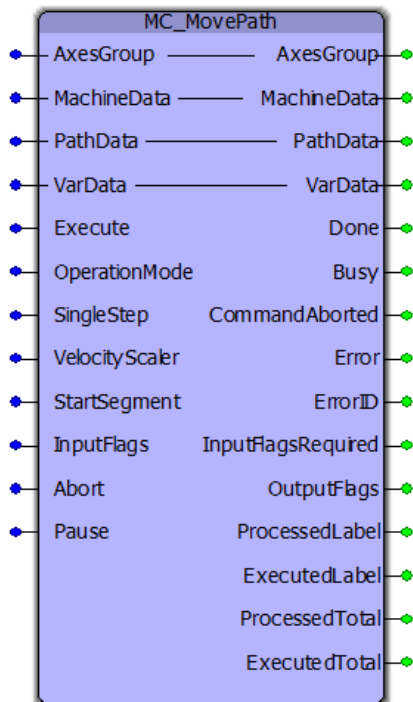
```

Example logic to start the home process.





MC_MovePath



This function block reads the information in the PathData struct and performs the required actions such as interpolated motion, waiting for inputs, setting outputs, etc. The PathData struct is typically populated by various sources, such as [Read_GCode_File](#) and [Read_GCode_Stream](#), although it is not required that G-Code be used to populate [MC_PATH_DATA_REF](#). Custom path generators can also be created.

Library

Group Toolbox

Parameters

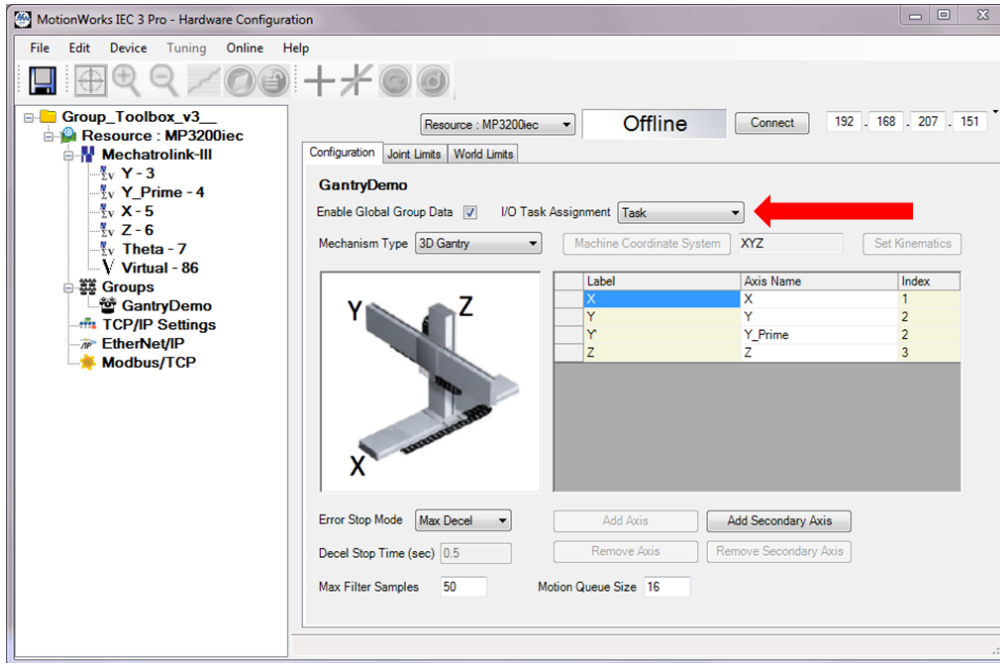
*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional sub-structures pertaining to the group.	
V	MachineData	MachineStruct	Contains parameters such as maximum velocities and accelerations, and support for 3D printers and tangent axes if required. Also included are Origin (Home) and Part Coordinate System offsets.	
V	PathData	MC_PATH_DATA_REF	Structure of data that contains the details for executing a path sequence. Typically PathData is populated by a source function such as Read_GCode_File , Read_GCode_Stream , but simple sequences could also be populated as a static script in ST code.	
V	VarData	VariableArray	An array of LREAL data which is only used with G-Code applications where registers contain variable data. For such applications, connect GCodeData.Variables from the Read_GCode_File function block to this VAR_IN_OUT. If this data is not applicable to the application, connect a dummy variable to satisfy the compiler.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	OperationMode	INT	Enumeration with the following meanings: GTB_OperationMode#Forward GTB_OperationMode#SingleStepFwd GTB_OperationMode#Neutral GTB_OperationMode#SingleStepRev (Future support) GTB_OperationMode#Reverse (Future support) GTB_OperationMode#InfiniteRepeat See Notes below for details.	GT_OperationMode#Forward
V	SingleStep	BOOL	When OperationMode is set for SingleStepFwd, the function will execute one Segment from the PathData at each rising edge of SingleStep.	FALSE
V	VelocityScaler	REAL	The 'FeedRate' element of each Segment is multiplied by the VelocityScaler Input. Changes only affect new motion segments executed after a change to VelocityScaler, previous segments already in the motion buffer will be executed at the Feedrate and VelocityScaler specified at the time they were processed. To change the velocity of the path segments already in the motion buffer at the firmware layer, use the MC_GroupSetOverride function block.	REAL#100.0
V	StartSegment	INT	The first Segment of the PathData to use upon the rising edge of the Execute input. For fault recovery, set this to a non zero value to resume a path that was interrupted. Refer to PathData.Buffer.ExecutedSegment. Capture this value in the event of an Error or other interruption of the path and set it as the StartSegment to recover a path in progress.	INT#0

V	InputFlags	DWORD	Specify up to 32 digital inputs which can be used to control path operations in conjunction with the following SegmentTypes: GTB_SegmentType#WaitForInputs GTB_SegmentType#LoopDecision GTB_SegmentType#BranchDecision GTB_SegmentType#NonBlockingInputCheck See Examples below.	DWORD#0
V	Abort	BOOL	Stops executing Segments and uses MC_GroupStop to stop any ongoing motion. The CommandAborted Output will be set and motion cannot be started again until the Execute Input is toggled.	FALSE
V	Pause	BOOL	This input puts MC_MovePath into an idle state where it will no longer process new segments, but segments already executed and in the motion buffer will be unaffected and continue until the buffer is empty. This may result in a delay from the time Pause is set high until the group physically stops. To Pause or "Interrupt" the group immediately, use the MC_GroupInterrupt function block separately in the MotionWorks IEC project.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
E	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	InputFlagsRequired	DWORD	For use as a debugging tool. This Output reports the InputConditions of the current Segment. For example, if the machine appears to be stuck, it is possible that the active Segment is waiting for a specific set of InputFlags to match before proceeding.	
V	OutputFlags	DWORD	PathData.Segment[].OutputFlags are copied to the OutputFlags Output based on the following conditions: 1) If motion is active, the OutputFlags associated with the Segment responsible for ongoing motion is applied. This technique relies on Group Parameters 2201 and 2202. 2) When there is no ongoing motion, the OutputFlags of the Segment being processed are applied.	
V	ProcessedLabel	YTB_STRING16	If PathData.Segment[].Label was populated by the source, this Output reports a descriptive name of the Segment being processed. If this is a motion segment, the motion instruction is executed, but it does not mean that it is physically causing motion. It may be buffered behind many other motion instructions based on the path data.	

V	ExecutedLabel	YTB_ STRING16	If PathData.Segment[].Label was populated by the source, this Output reports a descriptive name of the Segment actually causing motion. In the case of other non motion Segment Types, ProcessedLabel and ExecutedLabel are likely equal to each other.
V	ProcessedTotal	UDINT	Running count of the number of PathData.Segments[] processed since the rising edge of the Execute input. This value will likely be ahead of ExecutedTotal based on the path characteristics, and the size of the Group motion queue as configured in the Hardware Configuration.
V	ExecutedTotal	UDINT	Running count of the number of PathData.Segments[] executed since the rising edge of Execute. The difference between the ProcessedTotal and ExecutedTotal is the number of motion blocks still in the motion buffer.

Notes

- Yaskawa recommends executing this function block in a fast task such as 4 or 8 mSec. Factors to consider when determining the ideal task interval include any InputFlag / OutputFlag usage, and the motion segment density relative to required velocity.
- This function block references information in AxesGroup.Status, which is updated by the firmware at the task interval configured in the Hardware Configuration. The IO task assignment must be set to the same or faster task interval as where the MC_MovePath function is executing, or errors such as 4369 may occur.



Additional information about OperationMode:

- It is possible to change ControlMode on the fly.
- When using GT_OperationMode#SingleStepFwd, if the SingleStep input is pulsed before a motion Segment has completed, the request is not buffered and will be ignored.
- GT_OperationMode#InfiniteRepeat will permit MC_MovePath to remain Busy indefinitely (Done will not occur.) When the function reaches the last Segment as identified by comparing (PathData.Buffer.UsePointer = PathData.Fin-alSegment) it will immediately continue at the beginning of the PathData again. The advantage of this feature is to provide continuous motion for cyclic operations and improve overall cycle time (OEE). This mode is only available when the PathData has not been loaded via Ethernet stream. The total Path must be contained within the PataData structure without being overwritten, otherwise an Error will be generated.

G Code Support:

- Work Coordinate Offsets G54 through G59.3 are stored in the [MachineStruct.CoordinateSystem.Offset](#). The offsets can be updated by the application program via an HMI or PC, or from the G-Code data itself via the G10 command.
- Tool Compensation: T0 ~ T20, G40, G41, G42. Tool data must be loaded in the ToolStruct before executing [Read_GCode_File](#) or [Read_GCode_Stream](#). Select a tool using the T command.

Error Description

See the [Function Block ErrorID](#) list.

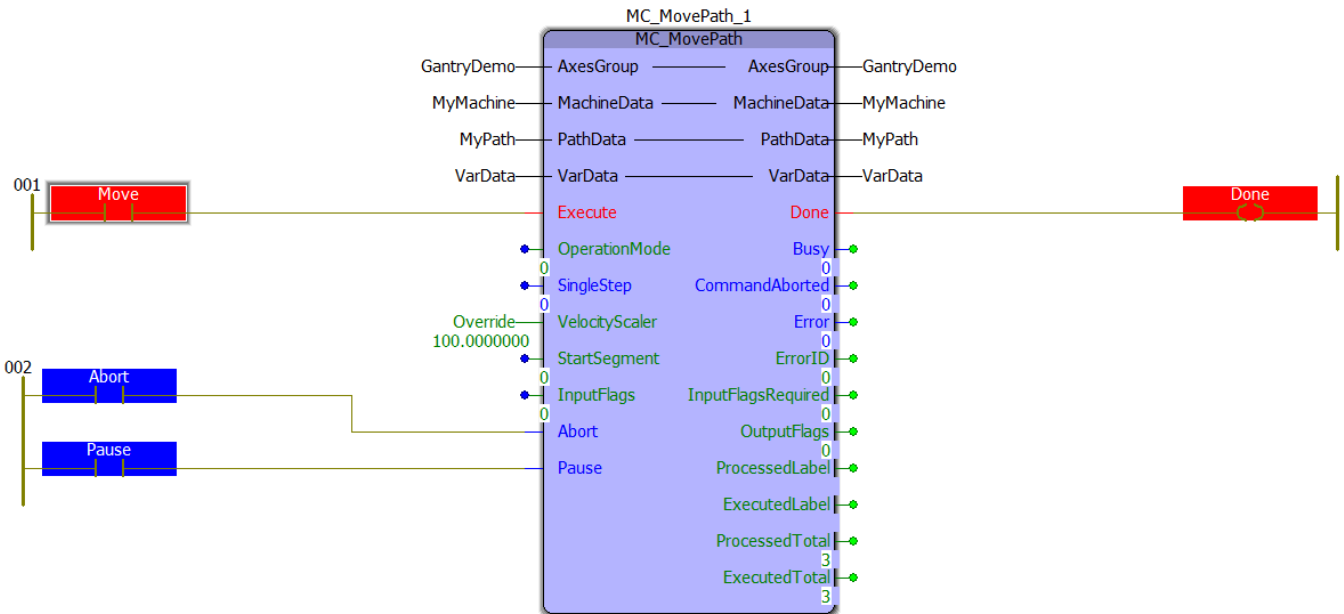
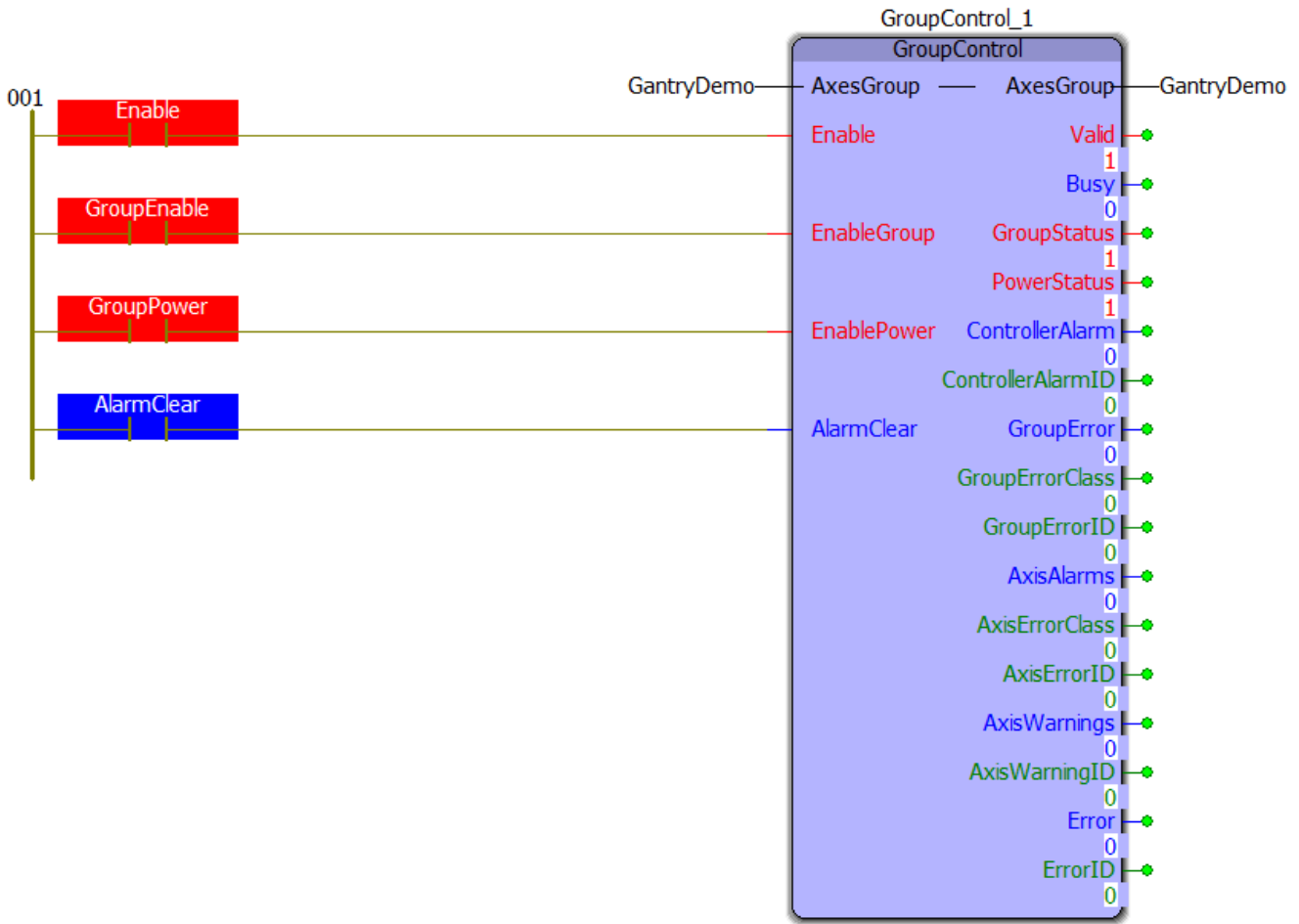
Example1 - Minimum setup required to get started

To learn how to configure the data to operate this block, this example includes hard coded initialization of the related data structures.

NOTE! If hard coding a sequence, starting in v350, the AxesFlags WORD must be set to indicate the coordinates that are being specified for each segment.

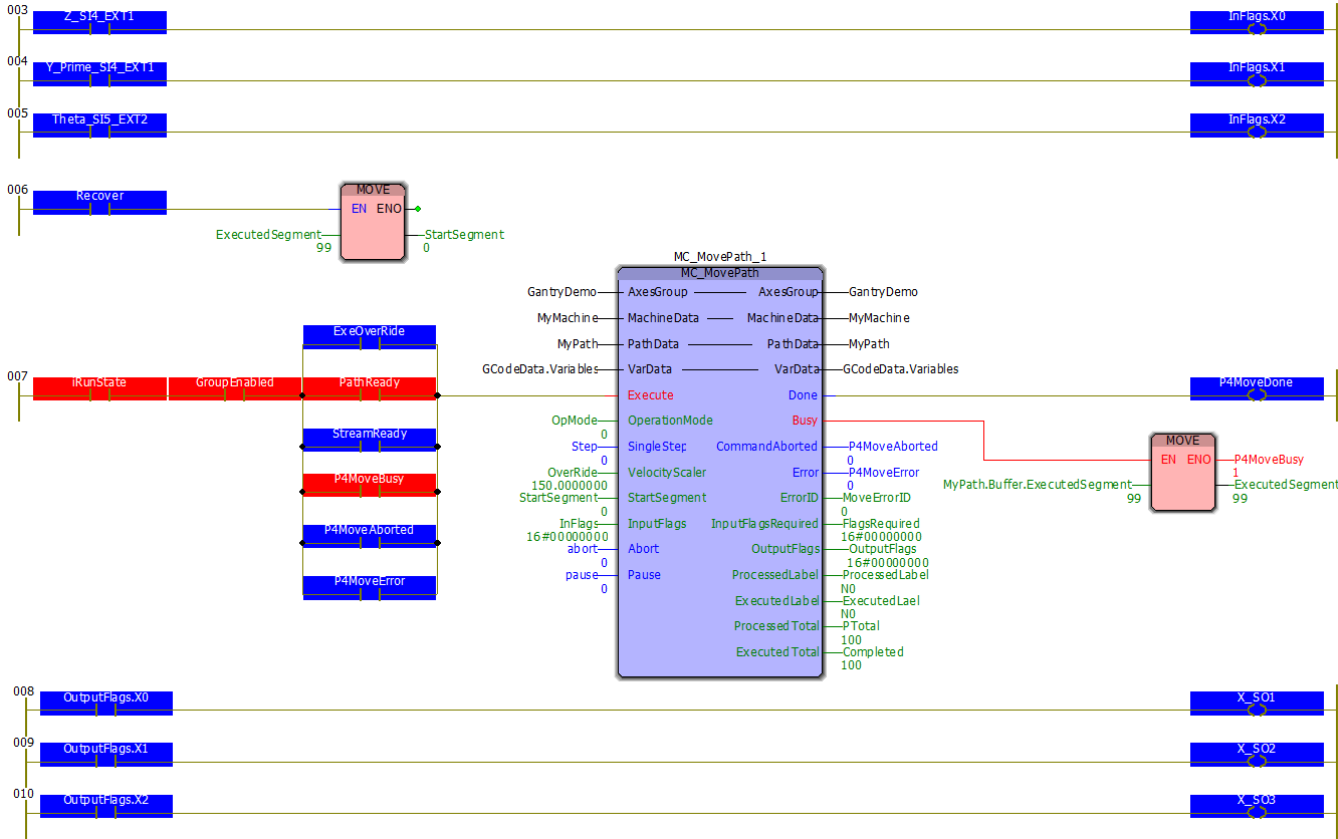
For example: `MyPath.Segment[x].AxesFlags:=WORD#16#3;` (* Bit codes represent Axes as Rz Ry Rx Z Y X = bits 5 4 3 2 1 0 *) (Not shown in the examples below.) If these are not set, no motion will occur.

```
1  (* Set machine parameters *)
2  MyMachine.Prms.Acceleration:=LREAL#100.0;
3  MyMachine.Prms.Deceleration:=LREAL#100.0;
4  MyMachine.Prms.MaxVelocity:=LREAL#250.0;
5  MyMachine.Prms.MaxAcceleration:=LREAL#300.0;
6  MyMachine.Prms.MaxDeceleration:=LREAL#300.0;
7  MyMachine.Prms.MaxSegmentsPerScan:=INT#1; (* Group Toolbox v340 will default to INT#1 if not set *)
8  MyMachine.MachineType:=GTB_MachineType#Printer;
9
10
11
12  MyPath.Colinearity:=LREAL#30.0; (* Degrees *)
13
14  (* Sample minimum Path information for straight line *)
15  x:=INT#0;
16
17  MyPath.Segment[x].SegmentType:=GTB_SegmentType#StraightLine;
18  MyPath.Segment[x].AbsoluteMode:=TRUE;
19  MyPath.Segment[x].CoordSystem:=MC_CoordinateSystem#MCS;
20  MyPath.Segment[x].FeedRate:=REAL#34.5; (* If FeedRate is not specified, then MachineData.MaxVelocity will be used *)
21  MyPath.Segment[x].X:=LREAL#3.5;
22  MyPath.Segment[x].Y:=LREAL#1.5;
23  x:=x + INT#1;
24
25
26  MyPath.Segment[x].SegmentType:=GTB_SegmentType#StraightLine;
27  MyPath.Segment[x].AbsoluteMode:=TRUE;
28  MyPath.Segment[x].CoordSystem:=MC_CoordinateSystem#MCS;
29  MyPath.Segment[x].X:=LREAL#8.25;
30  MyPath.Segment[x].Y:=LREAL#3.75;
31  x:=x + INT#1;
32
33
34  MyPath.Segment[x].SegmentType:=GTB_SegmentType#StraightLine;
35  MyPath.Segment[x].AbsoluteMode:=TRUE;
36  MyPath.Segment[x].CoordSystem:=MC_CoordinateSystem#MCS;
37  MyPath.Segment[x].X:=LREAL#0.5;
38  MyPath.Segment[x].Y:=LREAL#21.5;
39  x:=x + INT#1;
40
41
42  MyPath.Segment[x].SegmentType:=GTB_SegmentType#StraightLine;
43  MyPath.Segment[x].AbsoluteMode:=TRUE;
44  MyPath.Segment[x].CoordSystem:=MC_CoordinateSystem#MCS;
45  MyPath.Segment[x].X:=LREAL#7.5;
46  MyPath.Segment[x].Y:=LREAL#5.5;
47
48  MyPath.FinalSegment:=x;
49  MyPath.FillMethod:=GTB_DataSource#File; (* Group Toolbox v340 can operate with default of GTB_DataSource#na *)
50  MyPath.Buffer.StorePointer:=x; (* If using Group Toolbox v340 and FillMethod GTB_DataSource, this setting is unnecessary *)
```



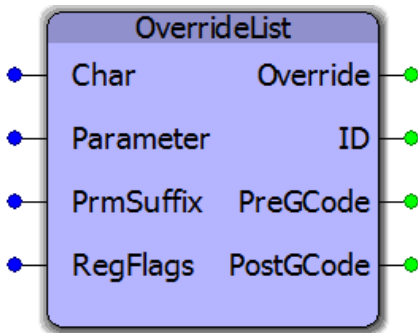
Example 2

This example shows a typical method to map physical inputs and outputs to the data used with the function block. See InFlags and OutFlags usage.





OverrideList



This is a special function block which must be copied and pasted from the Group Toolbox to the Logical POU tree of the main project for customization. Its purpose is to provide a way for the OEM to declare supported G-Codes that are NOT to be executed by the Group Toolbox, but via IEC code programmed by the OEM. While any command character can be declared, there are design limitations to the override feature, which is to allow the OEM to add machine specific operational sequences to features such as Tool changers and Spindle sequences. It is not designed to substitute core operational features such as the G1 command.

Library

Group Toolbox

Parameters

The function block interface is shown for reference only. There is no need for the OEM to include this function block in the main project nor provide VAR_INPUTS or receive VAR_OUTPUTS from this function block. It's called from within the G-Code processor to determine if overrides are declared. The VAR_OUPTUTS report back to the G-Code Processor based on the CASE statement populated by the OEM.

*	Parameter	Data Type	Description	
VAR_INPUT				Default
V	Char	INT	The Alpha character of the command to be overridden. Example - the 'G' in G28.	n/a
V	Parameter	INT	The numeric portion of the command to be overridden. Example - the '28' in G28.	n/a
V	PrmSuffix	INT	The numeric portion of the command to be overridden. Example - the '1' in L30.1.	n/a
V	RegFlags	DWORD	Status bits indicating which register(s) were provided on a given line of G-Code. Bit 0 is set for A, bit 1 for B and so on, using 26 of the 32 bit DWORD.	n/a

VAR_OUTPUT			
V	Override	BOOL	Result of the override check which signals the G-Code Processor.
V	ID	BYTE	When adding override to the CASE statement, the OEM must provide a unique number to each of the overrides declared. This ID will be passed to the Custom_Code_Execute and Custom_Code_Processor function blocks.
V	PreGCode	STRING	An optional command executed prior to passing control to Custom_Code_Execute for the OEM to handle the override. For example, if a tool changer is programmed, it might be necessary to first disable tool height compensation by pre executing a G49.
V	PostGCode	STRING	An optional command executed after passing control to Custom_Code_Execute. For example, if a tool changer is programmed, it might be necessary to re enable tool height compensation with a G43.

Procedure

1. Open the main project in MotionWorks IEC.
2. Open a second copy of MotionWorks IEC, and open the Group Toolbox, typically from the "C:\User-s\Public\Documents\MotionWorks IEC 3 Pro\Libraries" folder. It is also possible to locate the Group Toolbox referenced in your project by right clicking on the Group Toolbox User Library and viewing its Properties.
3. In the Project Tree Window of the second instance of MotionWorks IEC, locate the OverrideList function block in the Project Tree Window.
4. Copy it to the main project by right clicking on the Logical POU's section of the Project Tree Window and Pasting.
5. Important - Even though this function is called internally by the G-Code processor, it is necessary to add it to a POU in the main project so the compiler knows which instance to reference.
6. Add overrides to the CASE statement as necessary.

Example

Focusing on line 19, this sets an override for an M41 command, and assigns a unique ID to identify the override in the Custom_Code_Execute function block.

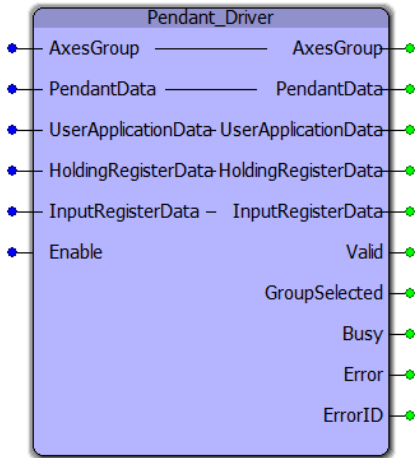
```

1  (*****
2  (*****                                     How to use this function block
3  (*****                                     *****)
4  (* #1 - For any supported code that is to be handled as a custom code instead, add the command number to the appropriate list. *)
5
6  Override      :=FALSE;                      (* Initialize *)
7  PreGCode     := STRING#'';                 (* Initialize *)
8  PostGCode    := STRING#'';                 (* Initialize *)
9
10 (* Only add (Char) ASCII codes for thr letters which have overrides required. *)
11 (* Set the 'Overridden' output for codes than are to be overridden. *)
12 CASE Char OF
13   65: (* ASCII A *);
14   77: (* ASCII M *)
15     CASE Parameter OF
16       3: Override:=TRUE;   ID:=BYTE#3;      (* M3: Spindle ON *)
17       5: Override:=TRUE;   ID:=BYTE#5;      (* M5: Spindle OFF *)
18       40: Override:=TRUE;  ID:=BYTE#40;     (* M40: OEM function A *)
19       41: Override:=TRUE;  ID:=BYTE#41;     (* M41: OEM function B *)
20       50: Override:=TRUE;  ID:=BYTE#50;     (* M50: OEM function C *)
21       51: Override:=TRUE;  ID:=BYTE#51;     (* M51: OEM function D *)
22       52: Override:=TRUE;  ID:=BYTE#52;     (* M52: OEM function E *)
23     END_CASE;
24   84: (* ASCII T *)
25     CASE Parameter OF
26       0:   Override:=TRUE;   ID:=BYTE#100;   (* T0: Unload Tool *)
27           PostGCode     := STRING#'G49';
28       1..99: Override:=TRUE; ID:=BYTE#100 + INT_TO_BYTE(Parameter); (* T1-T99: Load Change *)
29           ID:=BYTE#100 + INT_TO_BYTE(Parameter); (* Parameter gives the number of tool to load.
30           PostGCode     := STRING#'G43';      Execute code extracts tool to load by subtracting 100 from ID. *)
31     END_CASE;
32 END_CASE;
33

```




Pendant_Driver



The Pendant_Driver function block provides a communication link to a pendant (or PC application via GroupComm.DLL) to perform group related operations. The Pendant_Driver supports operations such as jogging, teaching, part frame definition, tool data definition, saving and recalling taught data, etc. Feedback and status data are sent to a pendant via the InputRegisterData structure. Teach point data and part frames are accessible via the UserApplicationData structure.

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.	
V	PendantData	PendantDataStruct	Structure that holds taught data like job points, part frame coordinates, tool data, interference zone definitions etc. This data is shared between the Pendant and the IEC Application program.	
V	UserApplicationData	UserApplicationData	Structure that holds taught data like job points, part frame coordinates, tool data, interference zone definitions etc. This data is shared between the Pendant and the IEC Application program.	
V	HoldingRegisterData	HoldingRegisters_ByteArray	Structure of read/write data which primarily contains commands sent from a pendant to the MPiec controller. The elements of this structure are used to perform group tasks like jogging, teaching etc.	
V	InputRegisterData	InputRegisters_ByteArray	Structure of read data which contains feedback and status information about the group.	
VAR_INPUT			Default	
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				

B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	GroupSelected	BOOL	Indicates that the group name entered on the pendant matches the name configured for the AxesGroup.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

- This function block generates two types of Errors. Logical errors (such as wrong control mode or group not enabled) are displayed on the pendant or host and are **not** reported at the VAR_OUTPUT of this function block. Other errors, such as a loss of communication are output as a traditional PLCopen function block Error.

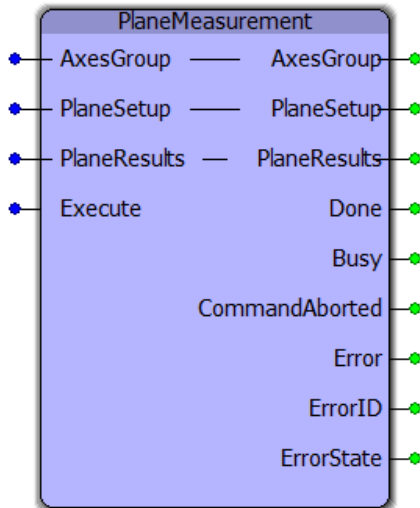
Error Description

See the [Function Block ErrorID](#) list.

Example



PlaneMeasurement



This function block moves a Group to find a plane by measuring three user specified locations. The recorded results are available for use with other functions, such as Y_GroupSetFrameOffset.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.
V	PlaneSetup	PlaneSetupStruct	Structure of data that provides information for measuring the specified plane.
V	PlaneResults	PlaneResultsStruct	Contains the measurement positions and the Offset which can be used with Y_GroupSetFrameOffset.
VAR_INPUT			Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
VAR_OUTPUT			

B	Done	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	ErrorState	INT	If an Error is reported, this is the code from the internal state machine which may be helpful when debugging.

Notes

- This block will detect if joint has a prime axis, and will monitor the torque of all axes involved when moving the joint to the touch positions. The first motor to indicate a torque limit will cause motion to stop and the measurement to be taken. A total of three servos operating a single joint is supported.
- Only Measurement by torque limited contact with the surface is supported as of v350.

Error Description

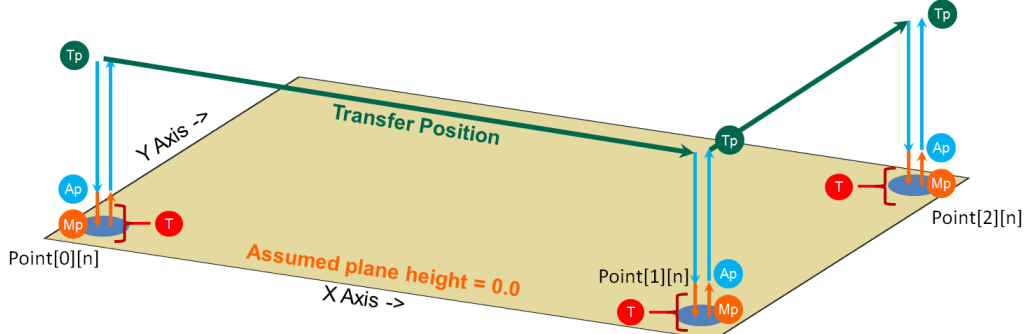
See the [Function Block ErrorID](#) list.

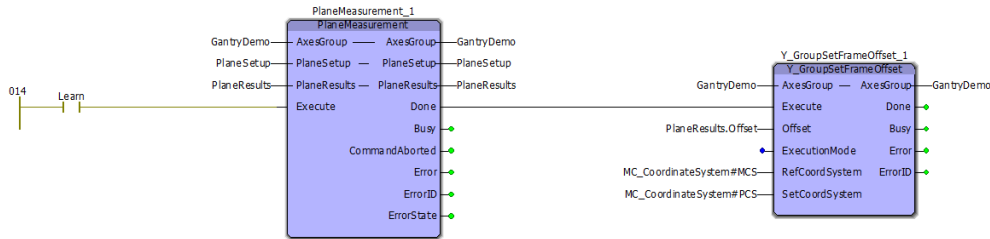
Example

```

214 PlaneSetup.MeasureMethod:=GTS_MeasurementType#Torque;          (* Use torque limited touch mode *)
215 PlaneSetup.MeasuredPlane:=2;                                  (* Measure the plane which the Z axis contacts *)
216 PlaneSetup.Tolerance:=LREAL#0.5;                             (* Measurement position must be within this value of the PlaneSetup.Point[n][3] for the plane axis *)
217
218 PlaneSetup.Acceleration:=MyMachine.Prms.Acceleration;
219
220 PlaneSetup.Point[0][1]:=LREAL#0.0;                            (* Coordinate for first measurement *)
221 PlaneSetup.Point[0][2]:=LREAL#0.0;
222 PlaneSetup.Point[0][3]:=LREAL#0.0;
223
224 PlaneSetup.Point[1][1]:=LREAL#0.0;                            (* Coordinate for second measurement *)
225 PlaneSetup.Point[1][2]:=LREAL#60.0;
226 PlaneSetup.Point[1][3]:=LREAL#0.0;
227
228 PlaneSetup.Point[2][1]:=LREAL#30.0;                           (* Coordinate for third measurement *)
229 PlaneSetup.Point[2][2]:=LREAL#60.0;
230 PlaneSetup.Point[2][3]:=LREAL#0.0;
231
232 PlaneSetup.TransferPosition:=LREAL#20.0;                      (* Position above plane where the tool moves from one measurement location to the next location *)
233 PlaneSetup.TransferVelocity:=LREAL#200.0;
234
235 PlaneSetup.ApproachPosition:=LREAL#4.0;                       (* Position at which the torque limit is applied in preparation for measuring the plane *)
236 PlaneSetup.ApproachVelocity:=LREAL#25.0;
237
238 PlaneSetup.MeasurementVelocity:=LREAL#1.0;                    (* This is the torque limit in percentage of rated torque for the servo motor *)
239 PlaneSetup.MeasurementTorqueLimit:=DINT#10;
240
241 PlaneSetup.ExternalPlane[0].AxisNum:=UINT#8;                 (* Only necessary if the group uses a virtual axis for the measured plane axis *)

```

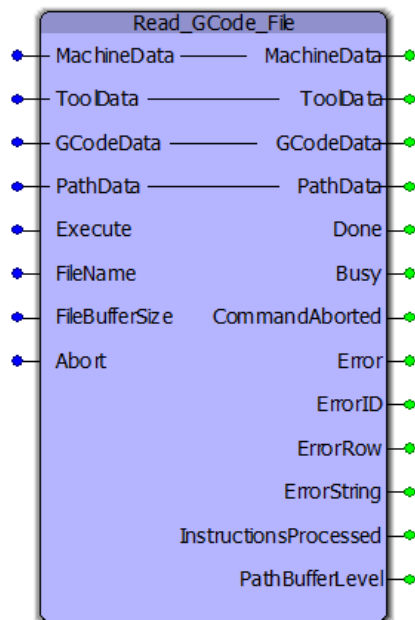




After setting the offset via Y_GroupSetFrameOffset, perform all motion on the plane using CoordinateSystem:=MC_CoordinateSystem#PCS.



Read_GCode_File



This function block reads a file containing G-Codes from the MPiec controller's flash or ramdisk file system and writes to the [PathData](#) for use with the [MC_MovePath](#) function block. Refer to the [list of supported G & M Codes](#).

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	MachineData	MachineStruct	Contains parameters such as maximum velocities and accelerations, and support for 3D printers and tangent axes if required. Also included are Origin (Home) and Part Coordinate System offsets.
V	ToolData	ToolDataStruct	Structure holding ToolData structures. If this data is not applicable to the application, connect a dummy variable to satisfy the compiler.
V	GCodeData	GCodeStruct	Working data set used by the G-Code Processor. It is made available as a VAR_IN_OUT mainly for debugging purposes. For applications using variables for register values, this GCodeData.Variables sub structure must be connected to the VAR_IN_OUT "VarData" of MC_MovePath.

V	PathData	MC_PATH_DATA_REF	Data structure containing the details parsed from the G-Code source and used by the MC_MovePath function block.
VAR_INPUT			Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
V	FileName	STRING	Name of file to be read or written. Example: STRING#'/-flash/user/data/myFile.csv' FileName can include any extension. Max characters for the total FileName is 24. The directory path is not included in this maximum.
V	FileBufferSize	UDINT	The number of bytes to read from the file 'per cycle' or in one CYCLIC task in which the function block is executing. The maximum is 16384. See notes below for more details.
VAR_OUTPUT			
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	ErrorRow	UINT	The row in the file which caused the error. If the file contains Nxxx line number codes, then the line number from the N code is reported. If no N codes are included, then the RowCount is from the very first line of the file starting at 1. If N codes are given on some lines but not all lines, ErrorRow may be inaccurate - it will report the last line which included an N code which may not be the one that caused an error.
V	ErrorString	STRING	The G-Code instruction which caused the error.
V	InstructionsProcessed	UDINT	The number of individual G-Code instructions processed since this function block was executed. Nxxx line number codes excluded from the count. For example, the command G1 X10 Y30 Z4 counts as four instructions, but only one Segment in PathData.Segment[].
V	PathBufferLevel	REAL	Percentage of the PathData.Segment[] which contains data waiting to be processed by MC_MovePath . The default PathData.Segment[] is declared with size = 250. If 200 instructions have been processed by this function block, but MC_MovePath and the physical machine have only processed 25, the PathBufferLevel would be $(200-25)/250 * 100 = 70\%$. This function will automatically wait until MC_MovePath has processed segments and continue filling the PathData structure with new data until the entire file has been read.

Notes

Processing is divided into two parts; Read_GCode_File and [MC_MovePath](#). These blocks may be put into separate tasks. Typically MC_MovePath is placed in a faster task, especially if InputFlags and OutputFlags are used. Read_GCode_File should be executed in a slower task, or even the DEFAULT task, which is recommended. It will parse the number of bytes as specified by FileBufferSize per cycle. A cycle is six scans. For example, if FileBufferSize is set to 10000, and Read_GCode_File is executed in a Cyclic task running every 50 mSec, a cycle is 300 mSec (50 * 6 = 300). Given these settings, the controller will process a maximum of 33,000 bytes/second, but may be less, based space available in the motion buffer. The datatype definitions of [MC_PATH_DATA_REF.Segments](#) and the number of characters in the ByteBufferStruct specified by FileBufferSize is configurable if necessary. Consult Yaskawa for details.

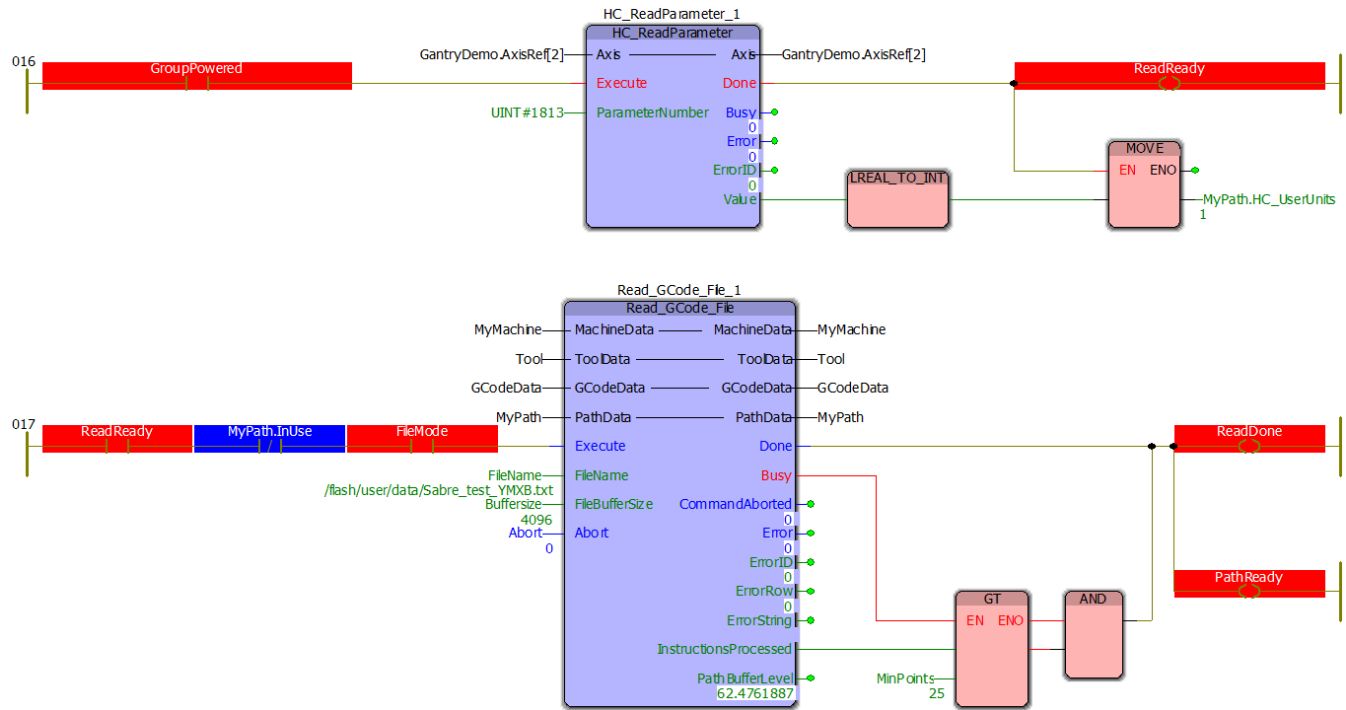
Error Description

See the [Function Block ErrorID](#) list.

Example

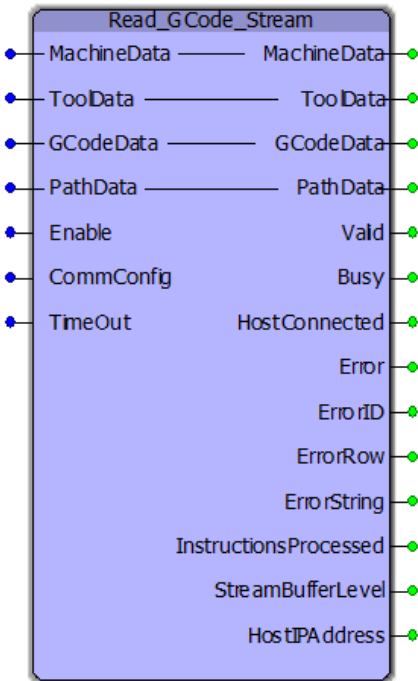
This example shows the HC_ReadParameter function block from the File_RW_Toolbox. It reads parameter 1813 to obtain the code for the user units selected for one of the Cartesian axes of the mechanism, which is copied into MyPath.HC_UserUnits. This allows the Read_GCode_File function block to compare the machine configuration to the G20 / G21 setting within data files and convert the position data as necessary.

This example also demonstrates that the InstructionsProcessed Output can be used to start motion. When the block is Busy and at least 6 datapoints have been processed, a BOOL variable is set which can be used to initiate motion using MC_MovePath.





Read_GCode_Stream



This function block reads and parses a G-Code stream from the configured communication device and writes to the [PathData](#), which can be used by [MC_MovePath](#).

Library

Group Toolbox

Parameters

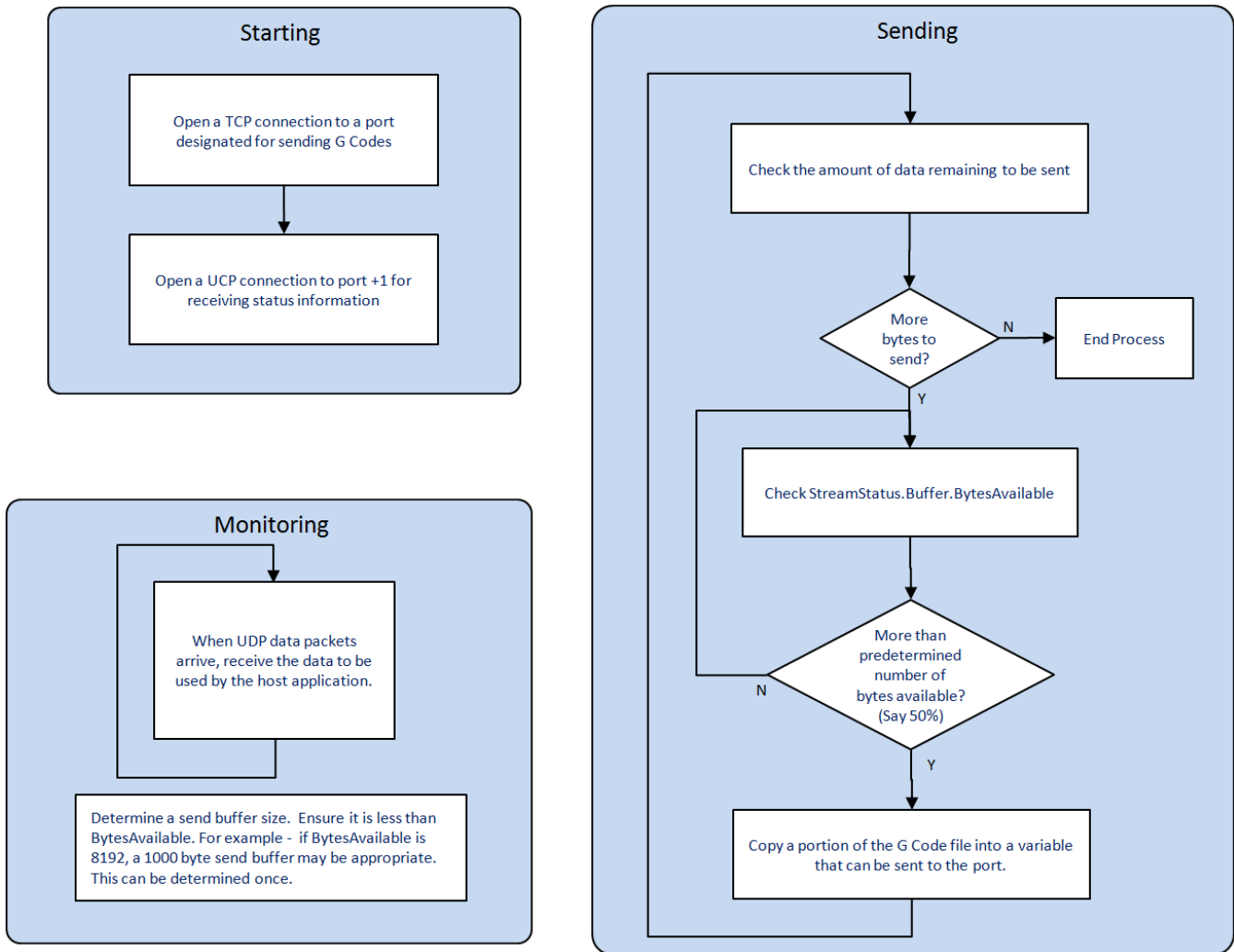
*	Parameter	Data Type	Description
VAR_IN_OUT			
V	MachineData	MachineStruct	Contains parameters such as maximum velocities and accelerations, and support for 3D printers and tangent axes if required. Also included are Origin (Home) and Part Coordinate System offsets.
V	ToolData	ToolStruct	Contains radius and length data for tools that may be selected.
V	GCodeData	GCodeStruct	Working data set used by the G-Code Processor. It is made available as a VAR_IN_OUT mainly for debugging purposes. For applications using variables for register values, this GCodeData.Variables sub structure must be connected to the VAR_IN_OUT "VarData" of MC_MovePath.

V	PathData	MC_PATH_DATA_REF	Data structure containing the details parsed from the G-Code source and used by the MC_MovePath function block.
VAR_INPUT			Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	CommConfig	CommStruct	Structure which configures this function block to communicate with the GCodeComm.DLL on a PC.
V	TimeOut	TIME	Set this value if the controller should close the connection and stop waiting for commands. Normally this value should be set to zero.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	HostConnected	BOOL	Confirms that a host has successfully initiated a connection.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	ErrorString	STRING	For some errors, this string is the command which caused the error.
V	InstructionsProcessed	UDINT	The number of individual G-Code instructions processed since this function block was enabled. Nxxx line number codes are excluded from the count. For example, the command G1 X10 Y30 Z4 counts as four instructions, but only one Segment in PathData.Segment[.].
V	StreamBufferLevel	REAL	Percentage of the PathData.Segment[.]. which contains data waiting to be processed by MC_MovePath. The default PathData.Segment[.] is declared with size = 500. If 200 instructions have been processed by this function block, but MC_MovePath and the physical machine have only processed 25, the StreamBufferLevel would be $25/500 * 100 = 5\%$.
V	HostIPAddress	STRING	The IP address of the device which initiated a connection request to the MPiec controller for G-Code streaming.

Notes

- The host application must read the [StreamStatus](#) to determine the number of bytes available in the bytearray. The host application and the Read_GCode_Stream function block will work together to transmit / receive the byte stream of G-Code information.
- Yaskawa recommends executing this function block in the DEFAULT task with the watchdog disabled.

Basic Flowchart of Host PC application



Error Description

See the [Function Block ErrorID](#) list.

Example

The following structured text shows the initialization of CommCfg.

```

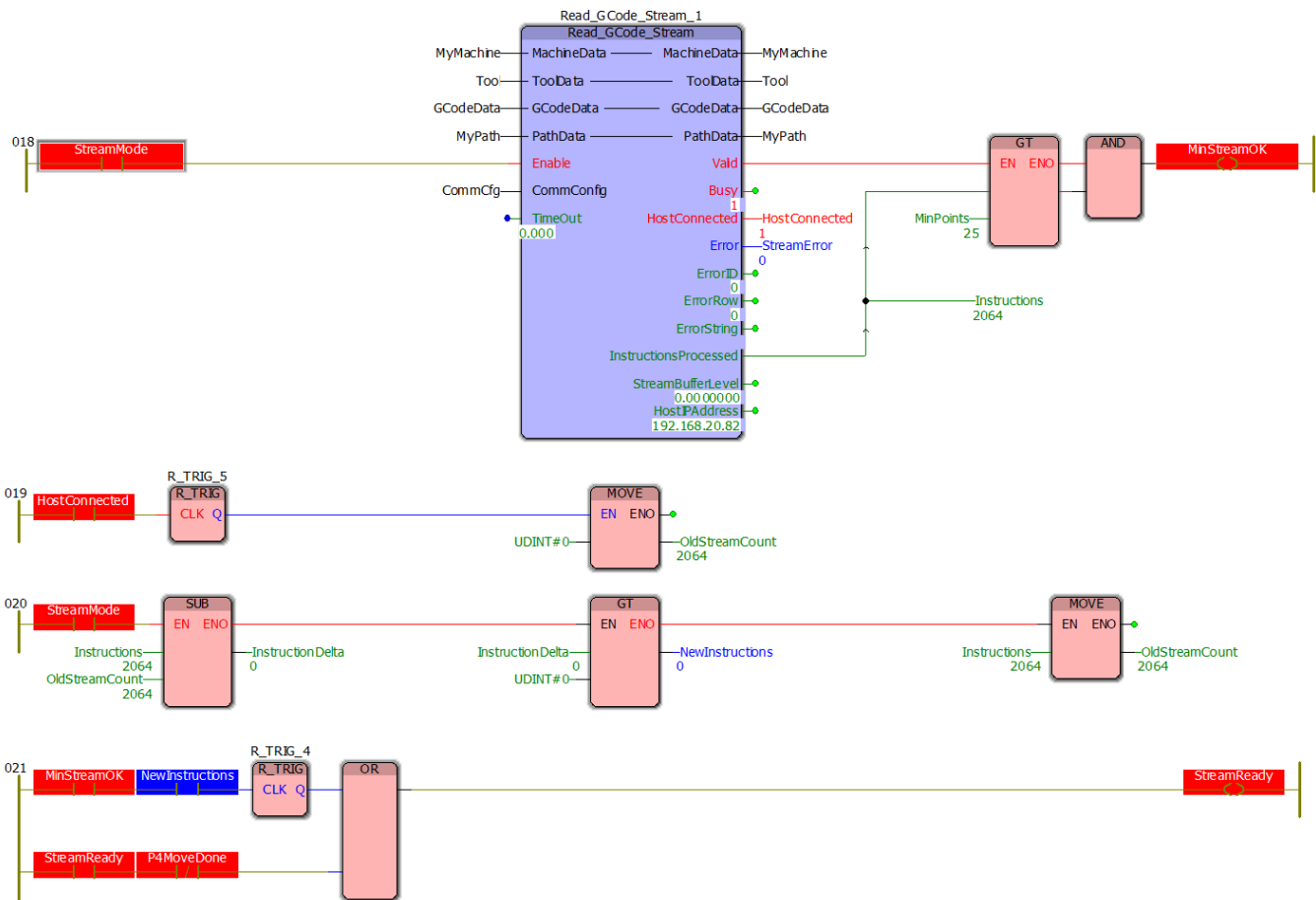
64      192.168.207.151 |CommCfg.Ethernet.LocalIPAddress:=STRING#'192.168.207.151';
65      1206 |CommCfg.Ethernet.LocalPort:=UINT#1206;
66      0.000 |CommCfg.InactivityTimeout:=T#0S;
67      2 |CommCfg.CommType:=Comm_Type#Ethernet;
68      0 |CommCfg.CommandType:=Command_Type#Variable;
69      0 |CommCfg.BufferSize:=UDINT#0; (* Y_ReadDevice reporting 8719 if non zero *)
  
```

- The LocalIPAddress is that of the MPiec controller. This is necessary because the controller may have more than one IP address and more than one network. It allows the function to listen for G-Code data on the correct network. This information can be automatically obtained by referencing the ControllerInfo structure and converting the IP address as a byte array into a string using BYTE_TO_STRING. See the DataTypes section in the PLCopenPlus help manual for more information.
- Local Port can be nearly any convenient port number that you choose. The host application must open a connection to the port specified here.

- Timeout is not supported and must be set to T#0s.
- CommType is typically Ethernet, but if the MPiec controller has a 218IF-Y1 option card, serial data is possible.
- CommandType must be set to Command_Type#Variable. This is because G-Code commands have a varying number of characters separated by delimiters as opposed to fixed sized packets.
- BufferSize is not used in Variable command length mode.

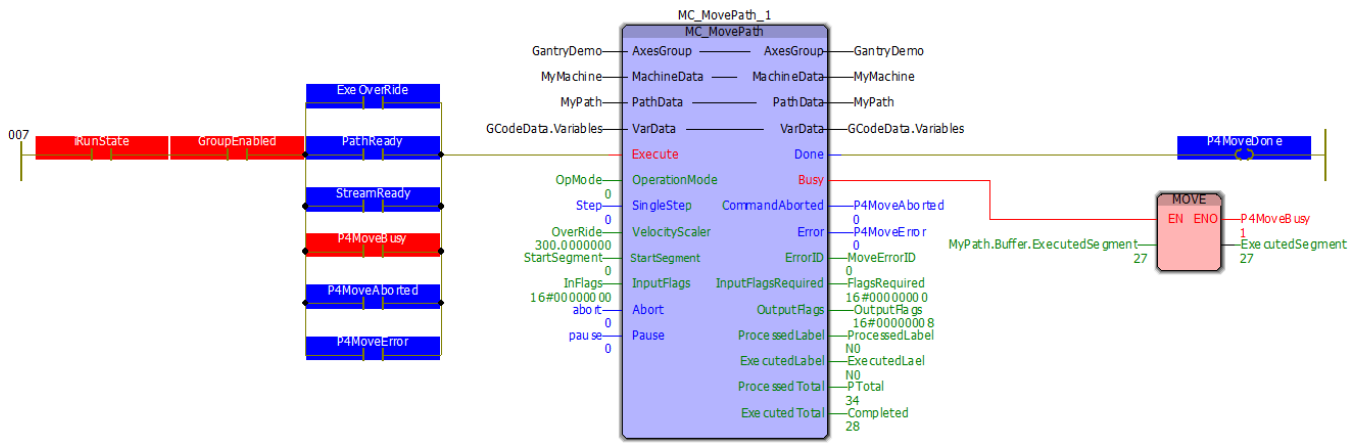
User Units

You can optionally use the [HC_ReadParameter](#) from the File Read Write Toolbox, which will read the Hardware Configuration setting for the axis selected. (Choose one of the axes that provides Cartesian movement.) Alternatively, if you know the user units of the group's MCS, load MyPath.HCUserUnits directly. 0=Inches, 1=millimeters, 2=microns. This setting allows the G-Code processing functions to convert values to the correct units for the machine if the G-Code data specifies different user units (Refer to [G20 / G21](#)).



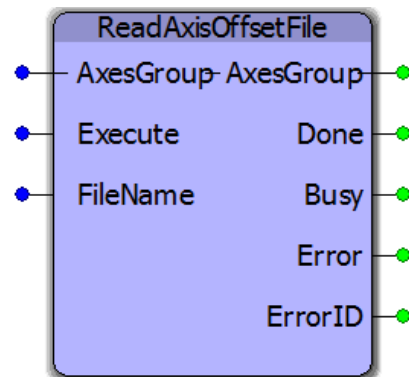
In the example above, a minimum number of points must be loaded into MC_PATH_DATA_REF.Segment[] before the StreamReady flag is set. This flag is referenced in another task which executes the [MC_MovePath](#) function block. This allows paths with many small segments to be buffered and sent to the motion engine more quickly to avoid data starvation when motions starts.

The example below shows the MC_MovePath function including StreamReady logic.





ReadAxisOffsetFile



This function block reads absolute encoder offset information from a file written by [WriteAxisOffsetFile](#). It restores the absolute encoder offsets retained in the MPiec controllers battery backed memory for all axes in an AxesGroup. Restoring encoder offsets is necessary in the event of an MPiec controller replacement or MPiec SRAM battery failure.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.
VAR_INPUT			Default
B	Execute	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors. FALSE
V	FileName	STRING	The file name as listed in the /Flash/Local directory on the controller. The extension is not required and will be automatically appended. STRING#''
VAR_OUTPUT			
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.

B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

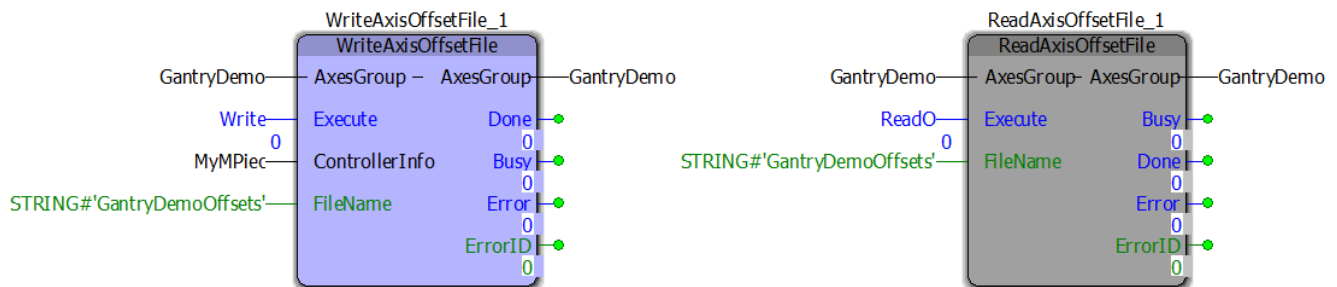
Notes:

- Firmware version 3.3.0 is required to use this function. It relies on MC_ReadParameter 1838, which was added for version 3.3.0.
- The Flash/Local directory was also added for firmware 3.3.0. This folder is not deleted when a project archive is deleted or the controller is restored to factory defaults.
- Once the file is created using [WriteAxisOffsetFile](#), it is highly recommended to save a backup copy of the file to another location other than the MPiec controller. This file can be replaced manually via the web UI in the event that a new MPiec controller is connected to the existing mechanical equipment.
- The offsets contained in this file are only valid in the following situations:
 - MPiec controller replacement .
 - MPiec SRAM battery failure.
- The offsets contained in this file become invalid in the following situations:
 - Absolute encoder battery failure or disconnection. (ServoPack has A.810 alarm)
 - Motor replacement.
 - Any mechanical alteration to the drive train, including belts, gearboxes, couplings, etc.

Error Description

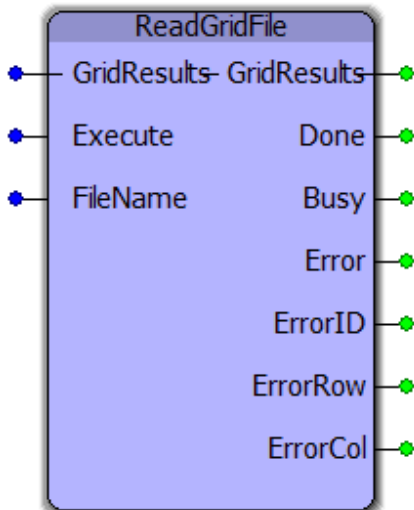
See the [Function Block ErrorID](#) list.

Example





ReadGridFile



This function block reads a GMD file from the /Flash/Local folder previously written by [WriteGridFile](#).

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	GridResults	GridResultsStruct	Contains the measurement information populated by the GridMeasurement function block.
VAR_INPUT			
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
V	FileName	STRING	Enter only the file name without path or extension. The file will be read from the /Flash/Local folder. This folder is never deleted even if the MPiec is restored to factory defaults or an archive is added or deleted. The extension .GMD will automatically be appended.
VAR_OUTPUT			
			Default
			FALSE
			STRING#"

B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	ErrorRow	INT	If applicable, indicates the row in the file which generated the error.
V	ErrorCol	INT	If applicable, indicates the column in the file which generated the error.

Notes

- Yaskawa recommends executing this function block in a slow (100mSec) or Default task.
- See related function blocks: [GridMeasurement](#), [WriteGridFile](#), GridLookup.

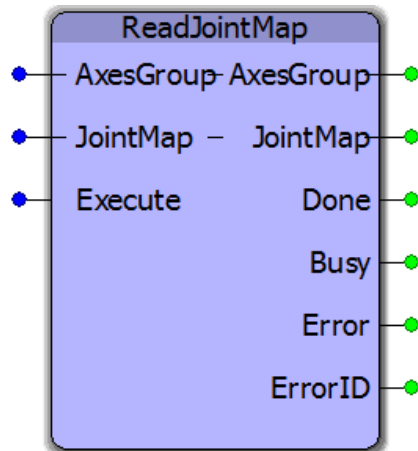
Error Description

See the [Function Block ErrorID](#) list.

Example



ReadJointMap



This function block will populate the JointMap with data linking Axis_Ref, Joint, and Joint Label. It can be useful when creating applications which must programmatically act upon specifically named joints or axes rather than predetermined AXIS_REF numbers for a specific machine.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.	
V	JointMap	JointMap	Structure containing group information populated by this function block.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	

B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

Error Description

See the [Function Block ErrorID](#) list.

Example

With a group configured as shown, the ReadJointMap function block will populate the JointMap as viewed in the Watch Window.

The configuration shown is a 3D gantry with a theta axis operated externally to the group using Y_SyncTangentAxisToGroup. The application uses a special configuration which inserts the theta axis into the AxesGroup structure, therefore axis 5 in the Jointmap shows an ExternalTanget as Axis_Ref 7.

The screenshot displays the MotionWorks IEC 3 Pro - Hardware Configuration interface. The left sidebar shows a tree view for 'IMTS 2016' with a resource 'MP3200iec'. Under 'Mechatrolink-III', there are axes Y-3, Y_Prime-4, X-5, Z-6, and Theta-7. A 'Groups' section contains 'GantryDemo', 'TCP/IP Settings', 'EtherNet/IP', and 'Modbus/TCP'. The main window shows the 'GantryDemo' configuration for resource 'MP3200iec' in 'Offline' mode. The 'Configuration' tab is active, showing 'GantryDemo' settings. A 3D model of the gantry is shown with X, Y, and Z axes. A table lists the axes:

Label	Axis Name	Index
X	X	1
Y	Y	2
Y	Y_Prime	2
Z	Z	3

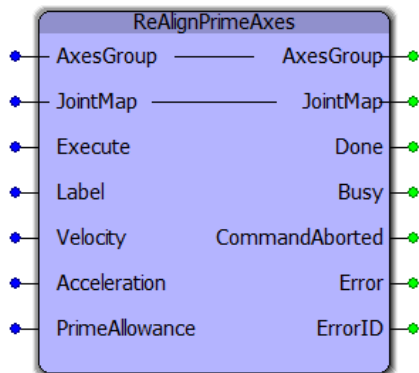
Below the table, there are controls for 'Error Stop Mode' (Max Decel), 'Decel Stop Time (sec)' (0.5), 'Max Filter Samples' (50), and 'Motion Queue Size' (128). Buttons for 'Add Axis', 'Add Secondary Axis', 'Remove Axis', and 'Remove Secondary Axis' are also present.

Watch Window

Variable	Value	Type
[-] JointMap		JointMap
[-] [1]		JointMapDetail
Label	X	STRING
Joint	1	INT
AxisRef	5	UINT
[-] [2]		JointMapDetail
Label	Y	STRING
Joint	2	INT
AxisRef	3	UINT
[-] [3]		JointMapDetail
Label	Y_Prime	STRING
Joint	2	INT
AxisRef	4	UINT
[-] [4]		JointMapDetail
Label	Z	STRING
Joint	3	INT
AxisRef	6	UINT
[-] [5]		JointMapDetail
Label	ExternalTangent	STRING
Joint	0	INT
AxisRef	7	UINT
[-] [6]		JointMapDetail
Label		STRING
Joint	0	INT
AxisRef	0	UINT



ReAlignPrimeAxes



Designed for Groups configured with more than one servo operating a single joint, this function block will move the secondary motor(s) to the commanded position of the main servo if they are already within the PrimeAllowance provided. Joints with multiple servos must have the same commanded position prior to executing MC_GroupEnable, otherwise the ErrorID 8966 will occur.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional sub-structures pertaining to the group.	
V	JointMap	JointMap	Structure of data that holds information for Joint Index, Label, AxisName and AXIS_REF.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	Label	STRING	Specify the Joint to be aligned. Use the 'Label' name as shown in the Hardware Configuration for the group.	STRING#''
B	Velocity	LREAL	Absolute value of the velocity in user units/second.	LREAL#0.0

B	Acceleration	LREAL	Value of the acceleration in user unit-s/second ² (acceleration is applicable with same sign of torque and velocity)	LREAL#0.0
V	PrimeAllowance	LREAL	Specify the maximum amount of difference between the main and prime servo commanded positions to permit safe alignment. See the Getting Started with Secondary Axes section.	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
E	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

- This block only supports Mechatrolink gantry groups, not remote hosted robots via MotomanSync.
- See [GroupReAlignPrimeAxes](#), which will iterate through all joints and calls this block as many times as necessary to align all joints.
- The File R/W Toolbox and the PROCONOS firmware library are required when using this function block.

Error Description

See the [Function Block ErrorID](#) list.

Example

This example shows how the function is used within [GroupToHome](#). A CASE statement sequences through the joints and executes ReAlignPrimeAxes for all the joints configured with secondary axes.

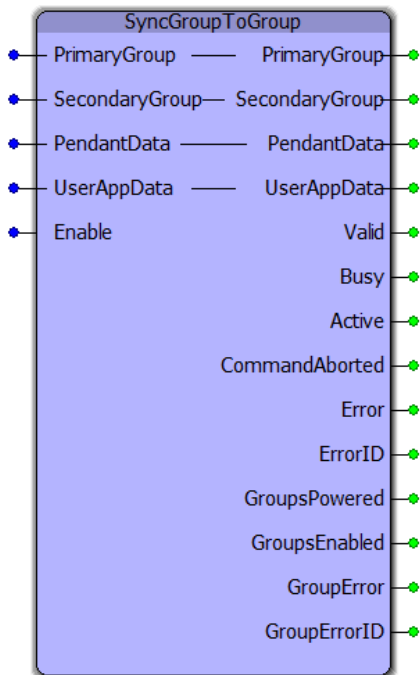
```

107 3: (* Align the Prime axis with the Main axis *)
108   IF ReAlignPrimeAxes_1.Busy THEN
109     Output[State].Busy:=TRUE;
110   ELSIF ReAlignPrimeAxes_1.Done THEN
111     _d:=_d + INT#1;
112     State:=INT#2;
113   ELSIF ReAlignPrimeAxes_1.CommandAborted THEN
114     Output[State].CommandAborted:=TRUE;
115   ELSIF ReAlignPrimeAxes_1.Error THEN
116     Output[State].Error:=TRUE;
117     Output[State].ErrorID:=ReAlignPrimeAxes_1.ErrorID;
118   END_IF;
119   ReAlignPrimeAxes_1.AxesGroup:=AxesGroup;
120   ReAlignPrimeAxes_1.JointMap:=JointMap;
121   ReAlignPrimeAxes_1.Execute:=iActive AND (State=INT#3);
122   ReAlignPrimeAxes_1.Label:=GroupHomeData.Sequence[_s].DOF[_d];
123   ReAlignPrimeAxes_1.Velocity:=GroupHomeData.Sequence[_s].Velocity;
124   ReAlignPrimeAxes_1.Acceleration:=GroupHomeData.Sequence[_s].Acceleration;
125   ReAlignPrimeAxes_1.PrimeAllowance:=GroupHomeData.PrimeAllowance;
126   ReAlignPrimeAxes_1();
127   AxesGroup:=ReAlignPrimeAxes_1.AxesGroup;
128   JointMap:=ReAlignPrimeAxes_1.JointMap;

```




SyncGroupToGroup



For applications that require custom kinematics or other tailoring of the groups commanded position. Typically a virtual group is configured to accept motion commands in world space (MCS) and a second group configured with the mechanisms actual axes is operated via [Y_GroupDirectControl](#) contained within this function block.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	PrimaryGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.
V	SecondaryGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional substructures pertaining to the group.
V	PendantData	PendantDataStruct	Contains the data which is shared with the host via GroupComm.DLL.
V	UserAppData	UserApplicationData	Structure which contains configuration, teach point and tool data.
VAR_INPUT			Default

B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Active	BOOL	For buffered modes, this output is set high at the moment the block takes control of the axis. For non buffered modes, the outputs Busy and Active have the same value.	
E	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	GroupsPowered	BOOL	Indicator that both groups are powered. (Y_GroupPower has been applied to the PrimaryGroup, and SyncGroupToGroup has applied the same action to the SecondaryGroup successfully.	
V	GroupsEnabled	BOOL	Indicator that both groups are enabled. (MC_GroupEnable has been applied to the PrimaryGroup, and SyncGroupToGroup has applied the same action to the SecondaryGroup successfully.	
V	GroupError	BOOL	Indicator that the SecondaryGroup has an error.	
V	GroupErrorID	UINT	The SecondaryGroups ErrorID.	

Notes

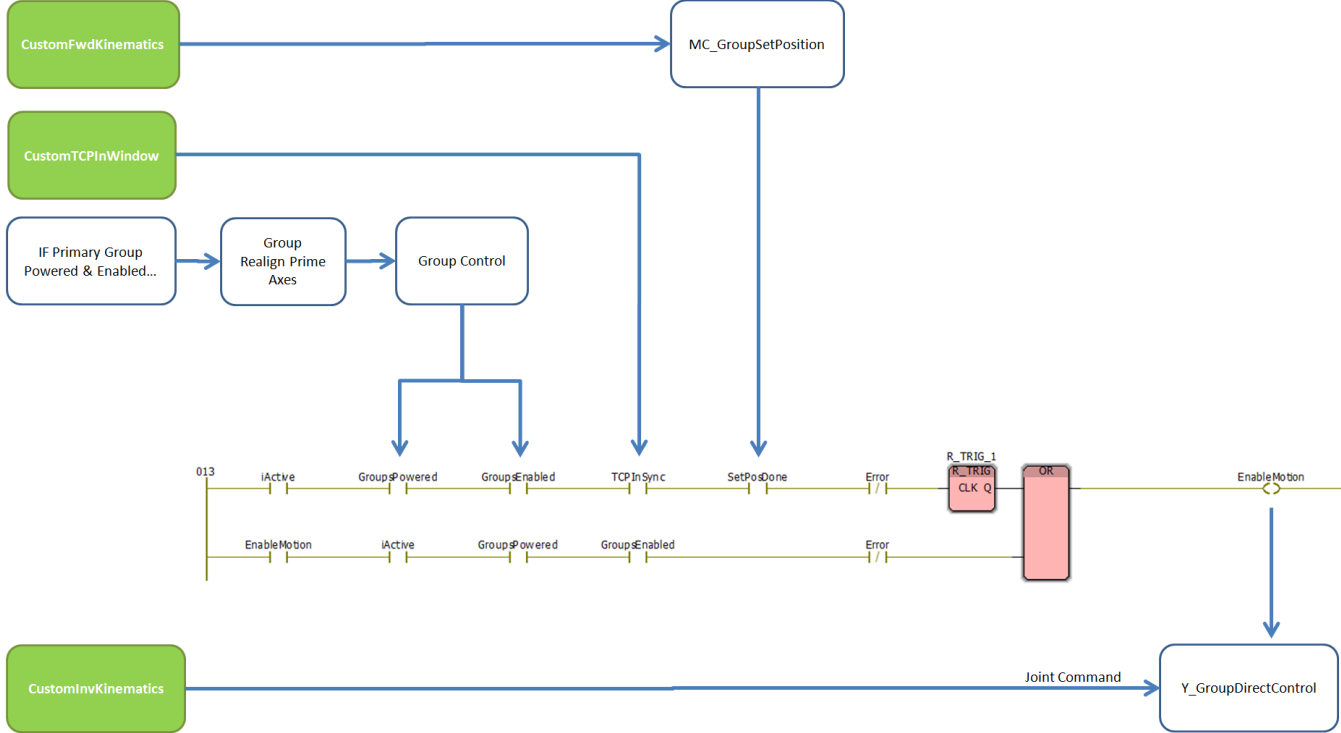
- A main benefit of this function block is the interlock code included to manage the commanded position of the SecondaryGroup to avoid causing the 3301 0018 alarm or causing the axes to jump aggressively when enabling [Y_GroupDirectControl](#). Warning - this may still occur if the kinematics provided are not correct.
- This function block was designed to make a dual group solution look like a single group when controlled via pendant or HMI software such as Compass. Status and Alarms on the Secondary group are propagated to the PendantData struct which normally interfaces with a single group.
- Ideally, execute SyncGroupToGroup in a task with the interval set to the same as the Mechatrolink network interval. This block sends the results of the kinematic equations directly to command the positions of the Secondary Group. Performance will be degraded if they are not updated as fast as possible.
- Customization of the following function blocks is required when using SyncGroupToGroup: (See Block Diagram and Example below.)
 - CustomInvKinematics
 - CustomFwdKinematics
 - CustomTCPInWindow
- Customization requires copying and pasting each function from the Group Toolbox to the main project. Without this step, the default instances of these functions in the Group Toolbox will be executed using the default calculations. *The compiler gives precedence to the instance in the main project if a function block with the same name as found in a user library.*

Error Description

See the [Function Block ErrorID](#) list.

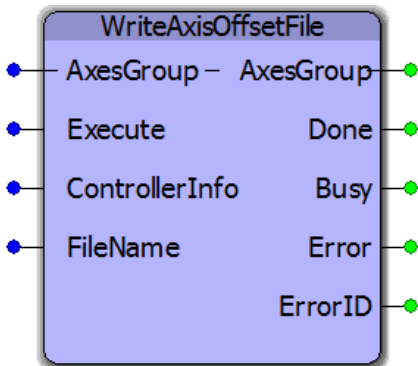
Block Diagram

The green blocks represent the stub function blocks which must be edited in the main project by the user. Copy and paste them from the Group Toolbox. This can be accomplished by opening a second copy of MotionWorks IEC and opening the Group Toolbox. The stub function blocks must not be added to a program or function however; they are called from within Syn-cGroupToGroup.





WriteAxisOffsetFile



This function block writes absolute encoder offset information to a file which can later be read by [ReadAxisOffsetFile](#). It records the absolute encoder offsets retained in the MPiec controllers battery backed memory for all axes in an AxesGroup. These offsets can be restored in the event of an MPiec controller replacement or SRAM battery failure.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	AxesGroup	AXES_GROUP_REF	A logical reference to a group of axes, which contains several additional sub-structures pertaining to the group.
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	ControllerInfo	CONTROLLER_INFO	Place this variable of type CONTROLLER_INFO at address %MD3.66560. This is required to include other data in the file such as the controller firmware version.
V	FileName	STRING	The file name without an extension, which will be automatically appended. Do not include the directory folders, they are also automatically added. See the example below.
VAR_OUTPUT			
	Done		
	Busy		
	Error		
	ErrorID		

B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

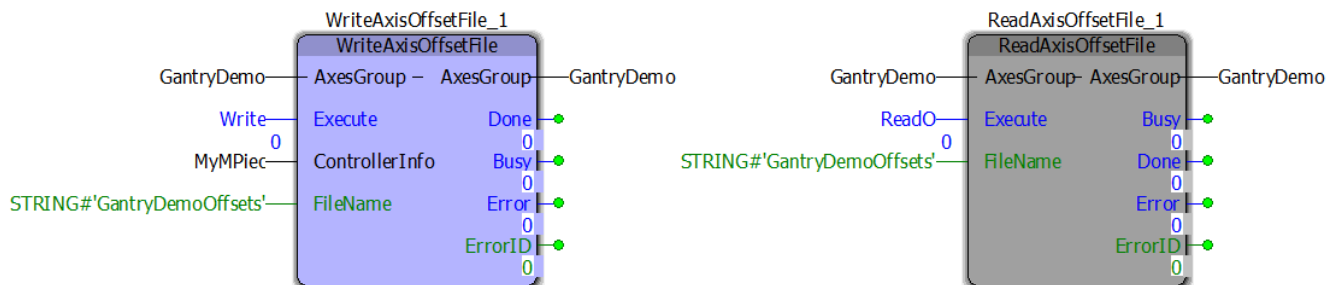
Notes:

- Firmware version 3.3.0 is required to use this function. It relies on MC_ReadParameter 1838, which was added for version 3.3.0.
- The Flash/Local directory was also added for firmware 3.3.0. It is not deleted when a project archive is deleted or the controller is restored to factory defaults.
- It is highly recommended to save a backup copy of the file to another location other than the MPiec controller. This file can be downloaded / uploaded from the MPiec web UI in the event that a new MPiec controller is connected to the existing mechanical equipment.
- The offsets contained in this file are only valid in the following situations:
 - MPiec controller replacement
 - MPiec SRAM battery failure.
- The offsets contained in this file become invalid in the following situations:
 - Absolute encoder battery failure or disconnection. (ServoPack has A.810 alarm)
 - Motor replacement
 - Any mechanical alteration to the drive train, including belts, gearboxes, couplings, etc.

Error Description

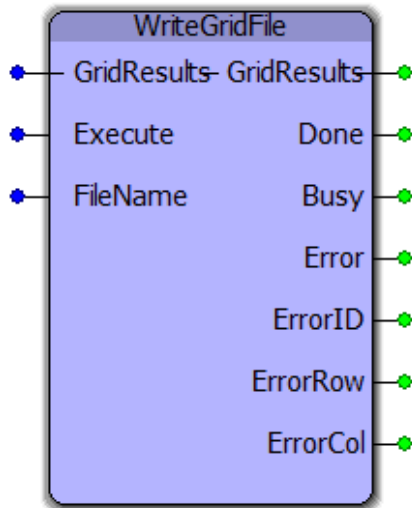
See the [Function Block ErrorID](#) list.

Example





WriteGridFile



This function block writes the GridResults structure that was populated by the [GridMeasurement](#) function block. The FileName will be appended with a GMD extension in the /Flash/Local folder on the MPiec controller.

Library

Group Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	GridResults	GridResultsStruct	Contains the measurement information populated by the GridMeasurement function block.	
VAR_INPUT			Default	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	FileName	STRING	Enter only the file name without path or extension. The file will be written to the /Flash/Local folder. This folder is never deleted even if the MPiec is restored to factory defaults or an archive is added or deleted. The extension .GMD will automatically be appended.	STRING#"
VAR_OUTPUT				

B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	ErrorRow	INT	If applicable, indicates the row in the file which generated the error.
V	ErrorCol	INT	If applicable, indicates the column in the file which generated the error.

Notes

- Yaskawa recommends executing this function block in a slow (100mSec) or Default task.
- See related function blocks: [GridMeasurement](#), [ReadGridFile](#), GridLookup.

Error Description

See the [Function Block ErrorID](#) list.

Example



Math Revision History

Current Version:

2022-05-11 v374 released for MotionWorks IEC 3.7.4

No changes, identical to v371.

Previous Versions:

2021-04-09 v372 released with MotionWorks IEC 3.7.2

No changes, identical to v371.

2020-11-02 v371 released with MotionWorks IEC 3.7.1

Calc3DCenter - Improved FB to be PLCopen compliant, protect against divide by zero. DCR 4787.

2020-02-06 v370 released with MotionWorks IEC 3.7.0

Internal changes for code reduction, use single instance of ATAN2. SCR 12546.

ATAN2_F - Function equivalent of ATAN2 as function block, saves on compiled code instance memory. SCR 13009.

2018-5-14 v350 released with MotionWorks IEC 3.5.0

No changes.

2017-8-14 v340 released with MotionWorks IEC 3.4.0

CalcFrameOffset - New FB added.

CrossProduct - New FB added.

FrameTypeTransformation - New FB added.

2017-01-15 v331 released with Toolbox Installer - Jan 2017 Collection

RECT_TO_POLAR - New FB added. Support for Tool Compensation in Group Toolbox.

2016-10-31 v330 released with MotionWorks IEC 3.3.0

CrossProduct - New FB added.

Multiply4x4 - New FB Added. Support tools for MC_MoveCircular* in PLCopen Part 4.

InvertFrameMatrix - New FB Added. Support tools for MC_MoveCircular* in PLCopen Part 4.

DecompFrameMatrix - New FB Added. Support tools for MC_MoveCircular* in PLCopen Part 4.

ConstructFrameMatrix - New FB Added. Support tools for MC_MoveCircular* in PLCopen Part 4.

Removed all Boolean logic and simple math functions. Use IEC-61131 functions instead .

CalcRadius - New FB Added. Support tools for MC_MoveCircular* in PLCopen Part 4.
CalcCenter - New FB Added. Support tools for MC_MoveCircular* in PLCopen Part 4.
CalcEndAngle - New FB Added. Support tools for MC_MoveCircular* in PLCopen Part 4.
CalcStartAngle - New FB Added. Support tools for MC_MoveCircular* in PLCopen Part 4.
CalcEndAngle - New FB added. Support tools for MC_MoveCircular* in PLCopen Part 4.
POLAR_TO_RECT - New FB added. Support tools for MC_MoveCircular* in PLCopen Part 4.

2015-01-31 v300 created

Identical to v202, but recompiled specifically for MotionWorks IEC v3.x.

2012-10-22 v202 released

1) Added functionality to the ATAN2 function block. New ENUM type as VAR_INPUT was added to configure it to operate in 0 to 2 pi radians or 0 to 360 degree in addition to -pi to pi radians. The default behavior (-pi to pi) is the same as previous versions.

2) New MathDataTypes file added. This contains enum types for ATAN2 input options.

2012-01-23 v201 released

Made change in REM function to prevent a negative result. Added TRUNC_DINT to the code in REM

Refer SCR 1241 on LREAL_TO_DINT fixed in FW 2.0.0.255

2011-07-29 v200 released

Upgraded to version 2.0 Project for MotionWorks IEC. Built from Math Toolbox v004beta .



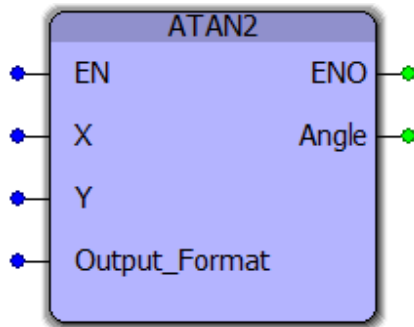
Enumerated Types for Math Toolbox

Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to zero-based integers (INT). Therefore, the first value equates to zero, the second to 1, etc. The format for enumerated types is as follows: ENUM:(0, 1, 2...) as displayed in the example below (MC_BufferMode#Aborting).

Enumerated Type	#INT Value	Enum Value	Description
TB_ATAN2_OutputType	0	NegPi_Pi	Output angle is defined from -Pi to Pi.
	1	Zero_TwoPi	Output angle is defined from 0 to 2Pi.
	2	ThreeSixty	Output angle is defined from 0 to 360.



ATAN2



The ATAN2 function is useful in many applications involving vectors, such as finding the direction from one point to another. This two argument function is a variation of the ATAN function. For any LREAL arguments x and y , $\text{atan2}(y, x)$ is the angle between the positive x -axis of a plane and the point given by the coordinates (x, y) on it.

Library

Math Toolbox

Parameters

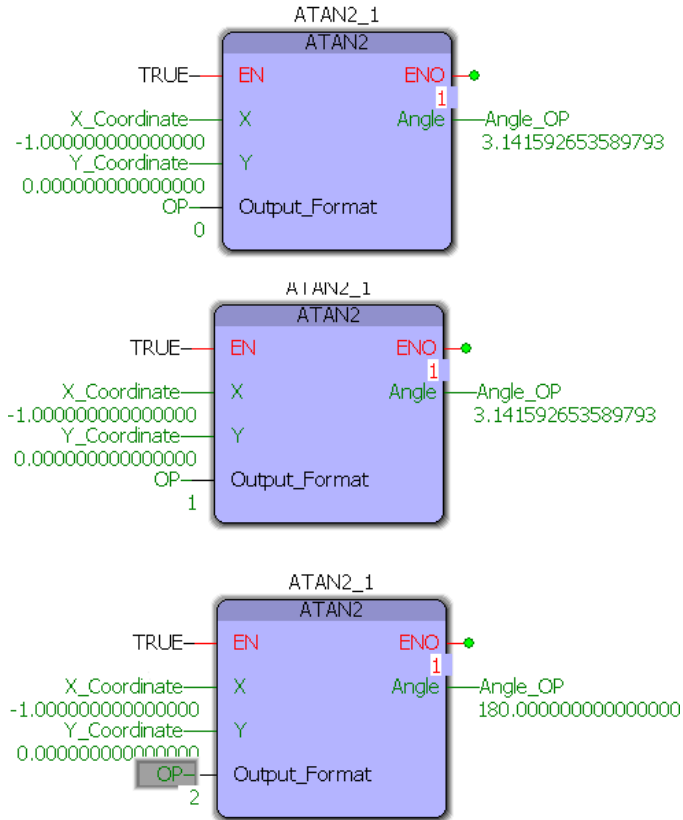
*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	EN	BOOL	This function will continue to calculate the ATAN2 result while EN is held high.	FALSE
V	X	LREAL	X coordinate	LREAL#0.0
V	Y	LREAL	Y coordinate	LREAL#0.0
V	Output_Format	INT	Format of the output value. 0: radians $(-\pi, \pi]$ 1: radians $[0, 2*\pi)$ 2: degrees $[0^\circ, 360^\circ)$	INT#0
VAR_OUTPUT				
B	ENO	BOOL	High if the function is executing normally.	
V	Angle	LREAL	The result of the ATAN2 calculation.	

Notes

This is a function, not a function block and only provides one output. If ENO is not high when EN is high, this function cannot calculate the Angle.

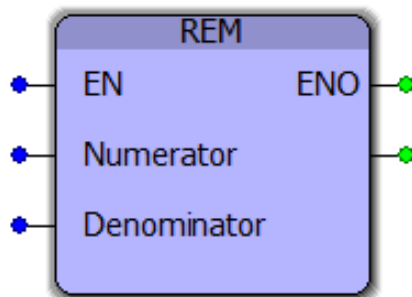
Example

ATAN2 used with various output formats:





REM



This function block returns the modulo division result of two LREAL inputs. It is useful for determining the position within a MachineCycle.

Library

Math Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	EN	BOOL	This function will continue to calculate the remainder while EN his held high.	FALSE
V	Numerator	LREAL	The numerator for division, such as the free running motor position, which may be outside a desired range of values, such as 0 to 360.0	LREAL#0.0
V	Denominator	LREAL	The denominator for division, which is the desired max value for the Numerator input, such as 360.0	LREAL#0.0
VAR_OUTPUT				
B	ENO	BOOL	High if the function is executing normally.	
V	REM	BOOL	This output contains the calculated remainder	

Error Description

This is a function, not a function block and only provides one output. If ENO is not high when EN is high, this function cannot calculate the remainder. Verify that the Denominator is not zero.

Example 1 - Structured Text

IF InternalMode=INT #1 THEN

(* These calculations are designed for a rotary knife, rotary placer, one way cam, etc. *)

Correction:=REM((-RegistrationData.BufferNonCyclic[TempUsePointer] - RegistrationData.SensorOffset), CamMasterCycle)
+ ((ControlData.EndSyncPosition - ControlData.StartSyncPosition) / LREAL#2.0);

Duration:=RegistrationData.SensorDistance - ((ControlData.EndSyncPosition - ControlData.StartSyncPosition) /
LREAL#2.0) - (ActualPositionNonCyclic - RegistrationData.BufferNonCyclic[TempUsePointer]);

ELSE

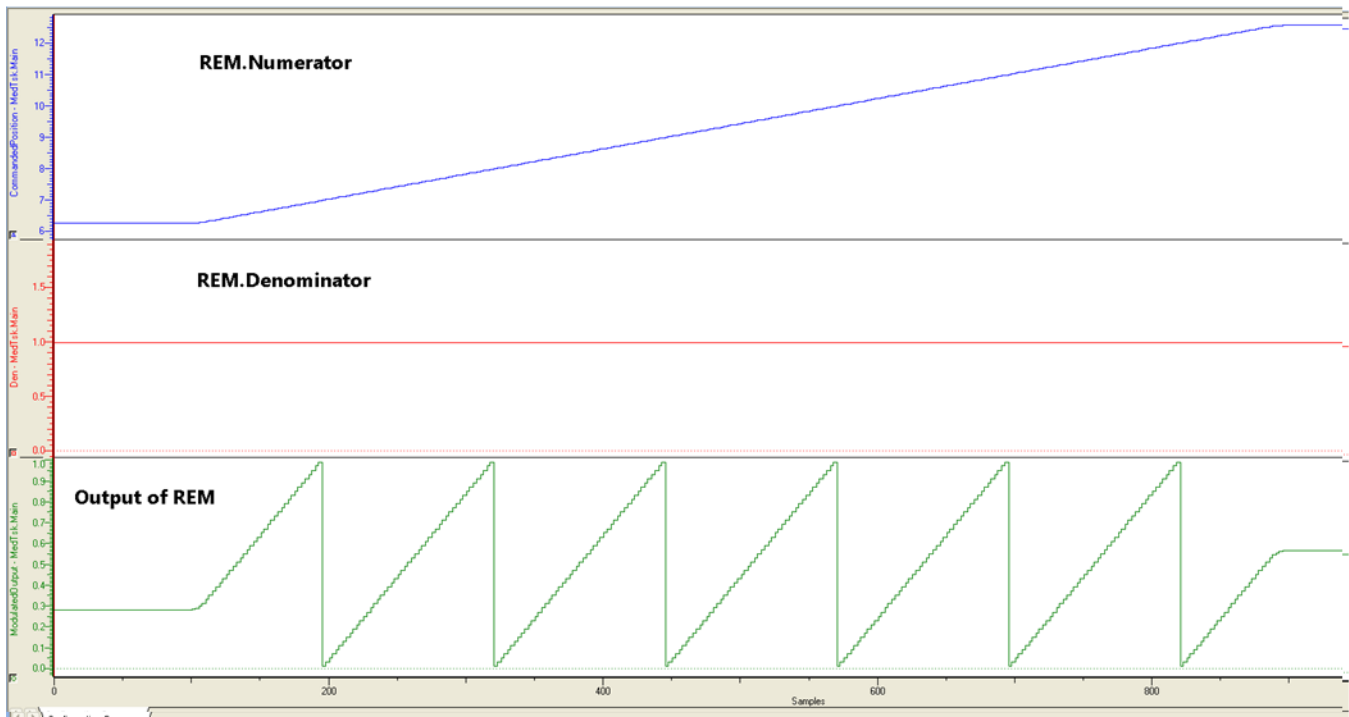
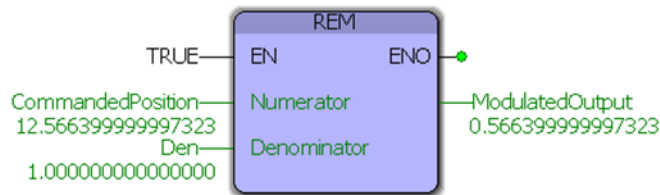
(* These calculations are designed for reciprocating cam profiles (Slave net change = zero each cycle, Out and Back *)

Correction:= - REM(REM(RegistrationData.BufferCyclic[TempUsePointer], CamMasterCycle) + (RegistrationData.SensorDistance - ControlData.StartSyncPosition - ((ControlData.EndSyncPosition - ControlData.StartSyncPosition) / LREAL#2.0))), CamMasterCycle);

Duration:=RegistrationData.SensorDistance - ControlData.StartSyncPosition - ((ControlData.EndSyncPosition - ControlData.StartSyncPosition) / LREAL#2.0);

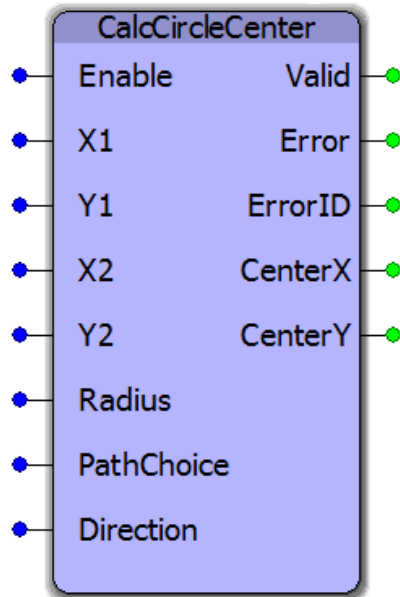
END_IF;

Example 2 - Function Block





CalcCircCenter



CalcCircCenter can be used whenever two points on the circle, the radius of the circle, the direction of the path (counterclockwise or clockwise), and the length of the path (shortest or longest) is given.

Library

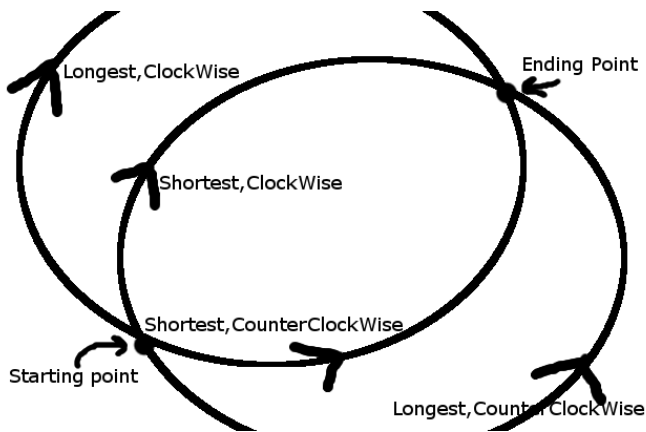
Math Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	X1	LREAL	X coordinate of starting point.	LREAL#0.0
V	Y1	LREAL	Y coordinate of starting point.	LREAL#0.0
V	X1	LREAL	X coordinate of ending point.	LREAL#0.0
V	Y1	LREAL	Y coordinate of ending point.	LREAL#0.0
V	Radius	LREAL	Radius of the circle.	LREAL#0.0

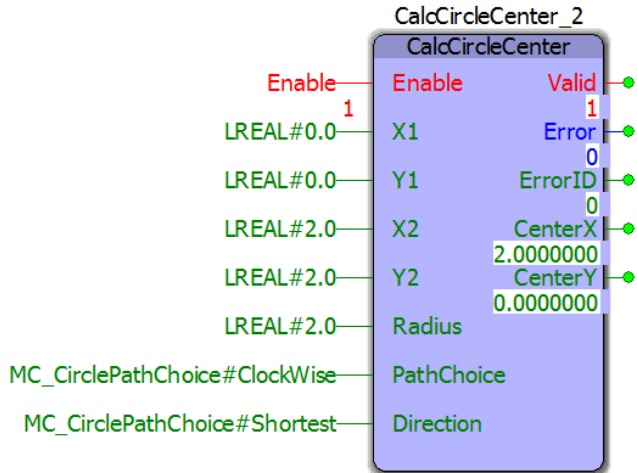
V	PathChoice	MC_CirclePathChoice	If the path length is larger than 180 degrees, use MC_CirclePathChoice#Longest. If the path length is 180 degrees or less, use MC_CirclePathChoice#Shortest.	MC_CirclePathChoice#ClockWise
V	Direction	MC_CirclePathChoice	If the circle path travels clockwise use MC_CirclePathChoice#Clockwise. Otherwise if the circle path travels counterclockwise use MC_CirclePathChoice#CounterClockWise.	MC_CirclePathChoice#ClockWise
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	CenterX	LREAL	X coordinate of the center of the circle	
V	CenterY	LREAL	Y coordinate of the center of the circle	

Notes



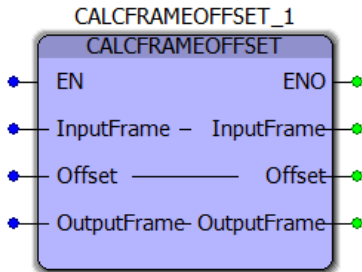
Whenever given two points and a radius, it is possible to create two circles that pass through both points with the same radius, and different center coordinates. In order to know which center coordinates we need we can either use a unique third point or the Direction (Clockwise vs CounterClockWise) and the PathChoice (Shortest or Longest). Currently support is only available for choosing the Direction and PathChoice method.

Example





CalcFrameOffset



This function block calculates an output frame by offsetting the input frame.

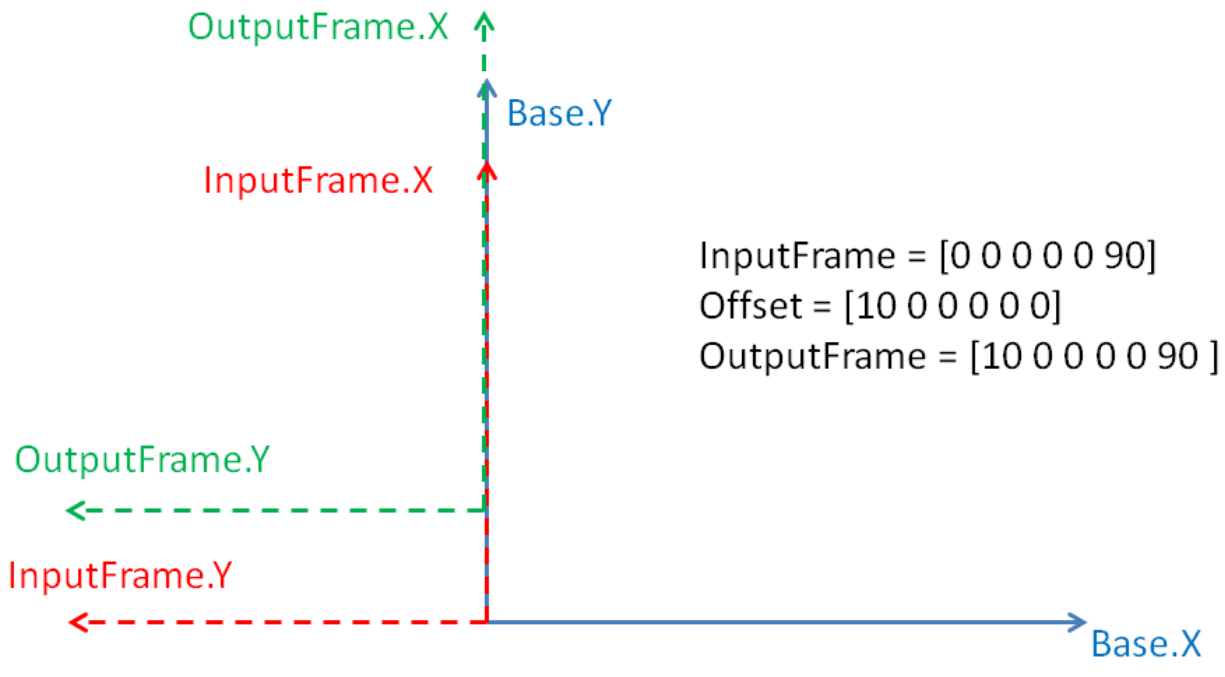
Library

Math Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	InputFrame	VECTOR	An array of values. The specific meaning of each value depends upon the Coordinate System specified and the mechanism, and the context. For example, the values could be world space positions or velocities.	
V	Offset	VECTOR	An array of values. The specific meaning of each value depends upon the Coordinate System specified and the mechanism, and the context. For example, the values could be world space positions or velocities.	
V	OutputFrame	VECTOR	An array of values. The specific meaning of each value depends upon the Coordinate System specified and the mechanism, and the context. For example, the values could be world space positions or velocities.	
VAR_INPUT			Default	
B	EN	BOOL	Enables the function.	FALSE
VAR_OUTPUT				
B	ENO	BOOL	High if the function is executing normally.	

Example





Getting Started with PackML Toolbox

Requirements for v350

To use the PackML Toolbox, your project must also contain the following:

Firmware libraries:

- PROCONOS

User libraries:

The following User Libraries must be listed above the PackML Toolbox and in the following order:

- Yaskawa_Toolbox (v300 or higher)

Using the PackML Toolbox

See Yaskawa's [Understanding PackML Webinar](#) for an in depth look at this toolbox.



PackML Revision History

Current Version:

2018-01-23: v340 Released

- 1) Altered the CM_Event and UN_Event FBs to pass the cfg_Event as an IN-OUT variable of EventCfgArray datatype.

Previous Versions:

2016-10-19: v302 Released

- 1) Altered the CM_Event and UN_Event FBs to pass the cfg_Event as an IN-OUT variable of EventCfgArray datatype.
- 2) Added EventNumber[int] as an input to the CM_Event and UN_Event FBs.
- 3) Removed the Prefix input from the CM_Event block.
- 4) Removed the entire EM event level. This level is deemed unnecessary.
- 5) Removed StS_Latched output of CM_Event block.
- 6) Many changes to the event handling. The goal of this revision was to move away from the fixed event handling method of built-in Servopack, Motion and Controller events in favor of User-defined events that could include messages looked up from the other built-in lists.
- 7) Added "UnitMachine.EM[EM_Index].ModuleActive AND" to CM_ControlInputs function in combination with the CM Mask to determine if the CM ModuleActive bit should be ON. This is so Events can be suppressed if the module is deactivated. Also made the same change in EM_ModuleSummation except commented out the rewrite of the CM_ModuleActive bit.
- 8) Added EventBoxNumber to EventStructs.
- 9) Removed the ResetFirstOut Input from the UN_EventSummation block. It was redundant since we could not think of a case where we would reset all active events but leave the FirstOut event still showing.

2016-07-06: v301 Released

- 1) Reconciled to the latest specification for PackML according to ISA-TR88 00 02 Edition 2. Rev 3D, 11/2014.
 - Automatic Mode renamed to Production Mode.
 - Added Unit Machine Layer to the PackTags. This is PackMLv30 datatype and contains PMLs, PMLc and PMLa.
 - Edited PackML datatypes to conform as closely as possible to v3.0 spec for Minimum Supported Set of PackTags.
 - Renamed Datatypes file and POUs to add the Prefix "PML_". This helps avoid superfluous warnings in the user's project.
- 2) Inputs for remote control have been removed from the PackML_State_Diagram function block.
- 3) Event categories was changed from a DWORD datatype to a DINT datatype. Most users set a category number for severity with lower numbers being higher severity (i.e. cat 1 = safety-type fault.)
- 4) Moved the Revision History from a POU into the comment section of the PML_FB_Palette POU to avoid nuisance Empty Worksheet Warnings.
- 5) renamed to v301beta.
- 6) Added AlarmHistory to the UN.admin datatypes.

7) Added UN_Event function block, similar to CM_Event, to capture events that are part of the general machine and not part of any particular Control Module.

8) Removed the Internal R_TRIGs of the State Diagram.

2015-01-31 v300 Released

1) Identical to v202, but recompiled specifically for MotionWorks IEC v3.x.

2012-03-28: v202 Released

1) Modified CM_Control_Inputs Function Block to turn off all CM commands if the EM is not active. Previously commands would still be sent unless the particular CM was deactivated.

2012-02-26: v201 Released

1) First official release

2) Updated Math Toolbox link

3) Improved interlocking in the PackML_State_Diagram for Stop and Abort. There were instances on the beta applications where the control could get stuck in a particular state.



Data Type: ControlModule_Array

Supporting array used to pass commands and machine status to individual Control Modules. The toolbox supports up to 16 Control Modules numbered 0 to 15.

Data Type Declaration

ControlModule_ARRAY : ARRAY[0..15] of PackML_Module_Commands_STRUCT;



Data Type: EquipmentModule_Array

Supporting Array used to pass commands and machine status to individual Equipment Modules. Up to 16 Equipment Modules are supported, numbered 0 to 15.

Data Type Declaration

EquipmentModule_ARRAY : ARRAY[0..15] of EquipmentModule_STRUCT;



Data Type: EquipmentModule_STRUCT

Supporting data type used by [EquipmentModule_ARRAY](#).

Data Type Declaration

*	Element	Data Type	Description	Usage
---	---------	-----------	-------------	-------

	MyEquipmentModule_STRUCT	EquipmentModule_STRUCT		
U	EnabledCMs	WORD	Number of enabled Control Modules contained in the Equipment Module	MyEquipmentModule_STRUCT.EnabledCMs
U	CMs_Active	WORD	Every bit in this word indicates if a control module is active	MyEquipmentModule_STRUCT.CMs_Active
U	CMs_NotDone	WORD	Every bit in this word indicates if a control module is done	MyEquipmentModule_STRUCT.CMs_NotDone
U	CM_InactiveMask	WORD	Every bit in this word indicates if a control module is Inactive	MyEquipmentModule_STRUCT.CM_InactiveMask
U	CM	ControlModule_ARRAY [0..15] OF PackML_Module_Commands_STRUCT	Array containing the Commands, Status and Active bits for the 16 Control Modules contained in the Equipment module	MyEquipmentModule_STRUCT.CM[0]...
U	Cmd_Reset	BOOL	Command to Reset the machine	MyEquipmentModule_STRUCT.Cmd_Reset
U	Sts_Resetting_SC	BOOL	When set, the machine is in the resetting state	MyEquipmentModule_STRUCT.Sts_Resetting_SC
U	Cmd_Start	BOOL	Command to Start the machine	MyEquipmentModule_STRUCT.Cmd_Start
U	Sts_Starting_SC	BOOL	When set, the machine is in the Starting state	MyEquipmentModule_STRUCT.Sts_Starting_SC
U	Cmd_Stop	BOOL	Command to Stop the machine	MyEquipmentModule_STRUCT.Cmd_Stop
U	Sts_Stopping_SC	BOOL	When set, the machine is in the Stopping state	MyEquipmentModule_STRUCT.Sts_Stopping_SC
U	Cmd_Hold	BOOL	Command to Hold the machine	MyEquipmentModule_STRUCT.Cmd_Hold
U	Sts_Holding_SC	BOOL	When set, the machine is in the Holding state	MyEquipmentModule_STRUCT.Sts_Holding_SC
U	Cmd_UnHold	BOOL	Command to Unhold the machine	MyEquipmentModule_STRUCT.Cmd_UnHold
U	Sts_UnHolding_SC	BOOL	When set, the machine is in the UnHolding state	MyEquipmentModule_STRUCT.Sts_UnHolding_SC
U	Cmd_Suspend	BOOL	Command to Suspend the machine	MyEquipmentModule_STRUCT.Cmd_Suspend
U	Sts_Suspending_SC	BOOL	When set, the machine is in the Suspending state	MyEquipmentModule_STRUCT.Sts_Suspending_SC
U	Cmd_UnSuspend	BOOL	Command to UnSuspend the machine	MyEquipmentModule_STRUCT.Cmd_UnSuspend
U	Sts_UnSuspending_SC	BOOL	When set, the machine is in the UnSuspending state	MyEquipmentModule_STRUCT.Sts_UnSuspending_SC
U	Cmd_Abort	BOOL	Command to Abort the machine	MyEquipmentModule_STRUCT.Cmd_Abort
U	Sts_Aborting_SC	BOOL	When set, the machine is in the Aborting state	MyEquipmentModule_STRUCT.Sts_Aborting_SC
U	Cmd_Clear	BOOL	Command to Clear the machine	MyEquipmentModule_STRUCT.Cmd_Clear
U	Sts_Clearing_SC	BOOL	When set, the machine is in the Clearing state	MyEquipmentModule_STRUCT.Sts_Clearing_SC
U	Sts_Executing_SC	BOOL	When set, the machine is in the Executing state	MyEquipmentModule_STRUCT.Sts_Executing_SC
U	Cmd_StateComplete	BOOL	Command to enter the Completing State	MyEquipmentModule_STRUCT.Cmd_StateComplete

U	Sts_Completing_SC	BOOL	When set, the machine is in the Completing state	MyEquipmentModule_STRUCT.Sts_Completing_SC
U	ModuleActive	BOOL	Indicates if the module is active to receive commands	MyEquipmentModule_STRUCT.ModuleActive

Data Type: Ingredient_ARRAY



An array that contains all the parameters for an ingredient

Data Type Declaration

Ingredient_ARRAY : ARRAY[0..31] OF Ingredient_STRUCT;



Data Type: Ingredient_STRUCT

A structure of parameters containing information for a specific ingredient. Support structure for [Ingredient_ARRAY](#).

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyIngredient_STRUCT	Ingredient_STRUCT		
U	ID	INT	ID value assigned to the ingredient	MyIngredient_STRUCT.ID
U	Parameter	Parameter_ARRAY [0..9] OF Parameter_STRUCT	An array of parameters used for the specified Ingredient	MyIngredient_STRUCT.Parameter[0]...

Data Type: Limit_ARRAY



An array containing user defined machine limits.

Data Type Declaration

Limit_ARRAY : ARRAY[0..9] OF Limit_STRUCT;



Data Type: Limit_STRUCT

Supporting structure for [Limit_ARRAY](#).

Data Type Declaration

	Element	Data Type	Description	Usage
*	MyLimit_STRUCT	Limit_STRUCT		
U	ID	INT	User defined ID for the limit, 0000 reserved for no limit assigned	MyLimit_STRUCT.ID
U	Name	STRING	Literal name for the limit	MyLimit_STRUCT.Name
U	Unit	STRING	Unit of the limit value	MyLimit_STRUCT.Unit
U	Value	REAL	Value assigned to the limit	MyLimit_STRUCT.Value



Data Type: Node_ARRAY

Array that contains information used to coordinating machine nodes in a cell of multiple units. The array can be expanded as needed.

Data Type Declaration

Node_ARRAY : ARRAY[0..7] OF Node_STRUCT;



Data Type: Node_STRUCT

Supporting structure for [Node_ARRAY](#).

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyNode_STRUCT	Node_STRUCT		
U	Number	INT	A chosen unique number of the Upstream/Down-stream PackML machine	MyNode_STRUCT.Number
U	ControlCmdNumber	INT	User defined command to be sent from one node on the network to another	MyNode_STRUCT.ControlCmdNumber
U	CmdValue	INT	A value to be associated with the ControlCmdNumber such as speed, or the mode requested to change to	MyNode_STRUCT.CmdValue
U	Parameter	Parameter_ARRAY [0..9] OF Parameter_STRUCT	An array of parameter names, values, and units of the parameter	MyNode_STRUCT.Parameter [0]...



Data Type: PackML_Commands_STRUCT

Supporting structure for [PackTags_Commands_STRUCT](#)

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyPackML_Commands_STRUCT	PackML_Commands_STRUCT		
U	Mode	DINT	Mode command, Mode's can be customized according to the PackML standard or for the user's needs. See template documentation for more on mode customization	MyPackML_Commands_STRUCT.Mode
U	Reset	BOOL	Command to Reset the Machine	MyPackML_Commands_STRUCT.Reset
U	Start	BOOL	Command to Start the Machine	MyPackML_Commands_STRUCT.Start
U	Stop	BOOL	Command to Stop the Machine	MyPackML_Commands_STRUCT.Stop
U	Hold	BOOL	Command to Hold the Machine	MyPackML_Commands_STRUCT.Hold
U	UnHold	BOOL	Command to UnHold the Machine	MyPackML_Commands_STRUCT.UnHold
U	Suspend	BOOL	Command to Suspend the Machine	MyPackML_Commands_STRUCT.Suspend
U	UnSuspend	BOOL	Command to UnSuspend the Machine	MyPackML_Commands_STRUCT.UnSuspend
U	Abort	BOOL	Command to Abort the Machine	MyPackML_Commands_STRUCT.Abort
U	Clear	BOOL	Command to Clear the Machine	MyPackML_Commands_STRUCT.Clear
U	StateComplete	BOOL	Command to enter the Completing State	MyPackML_Commands_STRUCT.StateComplete

Data Type: PackML_Module_Commands_STRUCT



Supporting data type used by [ControlModule_ARRAY](#).

Data Type Declaration

*	Element	Data Type	Description	Usage
---	---------	-----------	-------------	-------

	MyPackML_Module_Commands_STRUCT	PackML_Module_Commands_STRUCT		
U	Cmd_Reset	BOOL	Command to Reset the machine by moving from the Stopped state to the Resetting state.	MyPackML_Module_Commands_STRUCT.Cmd_Reset
U	Sts_Resetting_SC	BOOL	When set, the machine is in the resetting state.	MyPackML_Module_Commands_STRUCT.Sts_Resetting_SC
U	Cmd_Start	BOOL	Command to Start the machine by moving from the Idle state to the Execute state.	MyPackML_Module_Commands_STRUCT.Cmd_Start
U	Sts_Starting_SC	BOOL	When set, the machine is in the Starting state.	MyPackML_Module_Commands_STRUCT.Sts_Starting_SC
U	Cmd_Stop	BOOL	Command to Stop the machine by moving from any state other than Aborting, Aborted or Clearing to the Stopping state.	MyPackML_Module_Commands_STRUCT.Cmd_Stop
U	Sts_Stopping_SC	BOOL	When set, the machine is in the Stopping state.	MyPackML_Module_Commands_STRUCT.Sts_Stopping_SC
U	Cmd_Hold	BOOL	Command to Hold the machine by moving from the Execute state to the Holding state.	MyPackML_Module_Commands_STRUCT.Cmd_Hold
U	Sts_Holding_SC	BOOL	When set, the machine is in the Holding state.	MyPackML_Module_Commands_STRUCT.Sts_Holding_SC
U	Cmd_UnHold	BOOL	Command to Unhold the machine by moving from the Held state to the UnHolding state.	MyPackML_Module_Commands_STRUCT.Cmd_UnHold
U	Sts_UnHolding_SC	BOOL	When set, the machine is in the UnHolding state.	MyPackML_Module_Commands_STRUCT.Sts_UnHolding_SC
U	Cmd_Suspend	BOOL	Command to Suspend the machine by moving from the Execute state to the Suspending state.	MyPackML_Module_Commands_STRUCT.Cmd_Suspend
U	Sts_Suspending_SC	BOOL	When set, the machine is in the Suspending state.	MyPackML_Module_Commands_STRUCT.Sts_Suspending_SC
U	Cmd_UnSuspend	BOOL	Command to UnSuspend the machine by moving from the Suspended state to the UnSuspending state.	MyPackML_Module_Commands_STRUCT.Cmd_UnSuspend
U	Sts_UnSuspending_SC	BOOL	When set, the machine is in the UnSuspending state.	MyPackML_Module_Commands_STRUCT.Sts_UnSuspending_SC
U	Cmd_Abort	BOOL	Command to Abort the machine by moving from any state except Aborting or Aborted to the Aborting state.	MyPackML_Module_Commands_STRUCT.Cmd_Abort
U	Sts_Aborting_SC	BOOL	When set, the machine is in the Aborting state.	MyPackML_Module_Commands_STRUCT.Sts_Aborting_SC
U	Cmd_Clear	BOOL	Command to Clear the machine by moving from the Aborted state to the Clearing state.	MyPackML_Module_Commands_STRUCT.Cmd_Clear
U	Sts_Clearing_SC	BOOL	When set, the machine is in the Clearing state.	MyPackML_Module_Commands_STRUCT.Sts_Clearing_SC
U	Sts_Executing_SC	BOOL	When set, the machine is in the Executing state.	MyPackML_Module_Commands_STRUCT.Sts_Executing_SC
U	Cmd_StateComplete	BOOL	Command to enter the Completing State from the Execute state.	MyPackML_Module_Commands_STRUCT.Cmd_StateComplete
U	Sts_Completing_SC	BOOL	When set, the machine is in the Completing state.	MyPackML_Module_Commands_STRUCT.Sts_Completing_SC
U	ModuleActive	BOOL	Indicates if the Control Module is active and able to receive commands. Can also be used to enable the detection of Events for the Control Module	MyPackML_Module_Commands_STRUCT.ModuleActive



Data Type: PackML_States_STRUCT

Supporting structure for [PackTags_Status_STRUCT](#).

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyPackML_States_STRUCT	PackML_States_STRUCT		
U	Clearing	BOOL	Indicates the machine is in the Clearing State.	MyPackML_States_STRUCT.Clearing
U	Stopped	BOOL	Indicates the machine is in the Stopped State.	MyPackML_States_STRUCT.Stopped
U	Starting	BOOL	Indicates the machine is in the Starting State.	MyPackML_States_STRUCT.Starting
U	Idle	BOOL	Indicates the machine is in the Idle State.	MyPackML_States_STRUCT.Idle
U	Suspended	BOOL	Indicates the machine is in the Suspended State.	MyPackML_States_STRUCT.Suspended
U	Execute	BOOL	Indicates the machine is in the Execute State.	MyPackML_States_STRUCT.Execute
U	Stopping	BOOL	Indicates the machine is in the Stopping State.	MyPackML_States_STRUCT.Stopping
U	Aborting	BOOL	Indicates the machine is in the Aborting State.	MyPackML_States_STRUCT.Aborting
U	Aborted	BOOL	Indicates the machine is in the Aborted State.	MyPackML_States_STRUCT.Aborted
U	Holding	BOOL	Indicates the machine is in the Holding State.	MyPackML_States_STRUCT.Holding
U	Held	BOOL	Indicates the machine is in the Held State.	MyPackML_States_STRUCT.Held
U	UnHolding	BOOL	Indicates the machine is in the Unholding State.	MyPackML_States_STRUCT.UnHolding
U	Suspending	BOOL	Indicates the machine is in the Suspending State.	MyPackML_States_STRUCT.Suspending
U	UnSuspending	BOOL	Indicates the machine is in the Unsuspending State.	MyPackML_States_STRUCT.UnSuspending
U	Resetting	BOOL	Indicates the machine is in the Resetting State.	MyPackML_States_STRUCT.Resetting
U	Completing	BOOL	Indicates the machine is in the Completing State.	MyPackML_States_STRUCT.Completing
U	Complete	BOOL	Indicates the machine is in the Complete State.	MyPackML_States_STRUCT.Complete



Data Type: PackTags_Admin_STRUCT

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyPackTags_Admin_STRUCT	PackTags_Admin_STRUCT		
U	Alarm	EventHistoryArray	Array of Event information.	MyPackTags_Admin_STRUCT.Alarm[0]...
U	StateCurrentTime	DINT	Amount of time spent in the current state	MyPackTags_Admin_STRUCT.StateCurrentTime
U	StateCumulativeTime	StateCumulativeArray	Array containing all the times spent in the different states	MyPackTags_Admin_STRUCT.StateCumulativeTime[0]...
U	ModeCurrentTime	DINT	Amount of time spent in the current mode.	MyPackTags_Admin_STRUCT.ModeCurrentTime
U	ModeCumulativeTime	DINT_Array32	Array containing all the times spent in the different modes.	MyPackTags_Admin_STRUCT.ModeCumulativeTime[0]
U	AccumTimeSinceReset	DINT	Time since the cumulative and current times have been reset.	MyPackTags_Admin_STRUCT.AccumTimeSinceReset
U	ResetAllTimes	BOOL	Command to reset all timers.	MyPackTags_Admin_STRUCT.ResetAllTimes
U	ResetCurrentModeTimes	BOOL	Command to reset all Current Times being tracked.	MyPackTags_Admin_STRUCT.ResetCurrentModeTimes
U	TimeRollover	BOOL	Warning when the timer is approaching a roll over.	MyPackTags_Admin_STRUCT.TimeRollover
U	ProdProcessed	DINT	Cumulative number of primary packages processed since the machine's counters and timers were reset.	MyPackTags_Admin_STRUCT.ProdProcessed
U	DefectiveProd	DINT	Cumulative number of defective packages processed since the machine's counters and timers were reset.	MyPackTags_Admin_STRUCT.DefectiveProd
U	ReWorkProd	DINT	Cumulative number of re-workable primary packages processed.	MyPackTags_Admin_STRUCT.ReWorkProd
U	UpstreamMessage	DINT		MyPackTags_Admin_STRUCT.UpstreamMessage
U	DownstreamMessage	DINT		MyPackTags_Admin_STRUCT.DownstreamMessage
U	CurrentUpstreamNodeID	DINT		MyPackTags_Admin_STRUCT.CurrentUpstreamNodeID
U	CurrentDownstreamNodeID	DINT		MyPackTags_Admin_STRUCT.CurrentDownstreamNodeID



Data Type: PackTags_Commands_STRUCT

Data Type Declaration

*	Element	Data Type	Description	Usage
---	---------	-----------	-------------	-------

	MyPackTags_Commands_STRUCT	PackTags_Commands_STRUCT		
U	UnitMode	DINT	Unit Mode Commanded	MyPackTags_Commands_STRUCT.UnitMode
U	UnitModeChangeRequest	BOOL	1 = Change Machine Mode to Commanded Value	MyPackTags_Commands_STRUCT.UnitModeChangeRequest
U	ProcMode	DINT	Procedure Mode Commanded	MyPackTags_Commands_STRUCT.ProcMode
U	ProcModeChangeRequest	BOOL	1 = Change Procedure Mode to Commanded Value	MyPackTags_Commands_STRUCT.ProcModeChangeRequest
U	CurMachSpeed	DINT	Machine Speed - In Primary Line Packages	MyPackTags_Commands_STRUCT.CurMachSpeed
U	MatReady	DWORD	Material Interlocks	MyPackTags_Commands_STRUCT.MatReady
U	MatLow	DWORD	Material Interlocks	MyPackTags_Commands_STRUCT.MatLow
U	ResetPackMLTimes	BOOL	1 = Reset PackML Current Mode and State Current/Cumulative Times	MyPackTags_Commands_STRUCT.ResetPackMLTimes
U	CntrlCmd	DINT	Provides an alternate method of moving through the state diagram	MyPackTags_Commands_STRUCT.CntrlCmd
U	StateCmd	PackML_Commands_STRUCT	A structure for Coordinating machine nodes	MyPackTags_Commands_STRUCT.StateCmd
U	StateChangeRequest	BOOL	Indicates the state machine should proceed to the target state	MyPackTags_Commands_STRUCT.StateChangeRequest
U	CfgRemoteCmdEnable	BOOL		MyPackTags_Commands_STRUCT.CfgRemoteCmdEnable
U	RemoteModeCmd	DINT		MyPackTags_Commands_STRUCT.RemoteModeCmd
U	RemoteModeCmdChgReq	BOOL		MyPackTags_Commands_STRUCT.RemoteModeCmdChgReq
U	RemoteStateCmd	DINT		MyPackTags_Commands_STRUCT.RemoteStateCmd
U	RemoteStateCmdChgReq	BOOL		MyPackTags_Commands_STRUCT.RemoteStateCmdChgReq
U	TargetDownstreamNodeID	DINT		MyPackTags_Commands_STRUCT.TargetDownstreamNodeID
U	TargetUpstreamNodeID	DINT		MyPackTags_Commands_STRUCT.TargetUpstreamNodeID
U	ChangeNodeServicedUpstream	DINT		MyPackTags_Commands_STRUCT.ChangeNodeServicedUpstream
U	ChangeNodeServicedDownstream	DINT		MyPackTags_Commands_STRUCT.ChangeNodeServicedDownstream
THE FOLLOWING FIELDS ARE INITIALLY COMMENTED OUT TO SAVE MEMORY WHEN NOT REQUIRED				
U	Node	Node_ARRAY	Node (machine) interface & ID structure	MyPackTags_Commands_STRUCT.Node[0]...
U	ProcessVariables	ProcessVariable_ARRAY	Machine Engineering Parameters	MyPackTags_Commands_STRUCT.ProcessVariables[0]...
U	Product	Product_ARRAY	Machine Product/Recipe Parameters	MyPackTags_Commands_STRUCT.Product[0]...
U	Limits	Limit_ARRAY	Machine Parameter Prograble Limits	MyPackTags_Commands_STRUCT.Limits[0]...



Data Type: PackTags_Status_STRUCT

Data Type Declaration

*	Element	Data Type	Description	Usage
---	---------	-----------	-------------	-------

	MyPackTags_Status_STRUCT	PackTags_Status_STRUCT		
U	CommandRejected	BOOL	If an invalid request is given and rejected, this bit will be set	MyPackTags_Status_STRUCT.CommandRejected
U	UnitModeCurrent	DINT	Current Machine Mode	MyPackTags_Status_STRUCT.UnitModeCurrent
U	UnitModeCurBit	DWORD	Current Machine Mode Bit	MyPackTags_Status_STRUCT.UnitModeCurBit
U	UnitModeCurrentName	STRING	Current Machine Mode Name	MyPackTags_Status_STRUCT.UnitModeCurrentName
U	UnitModeRequested	BOOL	[1 = Acknowledges that a unit mode change has been requested]	MyPackTags_Status_STRUCT.UnitModeRequested
U	UnitModeChangeInProgress	BOOL	[1 = Requested unit mode change in process]	MyPackTags_Status_STRUCT.UnitModeChangeInProgress
U	ProcModeCurrent	DINT	Current Procedure Mode	MyPackTags_Status_STRUCT.ProcModeCurrent
U	ProcModeRequested	BOOL	[1 = Acknowledges that a procedure mode change has been requested]	MyPackTags_Status_STRUCT.ProcModeRequested
U	ProcModeChangeInProgress	BOOL	[1 = Requested procedure mode change in process]	MyPackTags_Status_STRUCT.ProcModeChangeInProgress
U	StateCurrent	DINT	Current Machine State	MyPackTags_Status_STRUCT.StateCurrent
U	StateCurBit	DWORD		MyPackTags_Status_STRUCT.StateCurBit
U	StateCurrentName	STRING	Current Machine State Name	MyPackTags_Status_STRUCT.StateCurrentName
U	StateRequested	BOOL	[1 = Acknowledges that a state change has been requested]	MyPackTags_Status_STRUCT.StateRequested
U	StateChangeInProgress	BOOL	[1 = Requested state change in process]	MyPackTags_Status_STRUCT.StateChangeInProgress
U	StateChangeProgress	DINT	Percent Complete of current state	MyPackTags_Status_STRUCT.StateChangeProgress
U	StateLastCompleted	DINT	Machine state last completed	MyPackTags_Status_STRUCT.StateLastCompleted
U	SeqNumber	DINT		MyPackTags_Status_STRUCT.SeqNumber
U	CurMachSpd	DINT	Current Machine Speed In Primary Line Packages Per Minute	MyPackTags_Status_STRUCT.CurMachSpd
U	MatReady	DWORD	Material Interlocks	MyPackTags_Status_STRUCT.MatReady
U	MatLow	DWORD	Material Interlocks	MyPackTags_Status_STRUCT.MatLow
U	MachDesignSpeed	REAL	Speed the machine is designed to operate at in it's installed environment	MyPackTags_Status_STRUCT.MachDesignSpeed
U	State	PackML_States_STRUCT		MyPackTags_Status_STRUCT.State
U	ModeChangeNotAllowed	BOOL	This bit is set if an invalid mode change is requested and ignored	MyPackTags_Status_STRUCT.ModeChangeNotAllowed
U	MachCycle	DINT	Indicates the number of completed machine cycles with or without product	MyPackTags_Status_STRUCT.MachCycle
U	ProdRatio	DINT	Quantity of primary packages per current package being produced	MyPackTags_Status_STRUCT.ProdRatio
U	Dirty	BOOL	Set when the machine becomes dirty and machine must run through a cleaning cycle before production continues	MyPackTags_Status_STRUCT.Dirty
U	Clean	BOOL	Bit is set after a cleaning cycle and reset once production begins again	MyPackTags_Status_STRUCT.Clean

U	TimeToDirty	DINT	Time remaining until machine becomes dirty again	MyPackTags_Status_STRUCT.TimeToDirty
U	EquipmentAllocatedToUnitModeID	DINT	Allocating a machine to operating a different mode than another duplicate machine	MyPackTags_Status_STRUCT.EquipmentAllocatedToUnitModeID
U	MachineReusableForUnitModeID	DINT	Indicates machine does not require immediate cleaning and can resume production in a specific time window	MyPackTags_Status_STRUCT.MachineReusableForUnitModeID
U	MachineReusableTimeLeft	DINT	Amount of time left for a system to be reusable for a specific Unit mode	MyPackTags_Status_STRUCT.MachineReusableTimeLeft
U	MachineStoringProductID	DINT	For machines that have a storing capability	MyPackTags_Status_STRUCT.MachineStoringProductID
U	MachineTransferringProductID	DINT	For machines used in conveying, compacting and/or separating product and transferring it to other machinery	MyPackTags_Status_STRUCT.MachineTransferringProductID
THE FOLLOWING FIELDS COME INITIALLY COMMENTED OUT TO SAVE MEMORY WHEN NOT USED				
U	Node	Node_ARRAY	Node (machine) interface & ID structure	MyPackTags_Status_STRUCT.Node[0]...
U	ProcessVariables	ProcessVariable_ARRAY	Machine Engineering Parameters	MyPackTags_Status_STRUCT.ProcessVariables[0]...
U	Product	Product_ARRAY	Machine Product/Recipe Parameters	MyPackTags_Status_STRUCT.Product[0]...
U	Limits	Limit_ARRAY	Machine Parameter Programmable Limits	MyPackTags_Status_STRUCT.Limits[0]...



Data Type: Parameter_ARRAY

An array containing the names, units and values of a given parameter

Data Type Declaration

Parameter_ARRAY : ARRAY[0..9] OF Parameter_STRUCT;



Data Type: Parameter_STRUCT

Supporting Structure for [Parameter_ARRAY](#)

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyParameter_STRUCT	Parameter_STRUCT		
U	ID	DINT	ID value assigned to the parameter	MyParameter_STRUCT.ID
U	Name	STRING	Literal description of the parameter	MyParameter_STRUCT.Name
U	Unit	STRING	Unit associated with the given parameter	MyParameter_STRUCT.Unit
U	Value	REAL	Numeric value associated with the given parameter	MyParameter_STRUCT.Value



Data Type: ProcessVariable_ARRAY

An array containing the names, units and values of a given parameter that are used across multiple machines and can be displayed on an HMI screen.

Data Type Declaration

```
ProcessVariable_ARRAY : ARRAY[0..9] OF ProcessVariable_STRUCT;
```



Data Type: ProcessVariable_STRUCT

Supporting structure for [ProcessVariable_ARRAY](#).

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyProcessVariable_STRUCT	ProcessVariable_STRUCT		
U	ID	DINT	ID value assigned to the parameter	MyProcessVariable_STRUCT.ID
U	Name	STRING	Literal description of the parameter. Can also be displayed on an HMI screen.	MyProcessVariable_STRUCT.Name
U	Unit	STRING_5	Unit associated with the given parameter. Can also be displayed on an HMI screen.	MyProcessVariable_STRUCT.Unit
U	Value	REAL	Numeric value associated with the given parameter. Can also be displayed on an HMI screen.	MyProcessVariable_STRUCT.Value

Data Type: Product_ARRAY



An array containing product information

Data Type Declaration

Product_ARRAY : ARRAY[0..9] OF Product_STRUCT;



Data Type: Product_STRUCT

A structure containing product information

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyProduct_STRUCT	Product_STRUCT		
U	ProductID	INT	Used to indicate to the machine what product it is producing, also displayed on all HMI screens	MyProduct_STRUCT.ProductID
U	ProcessVariables	ProcessVariable_ARRAY	Array of information containing parameters for multiple machines	MyProduct_STRUCT.ProcessVariables
U	Ingredients	Ingredient_ARRAY	An array containing all information regarding an ingredient	MyProduct_STRUCT.Ingredients



Data Type: UNitMachine_STRUCT

Contains all the information about the machine's current state for each EM and CM.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyUNitmachine_STRUCT	UNitmachine_STRUCT		
U	PackML_StateControlReady	BOOL	Indicates when the PackML_State_Diagram function block is ready to control the machine	MyUNitmachine_STRUCT.PackML_StateControlReady
U	EnabledEMs	INT	Number of enabled equipment modules in the machine	MyUNitmachine_STRUCT.EnabledEMs
U	EMs_Active	WORD	Every bit in this word indicates which equipment modules are Active	MyUNitmachine_STRUCT.EMs_Active
U	EMs_NotDone	WORD	Every bit in this word indicates which equipment modules are Not Done	MyUNitmachine_STRUCT.EMs_NotDone
U	EM_InactiveMask	WORD	Every bit in this word indicates which equipment modules are Inactive	MyUNitmachine_STRUCT.EM_InactiveMask
U	EM	EquipmentModule_ARRAY	ARRAY[0..15] of EquipmentModule_STRUCT	MyUNitmachine_STRUCT.EM
U	Sts_Resetting_SC	BOOL	When set, the machine is in the resetting state	MyUNitmachine_STRUCT.Sts_Resetting_SC
U	Sts_Starting_SC	WORD	When set, the machine is in the Starting state	MyUNitmachine_STRUCT.Sts_Starting_SC
U	Sts_Stopping_SC	WORD	When set, the machine is in the Stopping state	MyUNitmachine_STRUCT.Sts_Stopping_SC
U	Sts_Holding_SC	WORD	When set, the machine is in the Holding state	MyUNitmachine_STRUCT.Sts_Holding_SC
U	Sts_UnHolding_SC	WORD	When set, the machine is in the UnHolding state	MyUNitmachine_STRUCT.Sts_UnHolding_SC
U	Sts_Suspending_SC	BOOL	When set, the machine is in the Suspending state	MyUNitmachine_STRUCT.Sts_Suspending_SC
U	Sts_UnSuspending_SC	BOOL	When set, the machine is in the UnSuspending state	MyUNitmachine_STRUCT.Sts_UnSuspending_SC
U	Sts_Aborting_SC	BOOL	When set, the machine is in the Aborting state	MyUNitmachine_STRUCT.Sts_Aborting_SC
U	Sts_Clearing_SC	BOOL	When set, the machine is in the Clearing state	MyUNitmachine_STRUCT.Sts_Clearing_SC
U	Sts_Executing_SC	BOOL	When set, the machine is in the Executing state	MyUNitmachine_STRUCT.Sts_Executing_SC
U	Sts_Completing_SC	BOOL	When set, the machine is in the Completing state	MyUNitmachine_STRUCT.Sts_Completing_SC



Enumerated Types for PackML Toolbox

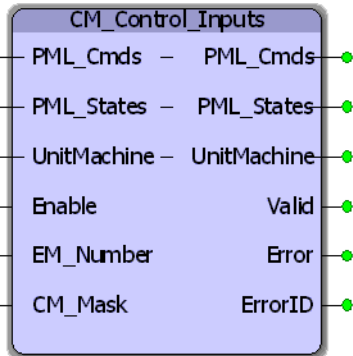
Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to zero-based integers (INT). Therefore, the first value equates to zero, the second to 1, etc. The format for enumerated types is as follows: ENUM:(0, 1, 2...) as displayed in the example below (MC_BufferMode#Aborting).

Enumerated Types Declaration

Enumerated Type	#INT Value	Enum Value	Description
PackMLState	ENUM Type for indicating the PackML state.		
	0	Undefined	
	1	Clearing	
	2	Stopped	
	3	Starting	
	4	Idle	
	5	Suspended	
	6	Execute	
	7	Stopping	
	8	Aborting	
	9	Aborted	
	10	Holding	
	11	Held	
	12	UnHolding	
	13	Suspending	
	14	UnSuspending	
	15	Resetting	
	16	Completing	
17	Complete		



CM_Control_Inputs



The CM_Control_Inputs function block passes the high level commands from the PackML_StateControl into each of the enabled and active Control Modules.

Library

Pack ML Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	PML_Cmds	PackML_Commands_STRUCT	Structure that contains the current Unit mode of operation and the commands sent by PackML_StateMachine.	
V	PML_States	PackML_States_STRUCT	Structure containing information about the current state of the machine.	
V	UnitMachine	UnitMachine_STRUCT	Structure containing all the information about the machines current state and mode of operation for all EMs and CMs.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	EM_Number	INT	The EM number corresponding to the EM in which this FB is located.	INT#0
V	CM_Mask	WORD	Mask to deactivate CMs. When a CM is deactivated, commands will not be sent down to the CM, for testing purposes. Each bit corresponds to the same number CM to deactivate. (Example: to deactivate CM_3, set CM_Mask.X3.	WORD#16#0000

VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

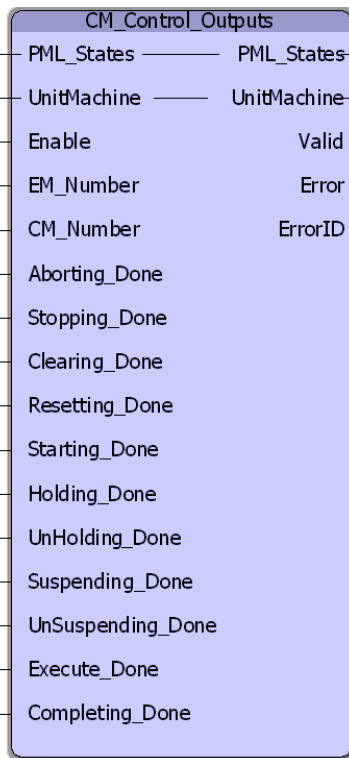
- See the PackML template documentation for further details on recommended usage.

Error Description

See the [Function Block ErrorID](#) list.



CM_Control_Outputs



The CM_Control_Outputs function block sets the State Complete bits for the control module to be passed up and assembled into the Equipment Module status in the EM00_ModuleControl worksheet.

Library

Pack ML Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	PML_States	PackML_States_STRUCT	Structure containing information about the current state of the machine.
V	UnitMachine	UnitMachine_STRUCT	Structure containing all the information about the machines current state and mode of operation for all EMs and CMs.

VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	EM_Number	INT	The EM number corresponding to the EM in which this FB is located.	INT#0
V	CM_Number	WORD	The CM number corresponding to the CM in which this FB is located.	WORD#0
V	Aborting_Done	BOOL	Setting this bit indicates that the current CM is done 'Aborting' and is ready to move to the next state.	FALSE
V	Stopping_Done	BOOL	Setting this bit indicates that the current CM is done 'Stopping' and is ready to move to the next state.	FALSE
V	Clearing_Done	BOOL	Setting this bit indicates that the current CM is done 'Clearing' and is ready to move to the next state.	FALSE
V	Resetting_Done	BOOL	Setting this bit indicates that the current CM is done 'Resetting' and is ready to move to the next state.	FALSE
V	Starting_Done	BOOL	Setting this bit indicates that the current CM is done 'Starting' and is ready to move to the next state.	FALSE
V	Holding_Done	BOOL	Setting this bit indicates that the current CM is done 'Holding' and is ready to move to the next state.	FALSE
V	UnHolding_Done	BOOL	Setting this bit indicates that the current CM is done 'UnHolding' and is ready to move to the next state.	FALSE
V	Suspending_Done	BOOL	Setting this bit indicates that the current CM is done 'Suspending' and is ready to move to the next state.	FALSE
V	UnSuspending_Done	BOOL	Setting this bit indicates that the current CM is done 'UnSuspending' and is ready to move to the next state.	FALSE
V	Execute_Done	BOOL	Setting this bit indicates that the current CM is done 'Executing' and is ready to move to the next state.	FALSE
V	Completing_Done	BOOL	Setting this bit indicates that the current CM is done 'Completing' and is ready to move to the next.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

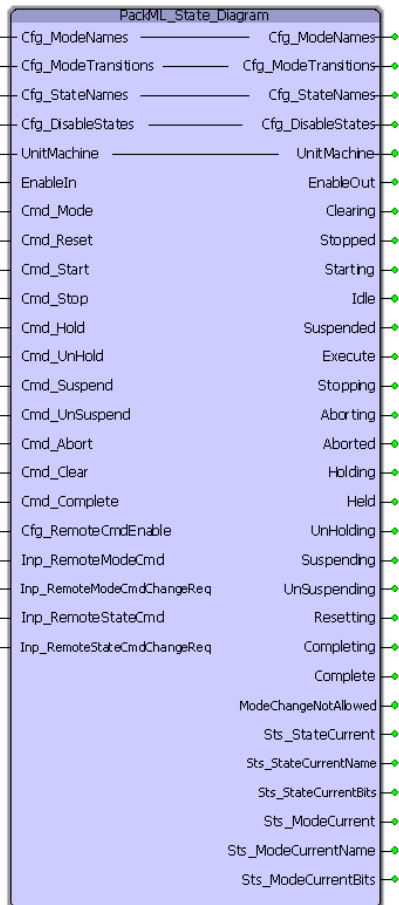
- See template documentation for further details on recommended usage.

Error Description

See the [Function Block ErrorID](#) list.



PackML_State_Diagram



The PackML_State_Diagram function block handles the operation of the state machine, including mode and state transitions, as defined in the OMAC PackML specification. This function block, when enabled, initializes the machine to be in mode 3 (Manual Mode) and in the Stopped state.

Library

Pack ML Toolbox

Parameters

*	Parameter	Data Type	Description
	VAR_IN_OUT		

V	Cfg_ModeNames	STRING_Array32	An array of strings containing the names of the different Unit modes of operation.	
V	Cfg_ModeTransitions	DINT_Array32	An array of acceptable mode transition states. Mode changes into the NEW MODE can only be performed at the chosen states. Each element in the array represents a mode, and each bit in the array element represents a state. (Ex. To allow Mode Transitions for Mode 1 at Aborted (bit 9), Stopped (bit 2), and Idle (bit 4) states 0000 0000 0000 0000 0000 0010 0001 0100 = 16#0000_0214 = DINT#532 = Cfg_ModeTransitions[1])	
V	Cfg_StateNames	STRING_Array18	An array of strings containing the names of all the PackML states.	
V	Cfg_DisableStates	DINT_Array32	An array representing each mode and their states. Each mode can disable certain states.(Ex In Manual Mode (Mode 3) disable Holding (10), Held(11), UnHolding(12), Suspended(5), Suspending(13), UnSuspending(14),Completing(16), Complete(17) = 0000 0000 0000 0011 0111 1100 0010 0000 = 16#0003_7C20 = DINT#228384 = Cfg_DisableStates[3])	
V	UnitMachine	UNitMachine_STRUCT	Structure containing all the information about the machines current state and mode of operation for all EMs and CMs.	
VAR_INPUT				
B	EnableIn	BOOL	The function will continue to execute while the enable is held high	Default FALSE
V	Cmd_Mode	DINT	The value of the new mode the machine will transition to if possible. If the input remains unchanged, the machine will stay in the same mode of operation	DINT#0
V	Cmd_Reset	BOOL	Setting this bit sends the 'Restart' command to all enabled and active EMs if it is a legal transition from	FALSE
V	Cmd_Start	BOOL	Setting this bit sends the 'Start' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
V	Cmd_Stop	BOOL	Setting this bit sends the 'Stop' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
V	Cmd_Hold	BOOL	Setting this bit sends the 'Hold' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
V	Cmd_UnHold	BOOL	Setting this bit sends the 'UnHold' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE

V	Cmd_Suspend	BOOL	Setting this bit sends the 'Suspend' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
V	Cmd_UnSuspend	BOOL	Setting this bit sends the 'UnSuspend' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
V	Cmd_Abort	BOOL	Setting this bit sends the 'Abort' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
V	Cmd_Clear	BOOL	Setting this bit sends the 'Clear' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
V	Cmd_Complete	BOOL	Setting this bit sends the 'Complete' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
V	Cfg_RemoteModeCmd	DINT	The remotely requested mode to transition to	0
V	Inp_RemoteModeCmdChangeReq	BOOL	When this input is set, the machine will transition to the mode set by Cfg_RemoteModeCmd if it is a legal transition from the current state of the machine	FALSE
V	Inp_RemoteStateCmd	DINT	The remotely requested state to transition to	0
V	Inp_RemoteStateCmdChangeReq	BOOL	When this input is set, the machine will transition to the state set by Cfg_RemoteStateCmd if it is a legal transition from the current state of the machine	FALSE
VAR_OUTPUT				
B	EnableOut	BOOL	Indicates that the outputs of the function are valid	
V	Clearing	BOOL	When this bit is set, the machine is in the 'Clearing' state	
V	Stopped	BOOL	When this bit is set, the machine is in the 'Stopped' state	
V	Starting	BOOL	When this bit is set, the machine is in the 'Starting' state	
V	Idle	BOOL	When this bit is set, the machine is in the 'Idle' state	

V	Suspended	BOOL	When this bit is set, the machine is in the 'Suspended' state
V	Execute	BOOL	When this bit is set, the machine is in the 'Execute' state
V	Stopping	BOOL	When this bit is set, the machine is in the 'Stopping' state
V	Aborting	BOOL	When this bit is set, the machine is in the 'Aborting' state
V	Aborted	BOOL	When this bit is set, the machine is in the 'Aborted' state
V	Holding	BOOL	When this bit is set, the machine is in the 'Holding' state
V	Held	BOOL	When this bit is set, the machine is in the 'Held' state
V	UnHolding	BOOL	When this bit is set, the machine is in the 'UnHolding' state
V	Suspending	BOOL	When this bit is set, the machine is in the 'Suspending' state
V	UnSuspending	BOOL	When this bit is set, the machine is in the 'UnSuspending' state
V	Resetting	BOOL	When this bit is set, the machine is in the 'Resetting' state
V	Completing	BOOL	When this bit is set, the machine is in the 'Completing' state
V	Complete	BOOL	When this bit is set, the machine is in the 'Complete' state
V	ModeChangeNotAllowed	BOOL	When this bit is set, the requested Mode change isn't allowed and the machine will remain in the current mode and state.
V	Sts_StateCurrent	DINT	Number in decimal corresponding to the current state the machine is in
V	Sts_StateCurrentName	STRING	The name of the current state the machine is in
V	Sts_StateCurrentBits	DINT	DWORD indicating the current state the machine is in (Ex. If Sts_StateCurrentBits[x] = 1, then the machine is in State x)
V	Sts_ModeCurrent	DINT	Number in decimal corresponding to the current mode the machine is in
V	Sts_ModeCurrentName	STRING	The name of the current mode the machine is in
V	StsModeCurrentBits	DWORD	DWORD indicating the current mode the machine is in (Ex. If Sts_ModeCurrentBits[x] = 1, then the machine is in State x)

Notes

- Should always be enabled when program is running to ensure proper operation of the state machine.
- See template documentation for further details on recommended usage.



PackMLCommands_Init



The PackMLCommands_Init function block clears all commands and sets the machine to be in the stopped state.

Library

Pack ML Toolbox

Parameters

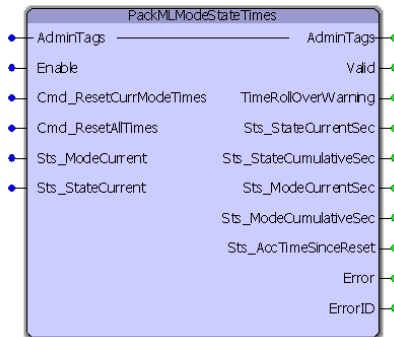
*	Parameter	Data Type	Description
VAR_IN_OUT			
V	INP_Pack-MLCommands	PackML_Module_Commands_STRUCT	Structure containing the current state and commanded actions
VAR_INPUT			Default
B	EN	BOOL	Enables the function. FALSE
VAR_OUTPUT			
B	ENO	BOOL	High if the function is executing normally.

Notes

- Intended to be executed when initially entering the stopped state to clear all previous commands.



PackMLModeStateTimes



The PackMLModeStateTimes function block keeps track of the times spent in each mode and state of operation for the machine.

Library

Pack ML Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	AdminTags	PackTags_Admin_STRUCT	Structure containing alarm data from the machine.	
V	UnitMachine	UNitMachine_STRUCT	Structure containing all the information about the machines current state and mode of operation for all EMs and CMs.	
VAR_INPUT			Default	
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
B	Cmd_ResetCurrModeTimes	BOOL	When set, all time counting will be stalled and all of the times being counted for the Sts_ModeCurrent will be cleared.	FALSE
B	Cmd_ResetAllTimes	BOOL	When set, all times being monitored will be reset to zero. Time counting will also be stalled while this input is held high.	FALSE
V	Sts_ModeCurrent	DINT	The current mode in which the machine is operating.	DINT #0
V	Sts_StateCurrent	DINT	The current state in which the machine is operating.	DINT #0
VAR_OUTPUT				

B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	TimeRollOverWarning	BOOL	A warning is sent when any of the time accumulators is approaching rolling over.
V	Sts_StateCurrentSec	DINT	Time (in seconds) spent in the current state.
V	Sts_StateCumulativeSec	StateCumulativeArray	An array containing the times spent operating in different modes and states.
V	Sts_ModeCurrentSec	DINT	Time (in seconds) spent in the current mode.
V	Sts_ModeCumulativeSec	DINT_Array32	An array of times spent in each mode.
V	Sts_AccTimeSinceReset	DINT	Accumulated time since Cmd_ResetAllTimes went high or the program was stopped for any reason.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

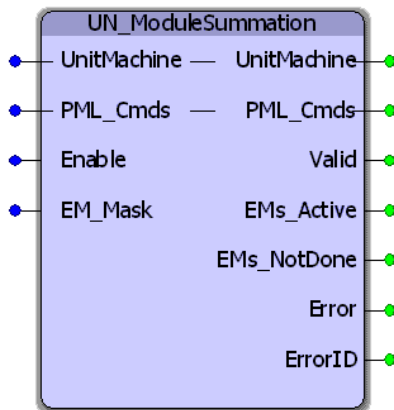
- See the PackML template documentation for further details on recommended usage.

Error Description

See the [Function Block ErrorID](#) list.



UN_ModuleSummation



The UN_ModuleSummation function block rolls up all the Equipment Module State Complete bits for active, enabled EMs. The result is an overall PMLs State Complete bit that is transferred to the PackML_StateControl function.

Library

Pack ML Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	UnitMachine	UNitMachine_ STRUCT	Structure containing all the information about the machines current state and mode of operation for all EMs and CMs	
V	PML_Cmds	PackML_Com- mands_ STRUCT	Structure that contains the current Unit mode of operation and the commands sent by PackML_StateMachine	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	EM_Mask	WORD	Mask to deactivate EMs. When an EM is deactivated, commands will not be sent down to the EM, for testing purposes. Each bit corresponds to the same number EM to deactivate. (Example: to deactivate EM_3, set EM_Mask.X3 =TRUE)	WORD#16#0000
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	

V	EMs_Active	WORD	The list of active EMs. Same bit scheme as EM_Mask. (Example: if EMs_Active.X4 = TRUE then EM_4 is active)
V	EMs_NotDone	WORD	A compilation of which Equipment Modules have not completed the transition task.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Error Description

See the [Function Block ErrorID](#) list.



Getting Started with PLCopen Toolbox

Requirements for v375

To use the PLCopen Toolbox, the project must also contain the following:

Firmware libraries:

- YMotion
 - Only required if using the [ReadAxisParameters](#) function block.

User libraries:

The following User Libraries must be listed above the PLCopen Toolbox and in the following order:

- DataTypes_Toolbox (v371 or higher)
- Math_Toolbox (v371 or higher)
 - Only required if using the [Jog_To_Position](#) or [ProductBuffer](#) function block.
- Yaskawa Toolbox (v371 or higher)
 - Only required if using the [Full_Closed_Control](#) or [Y_DigitalCamSwitch](#) function block.

Data Types

For versions prior to v205, this toolbox already includes the PLCTaskInfoTypes and MotionBlockTypes DataTypes files typically included when starting a new project, so delete them from your project to avoid compile errors that indicate duplicate DataType definitions. Starting in v205, the data type files that are included in the new project templates were added to a new User Library called DataTypes toolbox. You must include the DataTypes Toolbox in your project as mentioned above, and delete standard data type files from the main project to avoid compile errors that indicate duplicate DataType definitions. Starting with MotionWorks IEC v3.x, the new project templates will include the PLCopen and DataTypes Toolboxes, and the data types files will be excluded to save many steps during new project creation.

See the [PLCopen_Toolbox eLearning Modules](#) on Yaskawa's YouTube channel for video tutorials and examples.



PLCopen Revision History

Current Version:

2023-09-27 v375 released

- 1) HighSpeedOutput FB - Disabling and Re-enabling did not initialize properly. DCR 7036.

Previous Versions:

2022-05-09 v374 released

- 1) AxisControl - Add new VAR_OUTPUT AlarmShortDesc. DCR 5203.
- 2) Y_DigitalCamSwitch - Fixed string error when using External Encoder. DCR 5806.

2021-04-09 v372 released

- 1) AxisStatus FB - Added alarm code recognition for SGS7S. DCR 5104.

2020-11-04 v371 released

- 1) AbsolutePositionManager - Improve to use Flash file for retain data. This version fixes bugs introduced in 370 when the process was changed to use flash file instead of battery backed RAM. DCR 1784.

2020-02-06 v370 released

- 1) PLCopen Toolbox Datatype "Dependant" misspelling/wrong word. DCR 1117.
- 2) Cannot stop motor with Jog_To_Position function block (with specific vel and accel values). DCR 2638.
- 3) PLCopen_Toolbox_v350 Full_Closed_Control FB causes divide by zero error. DCR 2657.
- 4) Improve Feed_To_Length to account for additional marks in unexpected regions. DCR 2759.
- 5) Jog_To_Position giving an incorrect velocity graph for some profiles. DCR 2802.
- 6) AbsolutePositionManager has the wrong constant for a ControllerAlarm variable compare. DCR 2824.
- 7) Fix code change for Home_LS blocks to add FinishHoming (Done does not stay on). DCR 2873.
- 8) Home_* function blocks should not output Done if Error or CommandAborted. DCR 2975.
- 9) AbsolutePositionManager FB - Improve to use a Flash file for retain data (for Sigma7-Siec support.) DCR 1784.

2019-02-11 v352 released

- 1) SetAccDecPns - Function block use by the Home_LS* functions had issues fixed with writing Pn 305, pn 306. DCR 1405.
- 2) Home_LS* blocks - Fixed to set MotionState back to "StandStill" by executing MC_FinishHoming. DCR 1530.
- 3) HighSpeedOutput - Fixed double counting of the output setting. (Status output.) DCR 1963.

2018-03-08 v350 released

- 1) Jog - Added direction change detection. DCR 1483 *)
- 2) ProductBuffer - Fixed new ErrorID added in v340 which detects a duplicate latch caused by a fw bug which was fixed in fw 3.4.0. ProductBuffer was errantly indicating that condition on first latch if less then Lockout distance. DCR 1449.
- 3) SetAccDecLimits - Fixed bug related to negative deceleration settings. DCR 1569.
- 4) HighSpeedOutput - v351 in Toolbox Installer from August 2018 includes this bug fix. The OutputCount could increment twice for each time the output was set.

2017-11-22 v341 released (Filename is still 340, using new project version control)

This version is included in the MotionWorks IEC 3.4.0.233a

- 1) AxisControl - Reverted change to only show outputs when Enable is High in v340. The v340 behavior was not ideal. DCR 918.
- 2) SetAccDecPns - This was a new block added for v340 to enhance the capabilities of Home_LS_Pulse and Home_LS, but there was a compile error due to an incorrectly declared variable, now fixed. DCR 1318.

2017-08-14 v340 released

- 1) Home_LS_Pulse, Home_LS - Improved homing FBs by calculating required Pn305 / 306 / 80E for acc dec while in velocity mode. DCR 490.
- 2) Home_TouchProbe - HomeData.HomeDirection input was ignored. DCR 826.
- 3) AxisControl - Changed to only show outputs when Enable is High. DCR 918.
- 4) AbsolutePositionManager - Improved PositionValid output for the case when the user resets the Absolute Encoder even if there is not an alarm. DCR 949.
- 5) Y_DigitalCamSwitch - Upgraded functionality to support FT62 ServoPacks. DCR 1035.
- 6) Full_Closed_Loop - Fixed calculation for MaxCorrectionPerScan. DCR 1050.
- 7) UpdateUsePointer - updated to also update MultiUsePointer. DCR 1105.
- 8) Feed_To_Length - Improved operation so LatchPosition does not toggle to zero between moves. DCR 1167.
- 9) FeedRegistrationCheck - Added ErrorID to if negative latch distance is reported. DCR 1175.
- 10) Feed_To_Length - Improved operation in cases where latch comes very late in the move and the tuning is soft. DCR 1183.
- 11) ProductBuffer - Added ErrorID to catch firmware bug (SCR 10736) if previous latch is reported. DCR 1185.
- 12) HomeTouchProbe - Now executes MC_AbortTrigger if there is an Error. DCR 1187.

2016-10-31 v330 released

- 1) ReadMotorSpeed - New FB added. - Known issue: This function block only works for Sigma 5 motors.

2016-06-21 v302 released

- 1) Feed_To_Length - Added Active Output. DCR 695.
- 2) AbsoluteEncoderManager - Added Busy Output. DCR 830.

3) ProductBuffer - Added new check and ErrorID 10099 if the Axis type is not a servo or external encoder, and TestMode is not set TRUE. (This combination is not supported. Also suppressed an Error generated by internal function MC_AbortTrigger when an unsupported axis is selected while in TestMode. DCR 903.

4) Full_Closed_Control - New Function block added for applications that require full closed control but cannot use the Full Closed option card on the ServoPack, either because its an MP2600iec or because the safety option card is installed. DCR 930.

2015-08-19 v301 created

1) ProductBuffer - Updated to support M-III simultaneous latch (Two instances of ProductBuffer FB for the same axis. This function now uses new parameter 1034 to obtain the unmodularized latch for TRIGGER_REF.ID=1. Compatibility with older firmware on MPiec controllers using ID=1 for the first is maintained.

2015-01-31 v300 created

1) Identical to v207, but recompiled specifically for MotionWorks IEC v3.x.

2014-12-11 v207 released - developed using firmware 2.7.0)

1) Toolbox_DataTypes - Added new data types under ProductBufferStruct to support multiple latch patterns.

2) ProductBuffer - Added code to support multiple latch patterns, such as rising edge and falling edge .

3) ProductBuffer - Added code to automatically detect External Encoder type, even on MP3000iec series .

4) AxisInterLock - Replaced prm 1005 (actual position cyclic) with 1006 (actual position non cyclic.)

5) ReadMotorSpeed - New function block to read peak and rated speeds of connected rotary servos .

6) FeedToLength - Improved output logic - NOT Done added to RETURN rung. Enable added to Done rung.

7) ReadMotorSpeed - New function added to read the Motor part number string to report the Rated and Maximum speeds.

2014-03-03 v206 released - developed using firmware 2.5.0

1) UpdateUsePointer - New function block. This code can be used to update the ProductBufferStruct's UsePointer which is used in the ProductBuffer function block.

2) ProductBuffer - Added a Busy output because it takes 3 scans after Enable goes FALSE before iActive goes FALSE. ProductBuffer.Busy can be used as an interlock if the ProductBuffer FB is used within other user function blocks to prevent the code from going dormant before the block is finished.

3) HomeTouchProbe - New function block. Homes an axis based on a sensor wired to the high speed latch input on the ServoPack.

4) VelocityLimits - Fixed cut & paste typos in code.

5) MA_2Stage_Calc - New supporting function block used by MoveAbsolute_2Stage.

6) MoveAbsolute_2Stage - New function block. Provides two acceleration and two deceleration values switchable at a specified speed.

7) Jog_To_Position - Fixed typo with Accel variable causing complete inoperability in v205.

8) MoveRelative_ByTime - Added Acceleration and Deceleration inputs.

9) GetActionPointers - New function block. For use with user functions that use the ProductBuffer for multi operation support.

10) ProductBufferStruct - Multiple pointers for several actions are now supported.

11) Jog - Improved logic to account for continuously changing inputs without getting stuck.

12) Home_LS - Added NC contact LimitError to workaround issue described in SCR 8025.

13) Home_LS - Added NC contact LimitError to workaround issue described in SCR 8025.

14) HighSpeedOutput - Added iActive contact to various rungs to clear outputs from previous execution.

15) HighSpeedOutput - Changed the DIV order in rung 4. Converted DINT_TO_LREAL so that the variable factor can take a non integer value. This correction allows HighSpeedOutput FB functionality with the MinTime input.

2013-09-01 v205 released - developed using firmware 2.5.0

1) Removed references to Math Toolbox functions where possible. Only the ProductBuffer function block still requires the Math Toolbox.

2) Because of the reintroduction of functions with EN/ENO, the MP2600 requires firmware 2.1.

3) Moved all datatype definitions for firmware libraries to a new DataTypes Toolbox. Upgrading to PLCopen v205 will require deleting any Yaskawa firmware datatypes files and adding the DataTypes Toolbox.

4) JogToPosition - Fixed method in which a change of speed is detected to refire MC_MoveVelocity.

2013-03-15 v204 released - developed using firmware 2.4.0

1) ProductBuffer - Swapped position of RegistrationData and ProductAxis to conform to VAR_IN_OUT convention.

2) AccDecLimits - Fixed several copy / paste errors and variable naming confusion.

3) AbsoluteEncoderManager - Verified operation using Signa-II 2 digit alarm formats.

2012-10-29 v203 released - developed using firmware 2.4.0

1) AbsoluteEncoderManager - Removed the 'Active' contact from rung 5 to clear alarms that have been reset.

2) ReadAxisParameters - Added 14 parameters. (Mainly limit parameters)

3) Jog_To_Position - Improved deceleration ramp.

4) Feed_To_Length - Added. This function will index a default amount, and update the final target based on a registration input.

2012-06-29 v202 released - developed using firmware 2.2.1

1) ReadAxisParameters - Added the following parameters FilterCmdVelocity 1021, CmdAcceleration 1022, and postFilterCmdTorque 1024.

2) PLCTaskInfoTypes - Added DataTypes to mirror the 2.0 additions for high resolution task timing.

3) AbsolutePositionManager - Added additional alarm detection to catch A830, A840, and ACC0 alarms. Also added code to clear EncoderAlarmID and ControllerAlarmID when the block goes inactive.

4) Jog_To_Position - Added. For rotary applications that must stop at a specific location.

5) HighSpeedOutput - Fixed issue with MinTime. Was not working correctly if Min Time not zero. (YEU)

2011-12-08 v201 released - developed using firmware 2.0.0

1) ProductBuffer - Added two optional inputs to allow FB to operate in a test or simulation mode.

2) ReadAxisParameters - Disabled reading parameter 1311 because it causes an error on MP2600iec. This parameter is scheduled to return a zero instead of an ErrorID in firmware 2.2.

3) ReadAxisParameters - Fixed two swapped values CamOffset and CamScale were swapped in v200.

2011-07-29 v200 released - developed using firmware 2.0.0

Built from v022beta

ReadAxisParameters - Upgraded to use the new Y_ReadMultipleParameters firmware function block.

2011-02-24 v022beta created - developed using firmware 2.0.0

- 1) Home_Init - Added for users who prefer to avoid structured text POU for initializing the HomeStruct
- 2) Math Toolbox - Upgraded to v004 with Enable / Valid as function block I/O for compatibility with FW 2.1*)
- 3) Changed AxisControl to allow clearing a drive warning while the servo is enabled.

2011-01-24 v021 released - developed using firmware 1.2.3

- 1) HighSpeedOutput - Added. For simplified operation with the external encoder high speed output.
- 2) Home_LS_Pulse - Added a MC_MoveRelative between searching for the limit switch and C channel to prevent ErrorID 4397 from occurring: "Over travel limit still ON after attempting to move away from it."
- 3) Axes_Interlock - Enhanced to work with axes configured for rotary mode.

2010-10-04 v020 released. developed using firmware 1.2.2.9

- 1) Jog - Rewrote function to follow the 'Enable' template standard created for ST functions.
- 2) ProductBuffer - Improved lockout operation when a manual offset was applied. See ProductBuffer FB comments for more details.
- 3) Jog - Improved Done output (It will only pulse; this block is a special case of Enable type)
- 4) AxisParams Struct - Added CamTableCumulativeOutput
- 5) Home_LS - Fixed rung 6 (incorrect execute bit), duplicated StartOffset from rung 5.
- 6) DigitalCamSwitch - Added. See the initialize POU for example data setup.
- 7) ReadAxisParameters - Added LoadType and MachineCycle parameters.
- 8) AbsolutePositionManager - Added. For confirmation that the absolute position was set and valid
- 9) Moved Math functions to Math Toolbox

2010-02-03 v019 released

- 1) CamGenerator - Added.
- 2) CamSlaveFeedToLength - Removed MC_AbortTrigger.
- 3) Fixed Missed Latch counter (not initialized properly)
- 4) Added CamMaster_Lookup, and SlaveIndex_Lookup
- 5) Added MissedLatch and LatchPosition outputs to CamSlave_FeedToLength
- 6) Improved ProductBuffer FB to account for external encoder master (prm 1016 / 1006 switch)
- 7) Added CamBlend function block
- 8) Added WindowCheck function block
- 9) CamGenerator formula type 4 (Cycloidal) changed to 3 (Simple harmonic). It was incorrectly identified.*)
- 10) Added ParamTypes input to ReadAxisParameters to increase efficiency of the function (Provides selective parameter reads by group.)

- 11) MOVED ALL CAMMING SUPPORT FUNCTIONS TO CAM TOOLBOX - FOR PRO VERSION ONLY.
- 12) The "PLCTaskInfoTypes" DataType file was removed from this Toolbox. If you need to replace it in your project, open a second copy of MotionWorks IEC, and open a project that already has the PLCTaskInfoTypes DataType file, then copy & paste it into your project explorer.

2009-10-27 v018 released ***)**

- 1) Added SensorWindow input to CamSlave FeedToLength
- 2) Added PositionLimits, VelocityLimits, and AccDecLimits function blocks
- 3) Removed Enable Servo FB, use AxisControl FB
- 4) Removed the variable Speed from HomeStruct, it was not used for anything.
- 5) Converted Home blocks removed all Set or RESET coils.
- 6) Added MOVE_UNIT & MOVE_LREAL function block to provide compatibility with MP2600iec.
- 7) AxesInterlock does not support rotary mode axes.
- 8) ReadAxisParameters changed to increase efficiency.
- 9) Added some outputs such as 'Valid' to some blocks for increased consistency with PLCopen.
- 10) First version formalized with help documentation.

2009-07-15 v017 released

- 1) Created Home_Pulse, Homes to C Channel, performs moves offset and defines position.
- 2) Removed R_TRIGs from the ErrorID portion of Home_LS, Home_LS_Pulse, and Home_Pulse because it was preventing the blocks from showing errors.
- 3) Updated ProductBuffer function block for both modularized and non modularized latch data.
- 4) Updated ReadAxisParameters to include VAR_IN_OUT (for speed) and additional input parameter to specify axis type. Also reduced parameter set to eliminate those that typically do not change.
- 5) Added MC_Status data.
- 6) Improved interlock logic in Home_LS_Pulse, Home_LS, Home_Pulse functions, added CommandAborted as output, and fixed a typo in all three blocks where the variable attached to the Busy output of one of the internal blocks was referencing an error bit.

2009-05-28 v016 released

- 1) Y_AdjustMode in the DataTypes file was incorrectly named Y_AdjustMethod.
- 2) Added NOT(Busy) to the Execute of MC_TouchProbe in CamSlave_FeedToLength. New Error code in firmware 1.1.2.5 caused new problem if the block was executed when already executing. This may occur if there is bounce on the input sensor.
- 3) Fixed MoveRelative_ByTime - calculations would cause error if negative distance. Also added checks for negative time (causes error) and zero distance (No Error)

2009-05-07 v015 released

- 1) Added interlock to Jog's MC_MoveVelocity to prevent rising edge of exe if Stop is busy to prevent ErrorID 4370 from appearing.
- 2) Added Axes_Interlock function.

2009-04-16 v014 released

- (* Fixed AxisControl and Enable Servo to allow a re attempt to enable servo if MC_Power has Error. *)
- (* Previously they had a normally closed contact from the MC_Power FB preventing the block from enabling *)
- (* again. Also changed these two blocks to reset Error & ErrorID outputs when Enable=FALSE *)
- (* Changed the Jog Block Error and ErrorID outputs to only come on if JogFwd or JogRev is On *)
- (* Added CommandAborted to the Busy interlock circuits of Home_LS_Pulse and Home_LS. *)

2009-03-30 v013 released

Released version of v012.

- 1) Explicitly set some parameters in ReadAxisParameters to LREAL#0.0 and documented as being unavailable. because they were causing Access Violation Errors when viewed in the Watch Window.

2009-01-27 v012 created

- 1) This version was released to a few people as a work in progress.
- 2) PLCopenPlus-v_2_2 firmware library used and included with this version.
- 3) Added LatchPositionNonCyclic to the AxisParameterStruct structure for ReadAxisParameters FB.
- 4) Corrected naming of Cam parameters 1500, 1501, 1502.
- 5) Corrected AxisStatus FB, Drive Warnings and Errors were backwards.
- 6) Changed AxisControl.ControlAlarmID And AxisStaus.ControlAlarmID to a 32 bit UDINT output.
- 7) Jog converted to PLCopen convention (outputs) and code converted to ST.
- 8) Added CamSlave_FeedToLength, which uses MC_TouchProbe, SlaveRegistrationCheck, and Y_SlaveOffset.

2009-01-27 v011 released

- 1) PLCopenPlus-v_2_2 firmware library used and included with this version.
 - 2) Added AxisStruct STRUCT
- (* Fixes *)
- 3) Simplified MoveRelativeByTime function, removed additional interlocks, and just copied MC_MoveRelative outputs to MoveRelativeByTime outputs. *)
 - 4) Made corrections to the AxisParameterArray, added cam parameters. NOTE: will require controller firmware 1.1.0.4 or greater to read some of the cam parameters. Set the READ flag for those parameters to FALSE if you are using older firmware.

2009-01-12 v010 released

- 1) PLCopenPlus-v_2_1 firmware library used and included with this version.
- 2) Changed interface of homing blocks to use HomeStruct. Makes FB smaller and quicker to enter home data.
- 3) Added example initialization code as a Program POU to enable cut & paste to speed development.
- 4) Open the Toolbox as a project in a second copy of MotionWorks IEC as a project to see the Initialization POU.

5) Added 'ControllerAlarm' function block to provide BOOL output when there is a controller alarm.

(Uses Y_ReadAlarm and compares the AlarmID for non zero.

6) Added Homed BOOL to HomeStruct.

2008-11-05 v009 released

1) Completed and tested the MoveRelative_ByTime function.

2) Previous versions would not allow the block to run more than once.

2008-10-17 v008 released

1) In Home_LS_Pulse and Home_LS, added Reset Coil for Homing Done at the last rung.

2008-10-10 v007 released

1) Added BOOL outputs to AxisControl (DriveAlarm, DriveWarning)

2) Fixed DriveWarningID and DriveAlarmID, they were backwards.

2008-10-02 v005 released

Added Functions:

1) AxisControl

2) AxisStatus

Fixes:

3) Changed errant F_TRIG functions used in Home_LS_Pulse for ErrorID to R_TRIG.

2008-09-22 v004 released

Changes:

1) EnableServo, upgraded to include ErrorClass output from MC_ReadAxisError from PLCopen.

2) FIRMWARE library 1.0.4.5 and PLCopenPlus-v_2_1

3) Includes structures for axis parameters and homing functions

Not complete:

4) MoveRelative_ByTime

2008-08-29 v003 released

Added Functions:

1) Home_LS_Pulse

2) Home_LS

3) ReadAxisParameters

Not complete:

4) MoveRelative_ByTime

5) NOTE: v0035 supplied with the MP2300Siec_Sales_Demo_v001

2008-05-20 v002 released

Includes:

- 1) EnableServo
- 2) Jog

Not complete:

- 3) MoveRelative_ByTime



Data Type: AXIS_REF

The AXIS_REF data type identifies an axis and thus provides the interface to the hardware or virtual axes. AXIS_REF is referenced as a VAR_IN_OUT in all function blocks. It is represented as an input and an output connected by a horizontal line in the graphical representation of a function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAxisRef	AXIS_REF		
U	AxisNum	UINT	The value of AxisNum is determined by the logical axis number assigned by the Hardware Configuration. See the Configuration tab under each axis.	MyAxisRef.AxisNum

Example

```
MyServo.AxisNum:=UINT#3;
```



Data Type: AxisParamData

Supporting structure for [AxisPrmArray](#). Used by the [ReadAxisParameters](#) function block.

Data Type Declaration

TYPE

AxisParamData:ARRAY[0..60] OF IndividualParamDetails;

END_TYPE



Data Type: AxisParameterStruct

For use with the [CamSlave_FeedToLength](#) and [CamSlave_WindowCheck](#) function blocks.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAxisParameterStruct	AxisParameterStruct		
C	ActualPosition	LREAL	1000	MyAxisParameterStruct.ActualPosition
C	ActualPositionCyclic	LREAL	1005	MyAxisParameterStruct.ActualPositionCyclic
C	ActualPositionNonCyclic	LREAL	1006	MyAxisParameterStruct.ActualPositionNonCyclic
C	ActualTorque	LREAL	1004	MyAxisParameterStruct.ActualTorque
C	ActualVelocity	LREAL	1001	MyAxisParameterStruct.ActualVelocity
C	AtVelocity	BOOL	1141	MyAxisParameterStruct.AtVelocity
C	BufferedMotionBlocks	LREAL	1600	MyAxisParameterStruct.BufferedMotionBlocks
C	CamMasterCycle	LREAL	1512	MyAxisParameterStruct.CamMasterCycle
C	CamMasterPosition	LREAL	1500	MyAxisParameterStruct.CamMasterPosition
C	CamMasterShiftedCyclic	LREAL	1502	MyAxisParameterStruct.CamMasterShiftedCyclic
C	CamMasterShiftedPosition	LREAL	1501	MyAxisParameterStruct.CamMasterShiftedPosition
C	CamMasterScale	LREAL	1510	MyAxisParameterStruct.CamMasterScale
C	CamMasterShift	LREAL	1511	MyAxisParameterStruct.CamMasterShift
C	CamOffset	LREAL	1531	MyAxisParameterStruct.CamOffset
C	CamScale	LREAL	1530	MyAxisParameterStruct.CamScale
C	CamShiftRemaining	LREAL	1513	MyAxisParameterStruct.CamShiftRemaining
C	CamState	LREAL	1540	MyAxisParameterStruct.CamState
C	CamTableIDEngaged	LREAL	1541	MyAxisParameterStruct.CamTableIDEngaged
C	CamTableOutput	LREAL	1520	MyAxisParameterStruct.CamTableOutput
C	CommandedAcceleration	LREAL	1012	MyAxisParameterStruct.CommandedAcceleration
C	CommandedPosition	LREAL	1010	MyAxisParameterStruct.CommandedPosition
C	CommandedPositionCyclic	LREAL	1015	MyAxisParameterStruct.CommandedPositionCyclic
C	CommandedPositionNonCyclic	LREAL	1016	MyAxisParameterStruct.CommandedPositionNonCyclic
C	CommandedTorque	LREAL	1014	MyAxisParameterStruct.CommandedTorque
C	CommandedVelocity	LREAL	1011	MyAxisParameterStruct.CommandedVelocity
C	InPosition	BOOL	1140	MyAxisParameterStruct.InPosition
C	LatchPositionNonCyclic	LREAL	1031	MyAxisParameterStruct.LatchPositionNonCyclic
C	PositionError	LREAL	1130	MyAxisParameterStruct.PositionError
C	PositionWindow	LREAL	1120	MyAxisParameterStruct.PositionWindow



Data Type: AxisPrmArray

Used by the [ReadAxisParameters](#) function block.

Data Type Declaration

TYPE

AxisPrmArray: STRUCT

Param: [AxisParamData](#);

END_STRUCT;

END_TYPE



Data Type: AxisStruct

For use as a container for all axis related data. (Customizable)

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAxisStruct	AxisStruct		
U	Ref	AXIS_REF	Used with the Axis VAR_IN_OUT of PLCopen function blocks.	MyAxisStruct.Ref
U	JogSpeed	LREAL	In user units/sec as defined in the Hardware Configuration.	MyAxisStruct.JogSpeed
U	RunSpeed	LREAL	In user units/sec as defined in the Hardware Configuration.	MyAxisStruct.RunSpeed
U	Position	LREAL	In user units as defined in the Hardware Configuration.	MyAxisStruct.Position
U	Acceleration	LREAL	In user units/sec ² as defined in the Hardware Configuration.	MyAxisStruct.Acceleration
U	Deceleration	LREAL	In user units/sec ² as defined in the Hardware Configuration.	MyAxisStruct.Deceleration
U	Jerk	LREAL	In user units/sec ³ as defined in the Hardware Configuration.	MyAxisStruct.Jerk
U	Status	BOOL	To indicate if the axis is enabled.	MyAxisStruct.Status
U	Warning	BOOL	Indicates if the axis has a warning, typically an alarm code beginning with 9 such as A.910 on Sigma series ServoPacks.	MyAxisStruct.Warning
U	Alarm	BOOL	Indicates if the axis has an alarm, which may originate in either the controller or the drive.	MyAxisStruct.Alarm
U	DriveAlarmID	UINT	Indicates the drives alarm ID, typically equivalent to the alarm displayed on the front display of the drive if viewed in hex.	MyAxisStruct.DriveAlarmID
U	DriveWarningID	UINT	Indicates the drives warning ID, typically equivalent to the alarm displayed on the front display of the drive if viewed in hex.	MyAxisStruct.DriveWarningID
U	ControlAlarmID	UDINT	Indicates the controllers alarm ID, equivalent to the alarm displayed in the web server if viewed in hex.	MyAxisStruct.ControlAlarmID
U	Prm	AxisParameterStruct		MyAxisStruct.Prm
U	Home	HomeStruct		MyAxisStruct.Home
U	Latch	RegistrationStruct		MyAxisStruct.Latch
U	Cam	CamStruct		MyAxisStruct.Cam



Data Type: BufferPatternArray

Supporting structure for [ProductBufferStruct](#). Used by the [ProductBuffer](#) function block.

Data Type Declaration

TYPE

BufferPatternArray: ARRAY[0..9] OF TRIGGER_REF;

END_TYPE



Data Type: CAMSWITCH_ARRAY

Supporting structure for [CAMSWITCH_REF](#). Used by the [Y_DigitalCamSwitch](#) function block.

Data Type Declaration

TYPE

CAMSWITCH_ARRAY: ARRAY[0..255] OF [CAMSWITCH_STRUCT](#);

END_TYPE



Data Type: CAMSWITCH_REF

Used by the [Y_DigitalCamSwitch](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCAMSWITCH_REF	CAMSWITCH_REF		
U	MasterType	INT	0 = Infinite/Rotary, 1 = Finite/Linear	MyCAMSWITCH_REF.MasterType
U	MachineCycle	LREAL	This number should match the setting in the Hardware Configuration. Valid for Type = 0.	MyCAMSWITCH_REF.MachineCycle
U	LastSwitch	INT	To limit the evaluation of the array	MyCAMSWITCH_REF.LastSwitch
U	Switch	CAMSWITCH_ARRAY		MyCAMSWITCH_REF.Switch[0]...



Data Type: CAMSWITCH_STRUCT

Supporting structure for [CAMSWITCH_ARRAY](#). Used by the [Y_DigitalCamSwitch](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyCAMSWITCH_STRUCT	CAMSWITCH_STRUCT		
U	TrackNumber	INT	A reference to the track number to which this switch is to be applied. The PLS block will support up to 32 tracks. There is no limit to how many switches can be assigned to a single track except for the maximum of 256 switches.	MyCAMSWITCH_STRUCT.TrackNumber
U	FirstOnPosition	LREAL	Lower boundary where the switch is ON.	MyCAMSWITCH_STRUCT.FirstOnPosition
U	LastOnPosition	LREAL	Upper boundary where the switch is ON. If LastOnPosition < FirstOnPosition, then the switch should be OFF between the positions (inverse cam switch)	MyCAMSWITCH_STRUCT.LastOnPosition
U	AxisDirection	INT	The direction of the master for which this switch applies. 0 = Both Pos and Neg; 1 = Positive Only (future); 2 = Negative Only (future). ONLY 0 should be implemented at this time.	MyCAMSWITCH_STRUCT.AxisDirection
U	CamSwitchMode	INT	Position vs Time-Based output. 0 = Position. 1 = Time.	MyCAMSWITCH_STRUCT.CamSwitchMode
U	Duration	DINT	The duration of the switch. If CamSwitchMode = 0 (Position) AND Duration <> 0.0, this Duration will serve as a Maximum ON time for the switch. A setting of 0.0 means infinite time. If CamSwitchMode = 1 (Time), this duration will serve as the ON time of the switch once FirstOnPosition has been reached. A setting of 0.0 will result in a block error.	MyCAMSWITCH_STRUCT.Duration



Data Type: HomeStruct

For use with all HOME_*** function blocks.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyHomeStruct	HomeStruct		
U	Direction	INT	Specified the initial direction of homing. Refer to MC_Direction enumerated types.	MyHomeStruct.Direction
U	SwitchMode	INT	Configuration for action of the home sensor. [See MC_SwitchMode]	MyHomeStruct.SwitchMode
U	TorqueLimit	LREAL	Default if zero or unconnected is 100.00% of rated torque.	MyHomeStruct.TorqueLimit
U	ApproachVelocity	LREAL	Velocity at which the axis will travel in search of the first switch, usually a limit switch.	MyHomeStruct.ApproachVelocity
U	ApproachTimeLimit	LREAL	In seconds.	MyHomeStruct.ApproachTimeLimit
U	ApproachDistanceLimit	LREAL	The maximum distance the axis will travel in search of the first switch (typically limit switch) before aborting the homing operation and issuing an error.	MyHomeStruct.ApproachDistanceLimit
U	AccDec	LREAL	The acceleration and deceleration which will be applied to all homing moves.	MyHomeStruct.AccDec
U	CreepVelocity	LREAL	Velocity at which the axis will travel in search of the C channel on the encoder.	MyHomeStruct.CreepVelocity
U	CreepTimeLimit	LREAL	In seconds.	MyHomeStruct.CreepTimeLimit
U	CreepDistanceLimit	LREAL	The maximum distance the axis will travel in search of the C channel before aborting the homing operation and issuing an error.	MyHomeStruct.CreepDistanceLimit
U	Offset	LREAL	Position offset to MOVE after finding the last input device (switch or C channel, based on the function block being used.)	MyHomeStruct.Offset
U	OffsetVelocity	LREAL	Velocity at which the axis will travel during the home offset move.	MyHomeStruct.OffsetVelocity
U	Position	LREAL	This is the position that will be defined when all homing actions are complete, including the offset move.	MyHomeStruct.Position
U	Homed	BOOL	Flag to indicate that the axis was successfully homed. Not used by the homing function blocks, reserved for use by the application.	MyHomeStruct.Homed



Data Type: LatchBufferArray

Supporting structure for [ProductBufferStruct](#) Used by the [ReadAxisParameters](#) function block.

Data Type Declaration

TYPE

LatchBufferArray: ARRAY[0..100] OF LREAL;

END_TYPE



Data Type: MoveStruct

For use with MC_MoveAbsolute, MC_MoveRelative, and MC_MoveVelocity.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyMoveStruct	MoveStruct		
U	Position	LREAL	In user units as defined in the Hardware Configuration.	MyMoveStruct.Position
U	Velocity	LREAL	In user units/sec as defined in the Hardware Configuration.	MyMoveStruct.Velocity
U	Acceleration	LREAL	In user units/sec ² as defined in the Hardware Configuration.	MyMoveStruct.Acceleration
U	Deceleration	LREAL	In user units/sec ² as defined in the Hardware Configuration.	MyMoveStruct.Deceleration
U	Jerk	LREAL	In user units/sec ³ as defined in the Hardware Configuration.	MyMoveStruct.Jerk



Data Type: MultiUseData

Supporting structure for [ProductBufferStruct](#). Used by the [ProductBuffer](#) function block.

Data Type Declaration

TYPE

MultiUseData:ARRAY[0..9] OF MultiUsePointers;

END_TYPE



Data Type: MultiUsePointers

Supporting structure for [ProductBufferStruct](#). Used by the [ProductBuffer](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAxisStruct	AxisStruct		
U	UsePointer	INT	For applications that require several operations to be performed in sequence based on the same registration mark. This feature could be used when position data captured on the master axis must be shared by multiple axes or multiple actions on one axis.	MyProductBufferStruct.Multi[x].UsePointer
U	DependentAction	INT	Specify the previous action that, when complete, the next action can be processed.	MyProductBufferStruct.Multi[x].DependentAction



Data Type: PatternAwayDistanceArray

Supporting structure for [ProductBufferStruct](#). Used by the [ProductBuffer](#) function block.

Data Type Declaration

TYPE

PatternAwayDistanceArray: ARRAY[0..9] OF LREAL;

END_TYPE



Data Type: PatternPointerArray

Supporting structure for [ProductBufferStruct](#). Used by the [ProductBuffer](#) function block.

Data Type Declaration

TYPE

PatternPointerArray: ARRAY[0..100] OF UINT;

END_TYPE



Data Type: ProductBufferStruct

For use with the [ProductBuffer](#) and [Labeler](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
---	---------	-----------	-------------	-------

U	BufferSize	INT	Maximum number of registration marks to be tracked. (Circular buffer size). Allow enough extra capacity in the circular buffer. A larger size does not impact performance.	MyProductBufferStruct.BufferSize
C	BufferNonCyclic	LatchBufferArray	Array (circular buffer) of all recorded registration marks (unmodularized latch values).	MyProductBufferStruct.BufferNonCyclic[x]
C	BufferCyclic	LatchBufferArray	Array (circular buffer) of all recorded registration marks (modularized latch values).	MyProductBufferStruct.BufferCyclic[x]
U	Sensor	TRIGGER_REF	TRIGGER_REF for the axis which registration marks are to be detected. See TRIGGER_REF data type description in the PLCopen help.	MyProductBufferStruct.Sensor.Bit
U	SensorDistance	LREAL	Distance in units of the master axis from the registration sensor to the required synchronization point with a slave axis.	MyProductBufferStruct.SensorDistance
U	SensorOffset	LREAL	If the sensor is an exact multiple of machine cycles from cut position, this number would be zero. If for example the sensor was 3.5 machine cycles away from the synchronization point, then this value would be 1/2 of the machine cycle.	MyProductBufferStruct.SensorOffset
U	ManualOffset	LREAL	Amount to adjust the synchronization point, typically comes from an HMI.	MyProductBufferStruct.ManualOffset
U	LockoutDistance	LREAL	Distance after recording a latch that another latch must be ignored.	MyProductBufferStruct.LockoutDistance
U	ProductAwayDistance	LREAL	The distance the product travels from its initial detection until it is safely past the slave operation to slave processing the next product.	MyProductBufferStruct.ProductAwayDistance
C	StorePointer	INT	Array index of the latch data that was last stored by MC_TouchProbe.	MyProductBufferStruct.StorePointer
U	UsePointer	INT	Array index of the latch data to be used by the process.	MyProductBufferStruct.UsePointer
U	PrevUsePointer	INT	Array Index of the previously used latch data.	MyProductBufferStruct.PrevUsePointer

U	Multi	MultiUseData	Array of pointers for applications which process multiple operations based on a single registration mark. Examples of such applications are pick and place, score and punch etc.	MyProductBufferStruct.Multi[x].UsePointer
U	LastAction	INT	Total number of actions to be performed based on a single registration mark. If the default value of zero is used, non "Multi" operation is performed using the basic StorePointer and UsePointer method.	MyProductBufferStruct.LastAction
U	BufferPatternSize	INT	Size of the repeating pattern of TRIGGER_REFS. This value is only required if a pattern of latches must be captured. For example, if ServoPack signals EXT1 and EXT2 capture a products leading and trailing edge in a repeating sequence, set BufferPatternSize: =2. Set to default BufferPatternSize: =0 if only one TRIGGER_REF is required for the application.	MyProductBufferStruct.BufferPatternSize
U	BufferPattern	BufferPatternArray	Array of TRIGGER_REFS to be specified by the user. This provides support for capturing a rising and falling edge of the same sensor, such as for product length measurement.	MyProductBufferStruct.BufferPattern[x].Bit
U	PatternAwayDistance	PatternAwayDistanceArray	Array of product away distances corresponding to each TRIGGER_REF.	MyProductBufferStruct.PatternAwayDistance[x]
C	BufferedPattern	PatternPointerArray	Array of TRIGGER_REFS corresponding to recorded registration latches. This reports the type of signal that was captured (as configured by BufferPattern) for each corresponding element in the LatchBufferArrays.	MyProductBufferStruct.BufferedPattern[x]

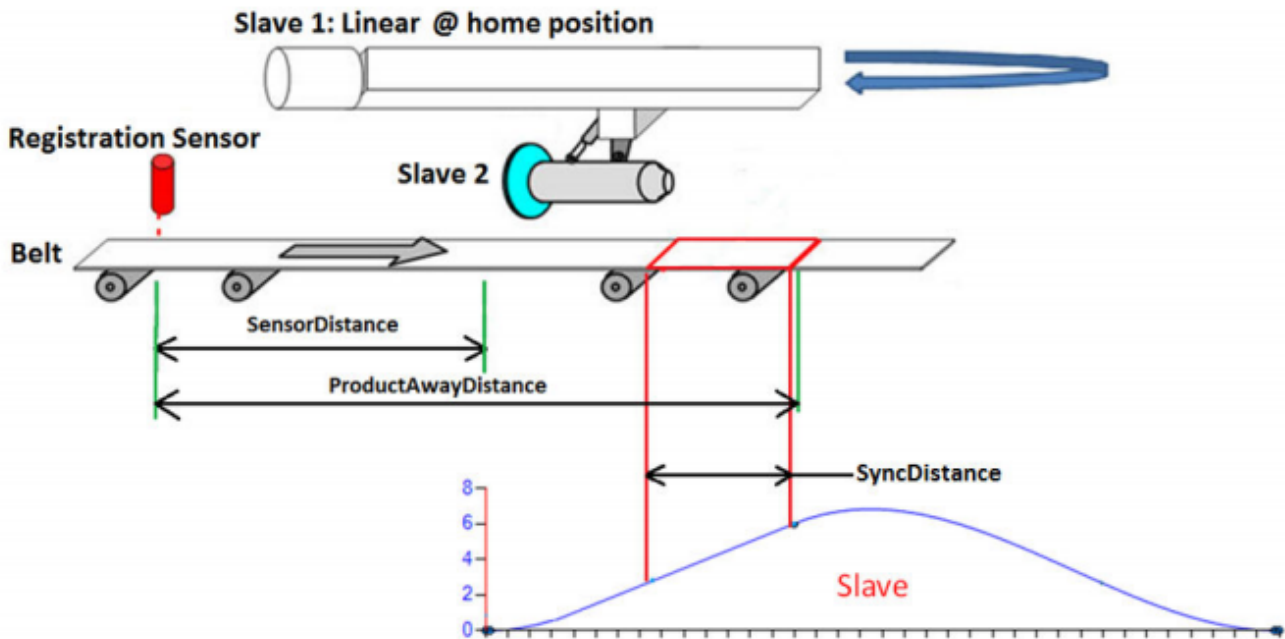
Notes:

The following structure values are not used by the ProductBuffer function block, but are included because typical applications that can benefit from this function require this data for successful operation:

- SensorDistance
- SensorOffset
- ProductAwayDistance

Example 1:

Example of ProductBufferStruct being initialized for a linear flying shear application



(*Data for Registration based Linear Flying Shear*)

(*ProductBufferStruct for Registration Data *)

(*=====*)

Products.BufferSize := INT#20; (* Maximum size of buffer*)

Products.LockoutDistance := LREAL#9.0; (* Looks for a new part only after conveyor has traveled LockOutDistance after previous part*)

Products.SensorDistance := LREAL#14.075; (* Distance from sensor to start of slave 1 home position (beginning of cam profile) *)

Products.ProductAwayDistance := LREAL#20.075; (* Distance from sensor to end of sync position in the cam table.

This is used to update the use pointer. Cam disengages only when use pointer = store pointer *)

Products.Sensor.Bit:=UINT#1; (* Equates to EXT1 on a Sigma-5 amplifier, see MC_TouchProbe help for details *)

Example 2:

Example of ProductBufferStruct being initialized for a linear flying shear application where rising and falling edges of a product are being captured.

(*ProductBufferStruct for Registration Data *)

(*=====*)

Products.BufferSize :=INT#20; (* Maximum size of Buffer*)

Products.LockoutDistance := LREAL#90.0; (* Looks for a new part only after conveyor has traveled LockOutDistance after previous part*)

Products.SensorDistance := LREAL#1580.0; (* Distance from sensor to start of slave 1 home position (beginning of cam profile) *)

Products.BufferPattern[0].Bit :=UINT#1; (* PART SENSOR Signal - RISING EDGE connected to Latch DI_01*)

Products.BufferPattern[1].Bit :=UINT#0; (* PART SENSOR Signal - FALLING EDGE Connected to PCL5 External Encoder Sensor...*)

Products.BufferPatternSize :=INT#2;

Products.PatternAwayDistance[0] :=LREAL#1730.0; (*Distance from Sensor*)

Products.PatternAwayDistance[1] :=LREAL#1930.0; (*Distance from Sensor + Minimum Part Length =200mm*)



Data Type: SWERROR_STRUCT

Used by the [Y_DigitalCamSwitch](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MySWERROR_STRUCT	SWERROR_STRUCT		
U	TrackNumber	INT	The last switch number where an invalid setting for TrackNumber occurred.	MySWERROR_STRUCT.TrackNumber
U	FirstOnPosition	INT	The last switch number where an invalid setting for FirstOnPosition occurred.	MySWERROR_STRUCT.FirstOnPosition
U	LastOnPosition	INT	The last switch number where an invalid setting for LastOnPosition occurred.	MySWERROR_STRUCT.LastOnPosition
U	AxisDirection	INT	The last switch number where an invalid setting for AxisDirection occurred.	MySWERROR_STRUCT.AxisDirection
U	CamSwitchMode	INT	The last switch number where an invalid setting for CamSwitchMode occurred.	MySWERROR_STRUCT.CamSwitchMode
U	Duration	INT	The last switch number where an invalid setting for Duration occurred.	MySWERROR_STRUCT.Duration
U	ImproperOnPosition	INT	The last switch number where an improper relationship between FirstOnPosition and LastOnPosition occurred.	MySWERROR_STRUCT.ImproperOnPosition
U	OnOffPositionError	INT	The last switch number where the OnCompensationScaler and/or OffCompensationScaler resulted in an improper relationship between the modified FirstOn and LastOn positions.	MySWERROR_STRUCT.OnOffPositionError



Data Type: TRACK_ARRAY

Supporting structure for [TRACK_REF](#). Used by the [Y_DigitalCamSwitch](#) function block.

Data Type Declaration

TYPE

TRACK_ARRAY: ARRAY[0..31] OF [TRACK_STRUCT](#);

END_TYPE



Data Type: TRACK_REF

Used by the [Y_DigitalCamSwitch](#) function block.

Data Type Declaration

TYPE

TRACK_REF:STRUCT

Track:[TRACK_ARRAY](#);

END_STRUCT;

END_TYPE



Data Type: TRACK_STRUCT

Supporting structure for [TRACK_ARRAY](#). Used by the [Y_DigitalCamSwitch](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyTRACK_STRUCT	TRACK_STRUCT		
U	OnCompensationScaler	LREAL	Compensation for the FirstOnPosition of each switch on the track. Positive values advance, negative values retard.	MyTRACK_STRUCT.OnCompensationScaler
U	OffCompensationScaler	LREAL	SpeedCompensation for the LastOnPosition of each switch on the track.	MyTRACK_STRUCT.OffCompensationScaler
U	Value	BOOL	The resulting status of the track after evaluating and combining all switches that affect the track.	MyTRACK_STRUCT.Value



Enumerated Types for PLCopen Toolbox

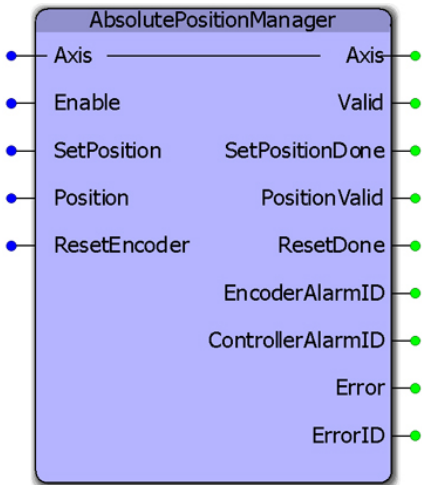
Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to zero-based integers (INT). Therefore, the first value equates to zero, the second to 1, etc. The format for enumerated types is as follows: ENUM:(0, 1, 2...) as displayed in the example below (MC_BufferMode#Aborting).

Enumerated Types Declaration

Enumerated Type	#INT Value	Enum Value	Description
TB_AxisType	Indicates the axis type for the ReadAxisParameters function block.		
	0	Servo	
	1	VFD	
	2	Stepper	
	3	Virtual	
MC_Direction	4	External	
	0	Positive_Direction	In a rotary application, forces the axis to move in a positive direction.
	1	Shortest_Way	For use in applications where the Load Type is configured as a rotary or modularized axis.
	2	Negative_Direction	In a rotary application, forces the axis to move in a negative direction.
	3	Current_Direction	For use in applications where the Load Type is configured as a rotary or modularized axis. Only applies if an existing move is in progress and another function block such as MC_MoveAbsolute or MC_MoveRelative is executed. Once the axis is at StandStill, using MC_Direction_CurrentDirection will default to the positive direction



AbsolutePositionManager



This function monitors for any controller or servo alarm related to the absolute encoder or battery backed encoder offset data stored in the controller. It can serve as the single point of monitoring, clearing, and defining the position of an absolute encoder. This function includes a retained Boolean output variable that once set, requires that the alarm be cleared through this function, and that the position of the encoder is redefined. The intention is to prevent the machine from operating until the position of the absolute encoder has been calibrated to the machine coordinates.

This function includes the following PLCopen function blocks: MC_ReadAxisError, MC_ReadAlarm, MC_ResetAbsoluteEncoder, Y_ClearAlarm and MC_SetPosition.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT			Default	
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	SetPosition	BOOL	Value of the axis position to be set when homing is Done.	FALSE

V	Position	LREAL	A positive or negative value within the coordinate system in user units.	LREAL#0.0
V	ResetEncoder	BOOL	Initiates the Y_ResetAbsoluteEncoder function to clear any absolute encoder related SERVOPACK alarm, including A.810 and A.CC0.	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	SetPositionDone	BOOL	Indicates that MC_SetPosition has successfully completed.	
V	PositionValid	BOOL	Indicates that the absolute encoder has no alarms, and the MC_SetPosition has been used at some point in the past to align the encoder with the mechanical system.	
V	ResetDone	BOOL	Indicates that the ResetEncoder request has completed successfully.	
V	EncoderAlarmID	UINT	ServoPack alarm related to the absolute encoder.	
V	ControllerAlarmID	UDINT	Controller alarm related to the SRAM or battery, which stores the absolute encoder offset.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

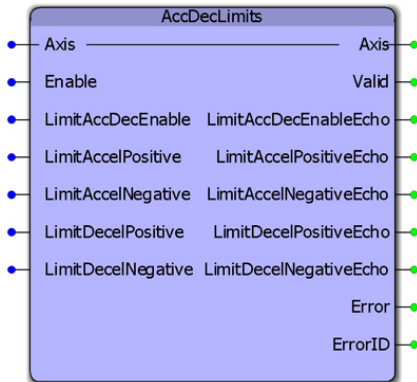
- The PROCONOS firmware library is required in the project when using this function block due to its use of some additional functionality provided there.
- This function block is not supported on the Sigma7-Siec because there is a Retain variable in the code which is not supported due to hardware limitations; there is no SRAM memory on the Sigma7-Siec.
- To clear the absolute encoder alarm from a ServoPack, use the 'ResetEncoder' input, then reset the ServoPack by either cycling power to the ServoPack or using the Y_ResetMechatrolink function block.
- Check the Hardware Configuration to ensure that the alarm format for Sigma III and higher drives is set for 3 digit alarm mode.
- See the [AbsolutePositionManager eLearning Module](#) on Yaskawa's YouTube channel.

Error Description

See the [Function Block ErrorID](#) list.



AccDecLimits



This function block manages the parameters associated with enabling/disabling the acceleration and deceleration limits. The limits can be enabled or disabled and the values of the limits can be input and verified at the output. The outputs are provided as an echo from the motion engine. This function allows for streaming of variable limits.

Library

PLCopen Toolbox

Parameters

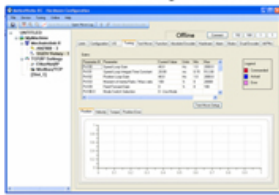
*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	LimitAccDecEnable	BOOL	Enables or Disables the Limit Accel Decel function. Parameter 1222 and 1232 are combined
V	LimitAccelPositive	LREAL	Parameter 1221
V	LimitAccelNegative	LREAL	Parameter 1220
V	LimitDecelPositive	LREAL	Parameter 1231
V	LimitDecelNegative	LREAL	Parameter 1230
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.

V	LimitAccDecEnableEcho	BOOL	Echo of Parameter 1222 ANDed with 1232
V	LimitAccelPositiveEcho	LREAL	Echo of parameter 1221 echoed from motion engine
V	LimitAccelNegativeEcho	LREAL	Echo of parameter 1220 echoed from motion engine
V	LimitDecelPositiveEcho	LREAL	Echo of parameter 1231 echoed from motion engine
V	LimitDecelNegativeEcho	LREAL	Echo of parameter 1230 echoed from motion engine
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

The function block uses MC_ReadBoolParameter, MC_WriteBoolParameter, MC_ReadParameter, and MC_WriteParameter.

Hardware Configuration



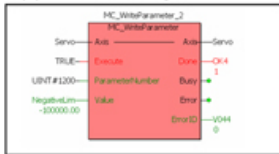
Always Loaded at power up

LimitAccelEnable [1222]
LimitAccelPositive [1221]
LimitAccelNegative [1220]

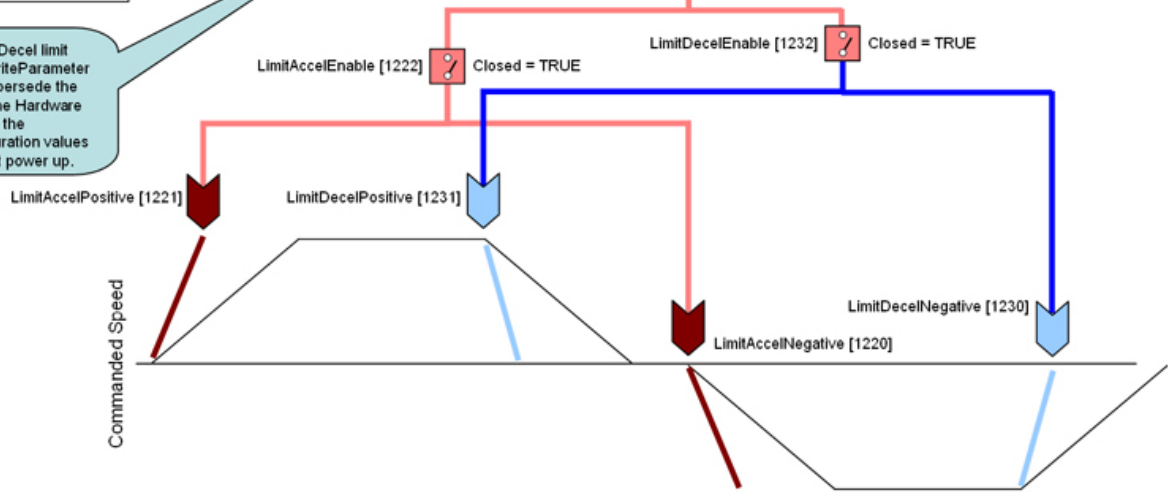
LimitDecelEnable [1232]
LimitDecelPositive [1231]
LimitDecelNegative [1230]



Application Program



Changing Accel / Decel limit values via MC_WriteParameter at run time will supersede the values stored in the Hardware Configuration, but the Hardware Configuration values will be reloaded at power up.



Accel / Decel Limits

- The software acceleration & deceleration limits are managed in the MPiec controller.
- When an acceleration or deceleration limit is exceeded, a controller alarm will be generated, obtainable via the MC_ReadAxisError function block, or the web server.
- The controller alarm will be 16#3202 0005 if the positive position limit is exceeded and 16#3202 0006 if the negative position limit is exceeded.

Acceleration Limits

- Acceleration is defined as increasing velocity away from zero.

- The parameters are called LimitAccelPositive and LimitAccelNegative, with values of UINT#1221 and UINT#1220 respectively. Use the MC_WriteParameter function block for these and all controller side parameters. Acceleration limit parameters are in user units / sec².
- To disable the acceleration limit, set LimitAccelEnable, parameter 1222 to zero.

Deceleration Limits

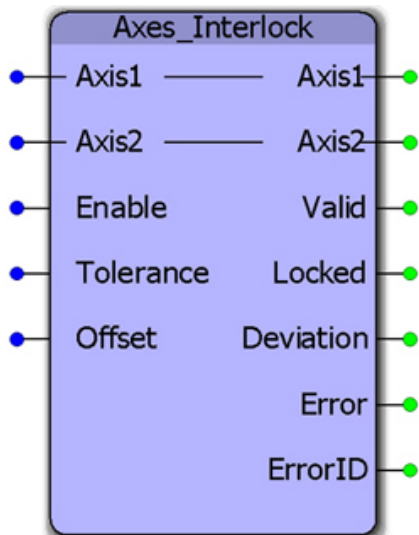
- Deceleration is defined by decreasing velocity towards zero.
- The parameters are called LimitDecelPositive and LimitDecelNegative, with values of UINT#1231 and UINT#1230 respectively. Use the MC_WriteParameter function block for these and all controller side parameters. Deceleration limit parameters are in user units / sec².
- To disable the deceleration limit, set LimitDecelEnable, parameter 1232 to zero.

Error Description

See the [Function Block ErrorID](#) list.



Axes_Interlock



This function block checks MC_ReadAxisError and the actual position of both axes to verify that they are both free of alarms and within the position tolerance specified. It is intended for use with axes that operate on the same mechanical load and must remain within tolerance to avoid equipment damage, such as an X, X Prime gantry system. The Locked output will be high to indicate that the axes are synchronized and free of errors.

Support for axes configured in rotary mode requires controller firmware 1.2.3 and PLCopen Toolbox v021.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis1	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
B	Axis2	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE

V	Tolerance	LREAL	The allowable position difference between the two axes in user units.	LREAL#0.0
V	Offset	LREAL	Offset between the two axes. This value will be considered when comparing the positions	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
V	Locked	BOOL	Indicates TRUE if neither axis has an alarm and the position deviation is less than the specified tolerance.	
V	Deviation	BOOL	The amount of positional difference between the two axes.	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

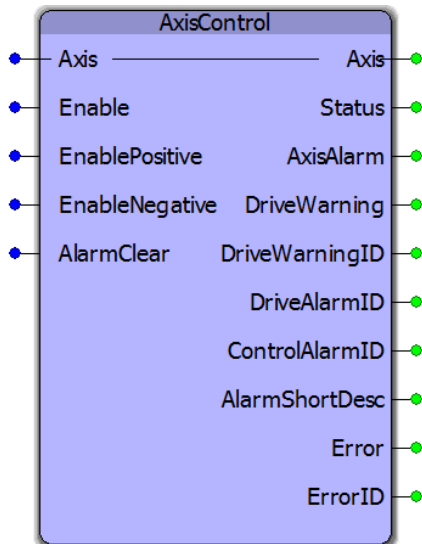
- It is assumed that the axes have the same user units because they are operating the same load.
- See the [AxesInterlock eLearning Module](#) on Yaskawa's YouTube channel.

Error Description

See the [Function Block ErrorID](#) list.



AxisControl



This function block combines MC_Power, MC_ReadAxisError, and MC_Reset and provides separate outputs for controller and drive alarms and warnings.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
E	EnablePositive	BOOL	Not Supported
E	EnableNegative	BOOL	Not Supported
V	AlarmClear	BOOL	Clears axis related alarms using MC_Reset.

VAR_OUTPUT			
B	Status	BOOL	TRUE if the drive is enabled. This output is derived from the Status output of MC_Power.
V	AxisAlarm	BOOL	Indicates if there is an axis specific alarm on either the controller or drive.
V	DriveWarning	BOOL	Indicates a warning on the drive, such as any A.9x display on a ServoPack.
V	DriveWarningID	UINT	Indicates the drive warning number, such as 95 (overload warning). Refer to the drive manual for troubleshooting.
V	DriveAlarmID	UINT	Indicates the drive alarm number, such as C9 (encoder disconnected). Refer to the drive manual for troubleshooting.
V	ControllerAlarmID	UDINT	Indicates the controller alarm ID number, such as 3302 0018. (shown in hex.) Refer to the Controller AlarmID list in the PLCopenPlus manual for troubleshooting.
V	AlarmShortDesc	STRING	Output from Y_GetAlarmDesc.ShortDesc.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

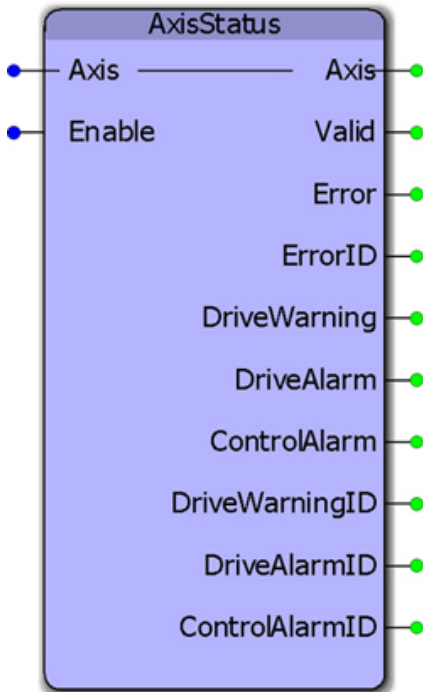
- When attempting to clear an alarm, the enable input must be FALSE or the alarm reset function will be blocked from executing.
- We recommend viewing the alarm and warning output ID's in Hex, because all Yaskawa ServoPack documentation lists the amplifier alarm codes in Hex. This simplifies alarm identification. Note that MotionWorks IEC may show the value at the output in decimal. For example, a DriveAlarmID of 2064 converted to hex is 810, which is the ServoPack alarm for the absolute encoder. "A81" will be displayed on the front of the ServoPack.
- This function only reports axis specific alarms and warnings. For general system alarms, use the Y_ReadAlarms function block from the PLCopenPlus firmware library.

Error Description

See the [Function Block ErrorID](#) list.



AxisStatus



This function block uses MC_ReadAxisError to provide further breakdown of the ErrorClass and AxisErrorID by providing BOOL and UINT outputs for the drive faults, and a DINT value for the controller alarm which is consistent with the 32 bit controller alarm reporting in the web server. This function was created for use inside the AxisControl function block in the PLCopen Toolbox. This function's outputs are available at the output of the AxisControl function block.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			Default

B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	DriveWarning	BOOL	Indicates a warning on the drive, such as any A.9x display on a ServoPack.	
V	DriveAlarm	BOOL	Indicates an alarm on the drive, such as A.71, overload. Refer to the appropriate drive manual for troubleshooting.	
V	ControllerAlarm	BOOL	Indicates a controller side axis alarm.	
V	DriveWarningID	UINT	Indicates the drive warning number, such as 95 (overload warning). Refer to the drive manual for troubleshooting.	
V	DriveAlarmID	UINT	Indicates the drive alarm number, such as C9 (encoder disconnected). Refer to the drive manual for troubleshooting.	
V	ControllerAlarmID	UDINT	Indicates the controller alarm ID number, such as 3302 0018. (shown in hex.) Refer to the Controller AlarmID list in the PLCopenPlus manual for troubleshooting.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

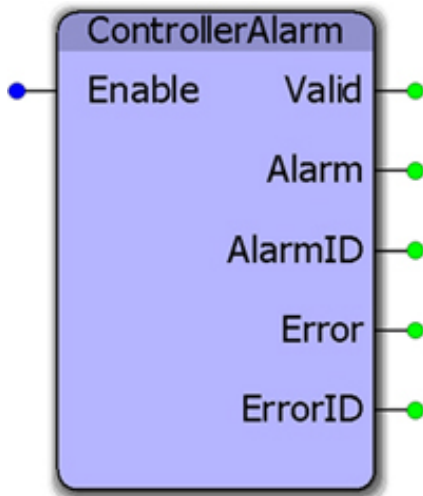
To simplify alarm identification, Yaskawa recommends viewing the alarm and warning output ID's in Hex, because all Yaskawa ServoPack documentation lists the amplifier alarm codes in Hex. Use the Debug Dialog menu in MotionWorks IEC to change the debug value display type. The controller alarm list in the webserver and in the PLCopenPlus help manual show the controller alarms in hex also.

Error Description

See the [Function Block ErrorID](#) list.



ControllerAlarm



This function block provides a BOOL output to indicate if there is a controller alarm not related to an axis. It uses the Y_ReadAlarm function block and determines if the AlarmID output is non-zero. This function is useful because the PLCopenPlus function Y_ReadAlarm does not have a Boolean output, just the AlarmID.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	Alarm	BOOL	Indicates if the controller has a non-axis related alarm.	
V	AlarmID	UDINT	This output provides the Controller Alarm ID. This output is reset when execute goes low.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

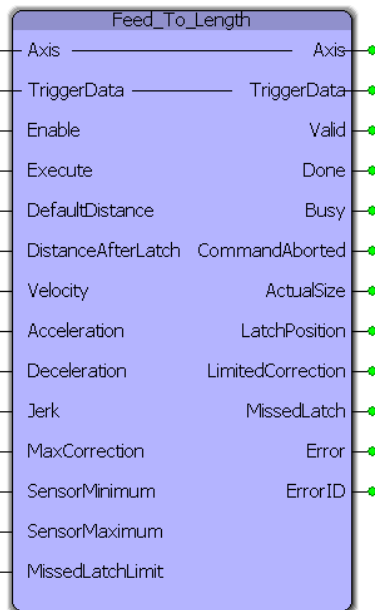
It is best to view the AlarmID in hex because the Controller AlarmID list in the PLCopenPlus manual displays all alarm codes in hex. This simplifies alarm category identification.

Error Description

See the [Function Block ErrorID](#) list.



Feed_To_Length



FeedToLength was designed for use with applications that index forward in one direction, and require on the fly adjustments of the actual index length based on a sensor input that occurs while the axis is moving. This block is a hybrid function block, meaning it use both types of PLCopen behaviors: Enable and Execute. The reason for this is so the function can monitor for consecutive latches and flag an Error for that condition. The Enable input allows this feature to operate. The Execute input initiates each move.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
V	TriggerData	TRIGGER_REF	Reference to the trigger signal source.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE

B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	DefaultDistance	LREAL	The default product length. This is the distance the axis will travel if a registration mark is not detected.	LREAL#0.0
V	DistanceAfterLatch	LREAL	The desired additional travel distance after the registration mark is detected	LREAL#0.0
B	Velocity	LREAL	Absolute value of the velocity in user units/second.	LREAL#0.0
B	Acceleration	LREAL	Value of the acceleration in user units/second ² (acceleration is applicable with same sign of torque and velocity)	LREAL#0.0
B	Deceleration	LREAL	Value of the deceleration in user units/second ² (deceleration is applicable with opposite signs of torque and velocity.)	LREAL#0.0
E	<i>Jerk</i>	<i>LREAL</i>	<i>Not supported; reserved for future use. Value of the jerk in [user units / second³].</i>	<i>LREAL#0.0</i>
V	MaxCorrection	LREAL	Limits the amount of correction that can be applied. This prevents the machine from trying to make correction that is too large to occur and still make a good product. This is the most amount of change to the DefaultDistance that will be made to any one product index.	LREAL#0.0
V	SensorMinimum	LREAL	The earliest slave position where a sensor position is valid for correction.	LREAL#0.0
V	SensorMaximum	LREAL	The latest slave position where a sensor position is valid for correction.	LREAL#0.0
V	MissedLatchLimit	UINT	The number of consecutive DefaultDistances allowed to occur without seeing a registration mark in the window, and not cause an Error. Valid registration marks will reset the internal counter.	UINT#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
V	ActualSize	LREAL	The actual indexed distance.	
V	LatchPosition	LREAL	The slave's position in the CamTable when the latch occurred.	
V	LimitedCorrection	BOOL	Indicates that the MaxCorrection is limiting the required correction.	
V	MissedLatch	BOOL	Flag which indicates that the controller did not find a valid registration mark within the SensorMinimum and SensorMaximum positions.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

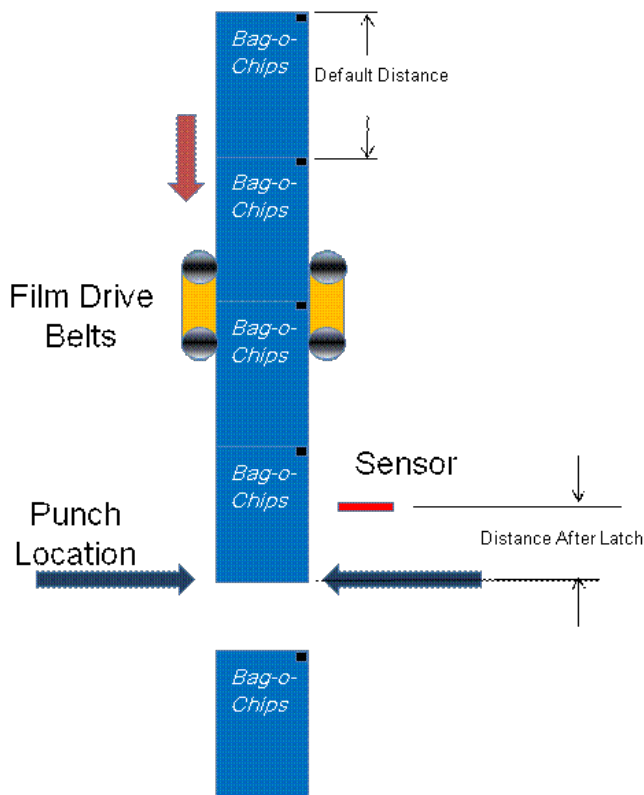
- This function block is designed to use a high speed registration sensor wired into the ServoPack's latch input hardware. Use the [TRIGGER_REF](#) input to specify the input on the amplifier where the sensor is wired (EXT1, EXT2, or EXT3.) The sensor must be wired to one of these inputs for this function block.

Error Description

See the [Function Block ErrorID](#) list.

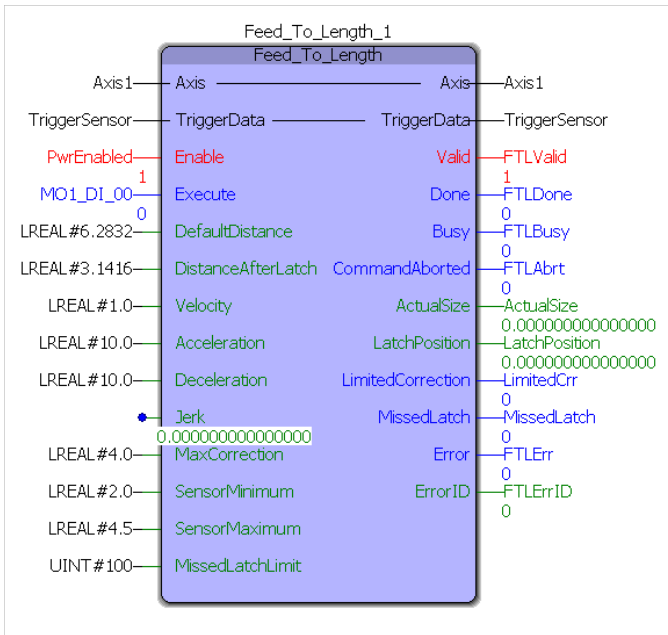
Example

Consider a case where the default distance between successive products is 6.2832 units. Let the distance between the sensor (wired to the high speed registration input) and the target position where the product will be processed be 3.1416 units. DistanceAfterLatch = 3.1416.

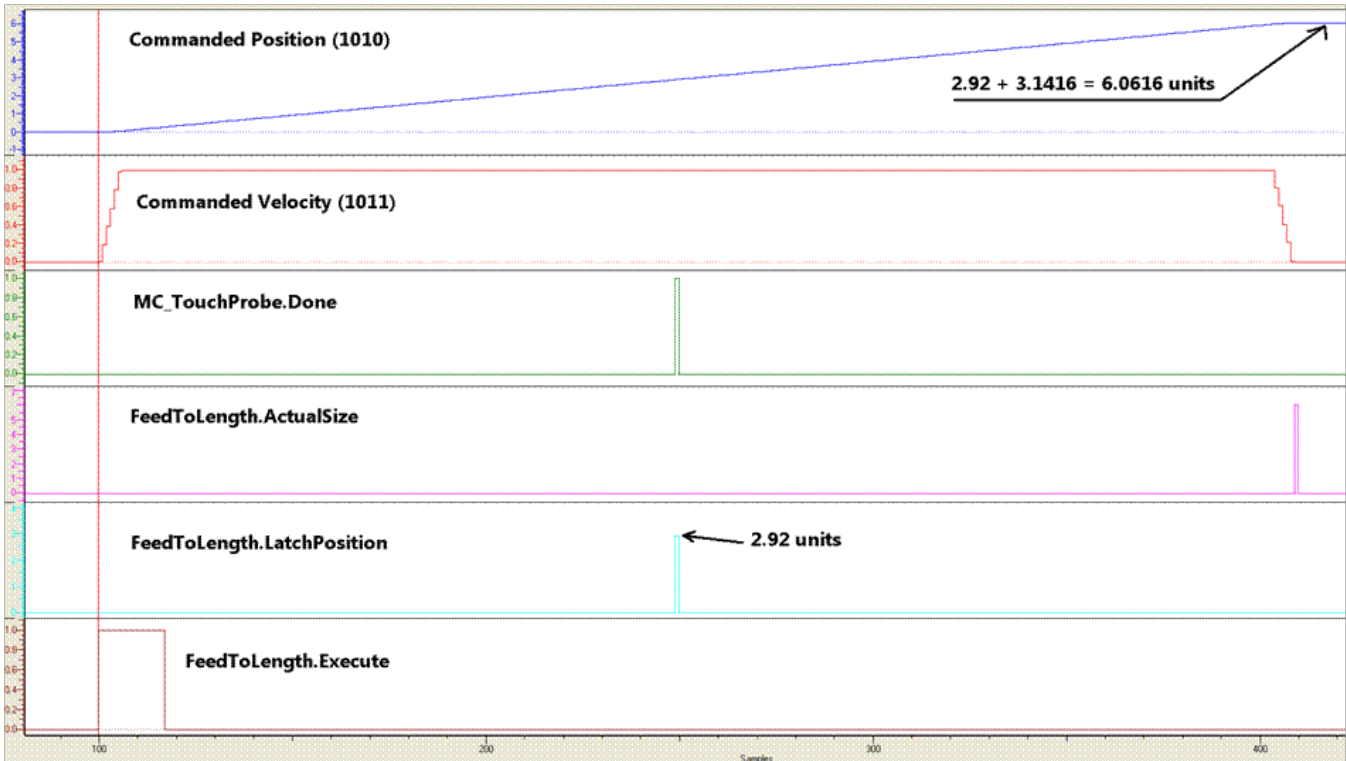


MaxCorrection limits the correction if an erroneous registration mark is captured and the calculation results in a large correction distance.

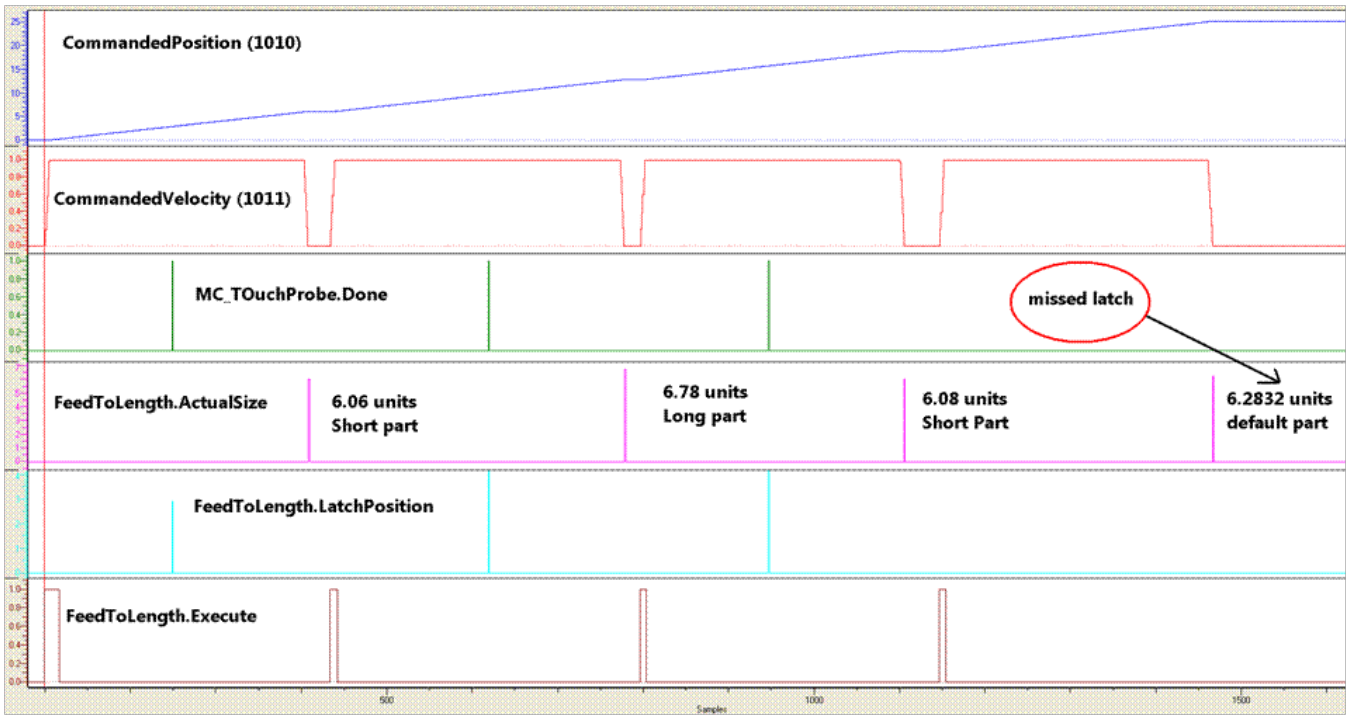
SensorMinimum and SensorMaximum provide window in which a registration mark must be detected to be considered a valid registration mark. In this example, the mark is expected around 3.1416 units, and only marks detected between 2.0 to 4.5 are accepted. Set the window as small as appropriate for the application.



The Feed_To_Length function block will position the axis exactly 3.1416 units (DistanceAfterLatch) after the registration mark was detected.

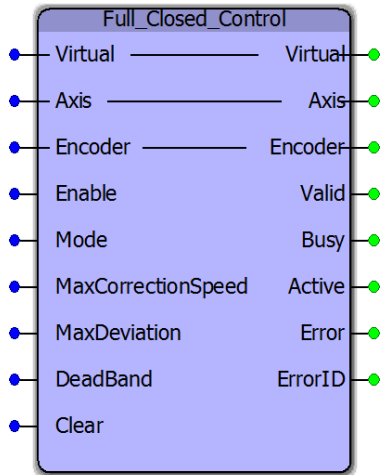


The Feed_To_Length function block will position the axis exactly 3.1416 units (DistanceAfterLatch) after the registration mark is detected for varying product lengths.





Full_Closed_Control



This function block uses an external encoder position to provide improved positioning for machines that have loose mechanics or applications that must account for material slippage. This function block is very useful for MP2600iec applications which cannot take advantage of the FC100 option card. Other features include the ability to switch from full closed to normal motor encoder feedback, which is useful for applications where the external encoder is tracking a product which may not be present at all times.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Virtual	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
V	Encoder	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT			Default	
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE

V	Mode	BOOL	This turns on and off full closed loop operation. Set FALSE to select Full closed operation, set TRUE to disable Full Closed operation.	FALSE
V	MaxCorrectionSpeed	LREAL	Limit the Maximum correction applied to prevent overshoot and instability. Set this value in user units/sec. It applies only to the correction portion of the command. For example, if the virtual axis is commanded to operate at a velocity of 2000 user units/sec and the MaximumCorrectionSpeed is set to 250 user units/sec, the axis may achieve a velocity of 2250 user units / sec.	LREAL#0.0
V	MaxDeviation	LREAL	If the absolute difference between the axis position and the full closed encoder position exceeds MaxDeviation, the function block will report Error and stop operating the axis. Set this value in user units.	LREAL#0.0
V	DeadBand	LREAL	When the absolute difference between the axis and the full closed encoder is less than this amount, no correction will be applied. Set this value in user units.	LREAL#0.0
V	Clear	BOOL	Resets the internal measurement of the difference between the motor encoder and the full closed encoder.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Active	BOOL	For buffered modes, this output is set high at the moment the block takes control of the axis. For non buffered modes, the outputs Busy and Active have the same value.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

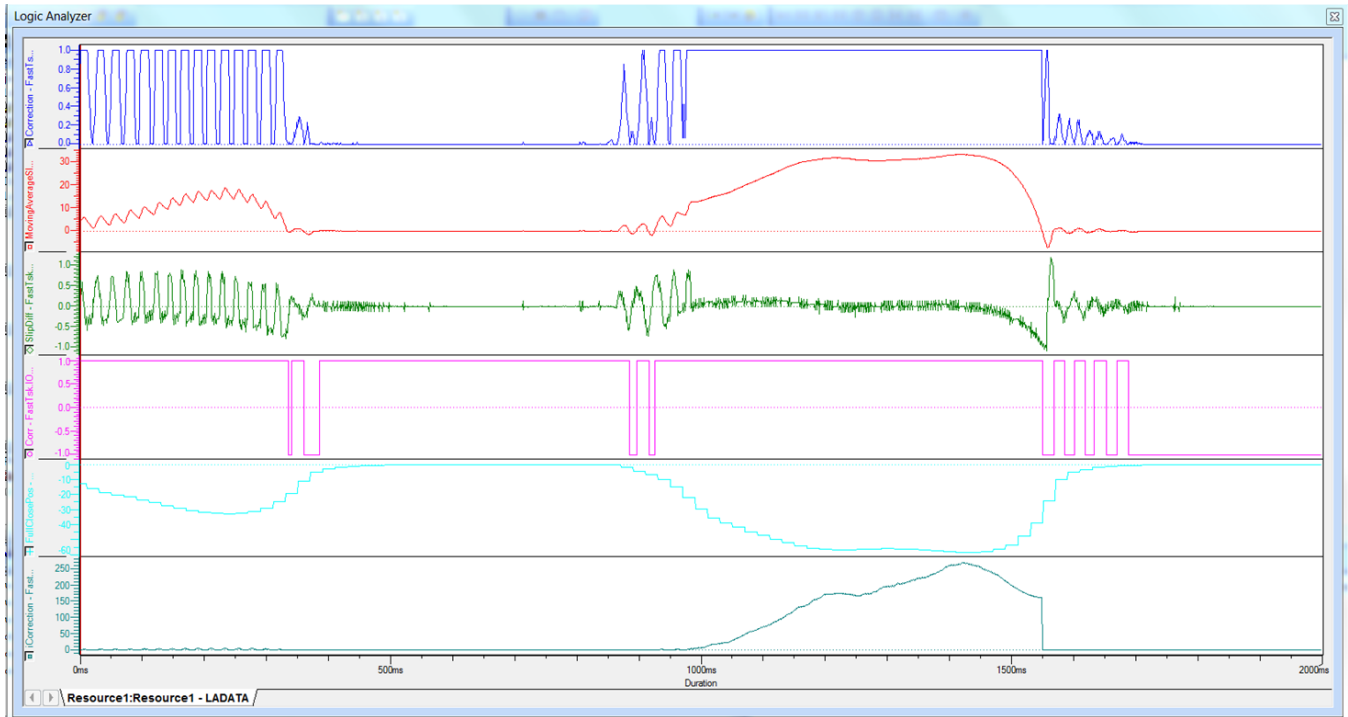
- The Yaskawa Toolbox v300 or higher is required when using the Full_Closed_Control function block. The Yaskawa Toolbox must be included and placed above PLCOpen Toolbox in the Libraries folder of your MotionWorks IEC project.
- For the best performance, this function block must be executed in a task running at the same interval as the Mechatrolink (or MP2600iec DP ram) update rate. Ideally, the function is executed in a task at 4 mSec or faster.
- The user application must pre-set the position of the Virtual axis and the Encoder axis before enabling this function. The two positions must be less than MaxDeviation, or an Error will be generated immediately.
- For applications where the full closed encoder is in contact with product fed into the machine and may experience slip due to feed roll pressure, etc. the Clear input can be used in conjunction with the cycle of the machine, or each index motion. For example, if up to 1 mm of slip is known to occur normally while indexing 25 cm, set the MaxDeviation input to 2 or 3 mm, and trigger the Clear input after each index is finished. This will allow the function to monitor for excessive slippage and generate an error only if it exceeds 2 mm during a single index, but will allow for much more deviation to accumulate over long periods of operation.
- Mode can be set TRUE for situations when the material monitored by the full closed encoder is not in contact with the full closed encoder. This may be when the machine is being set up, or a jam is being cleared. The Full_Closed_Control function block can operate the motor using the motor encoder alone. If during the time that Mode=TRUE the MaxDeviation is exceeded, it will not cause an Error, however, when Mode is again set to FALSE, the difference between the Virtual axis and the Encoder must be within MaxDeviation, or an Error will be generated. The application program must set the two positions accordingly. It may be necessary to disable this function block and re-Enable it after using the Mode which doesn't close the position around the Encoder.

Error Description

See the [Function Block ErrorID](#) list.

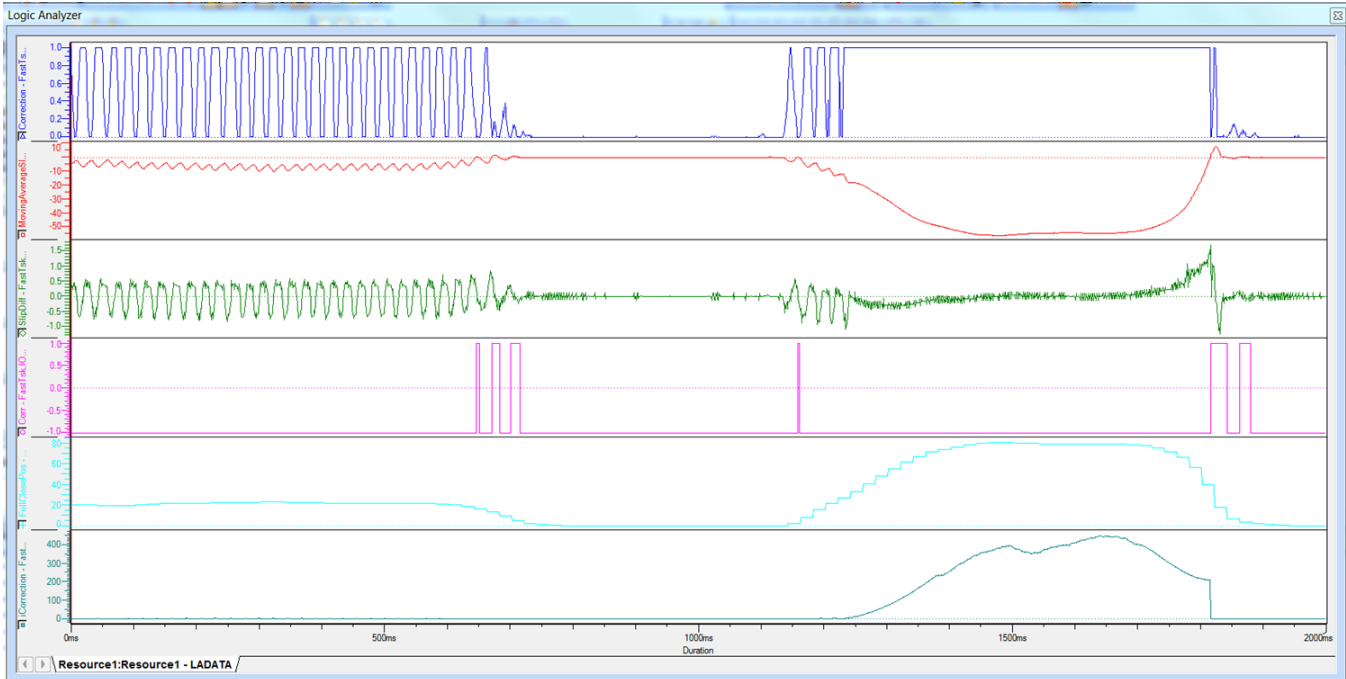
Example 1

In this example, the Logic Analyzer is showing when positive slip happens, correction is added to the Axis and how the moving average slip changes during the correction.



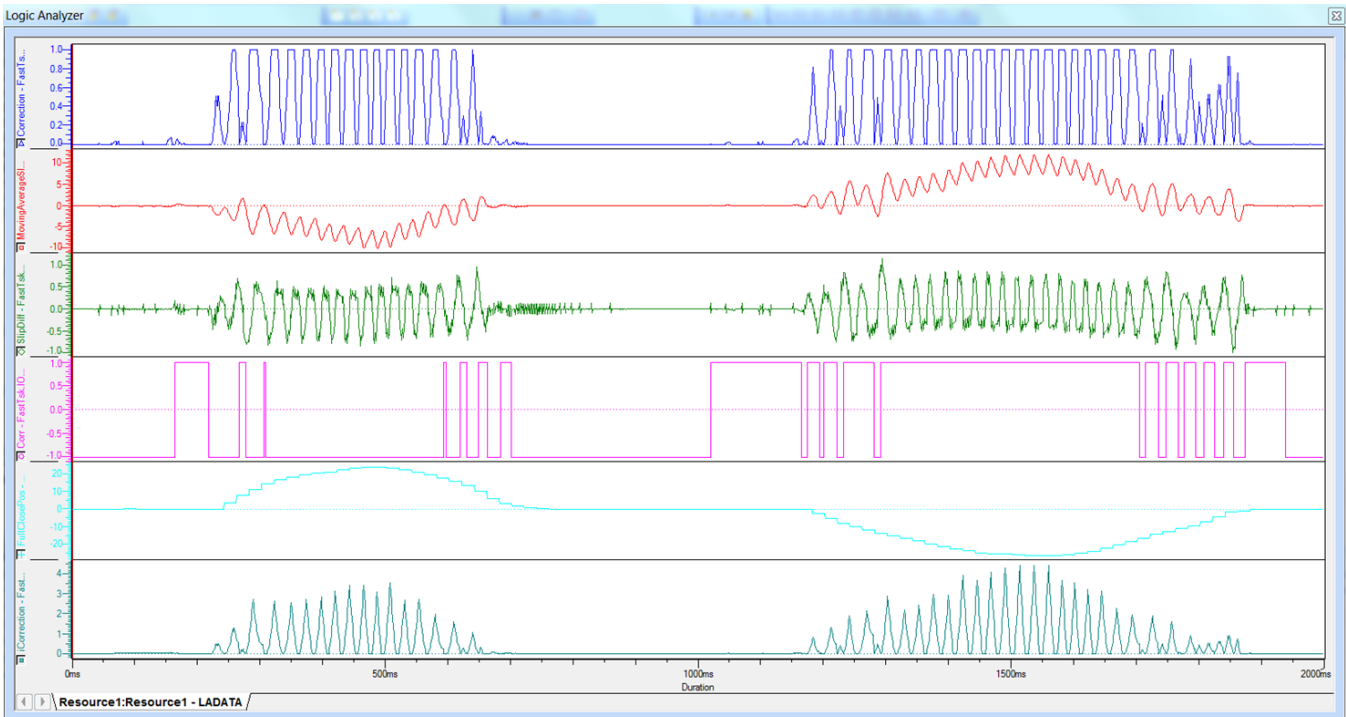
Example 2

In this example, the Logic Analyzer shows negative slip, and the correction added to Axis.



Example 3

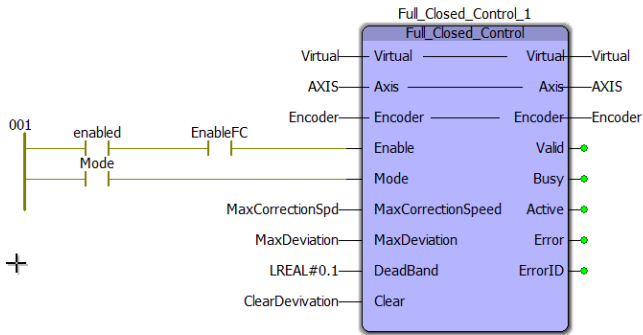
In this example, the Logic Analyzer shows both positive and negative slip.



Example 4

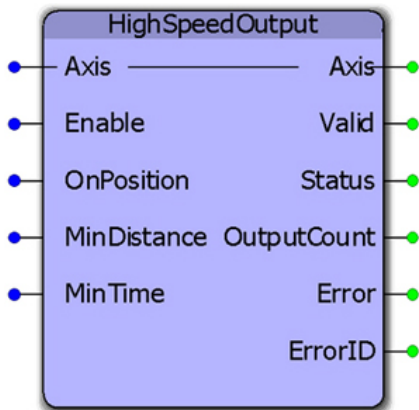
When Mode is ON, full closed loop function is disabled.

When Mode is OFF, full closed loop function operates.





HighSpeedOutput



This function block combines several of the parameters for use with the High Speed Output function available on the LIO-01, LIO-02, LIO-06, and MP2600iec. It allows changing the "OnPosition" value on the fly. While the "OnPosition" will be triggered at the hardware level with a response time of 13us, the output will be turned off when either the MinDistance has been travelled or the MinTime has elapsed, which will be based on the application scan in which this function is operating.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	OnPosition	LREAL	Position at which output must turn on.	LREAL#0.0
V	MinDistance	LREAL	Minimum distance that must occur before the output turns off.	LREAL#0.0
V	MinTime	TIME	Minimum time that must elapse before the output must turn off.	T#0s
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	Status	BOOL	Indicates the status of the hardware.	

V	OutputCount	UDINT	Indicates the number of times the output turned on.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

High Speed Output Quick Reference

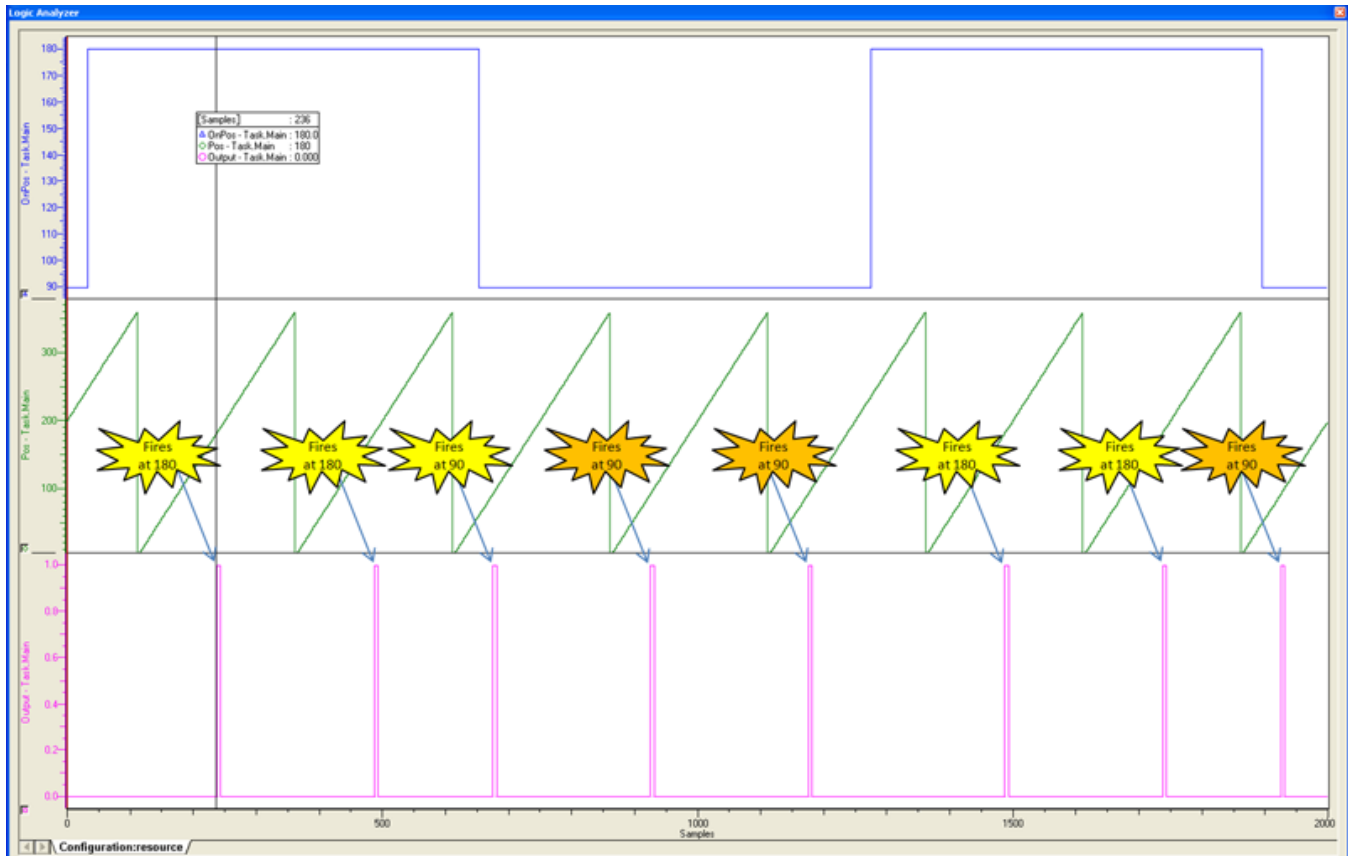
Device	Output Number	Pin Number	Software Default Name
LIO-01	DO-00	B14	Mpp_DO_00
LIO-02	DO-00	B14	Mpp_DO_00
LIO-06	DO-07	49	Mpp_DO_07
MP2600	DO-07	44, 49	MO1_DO_01

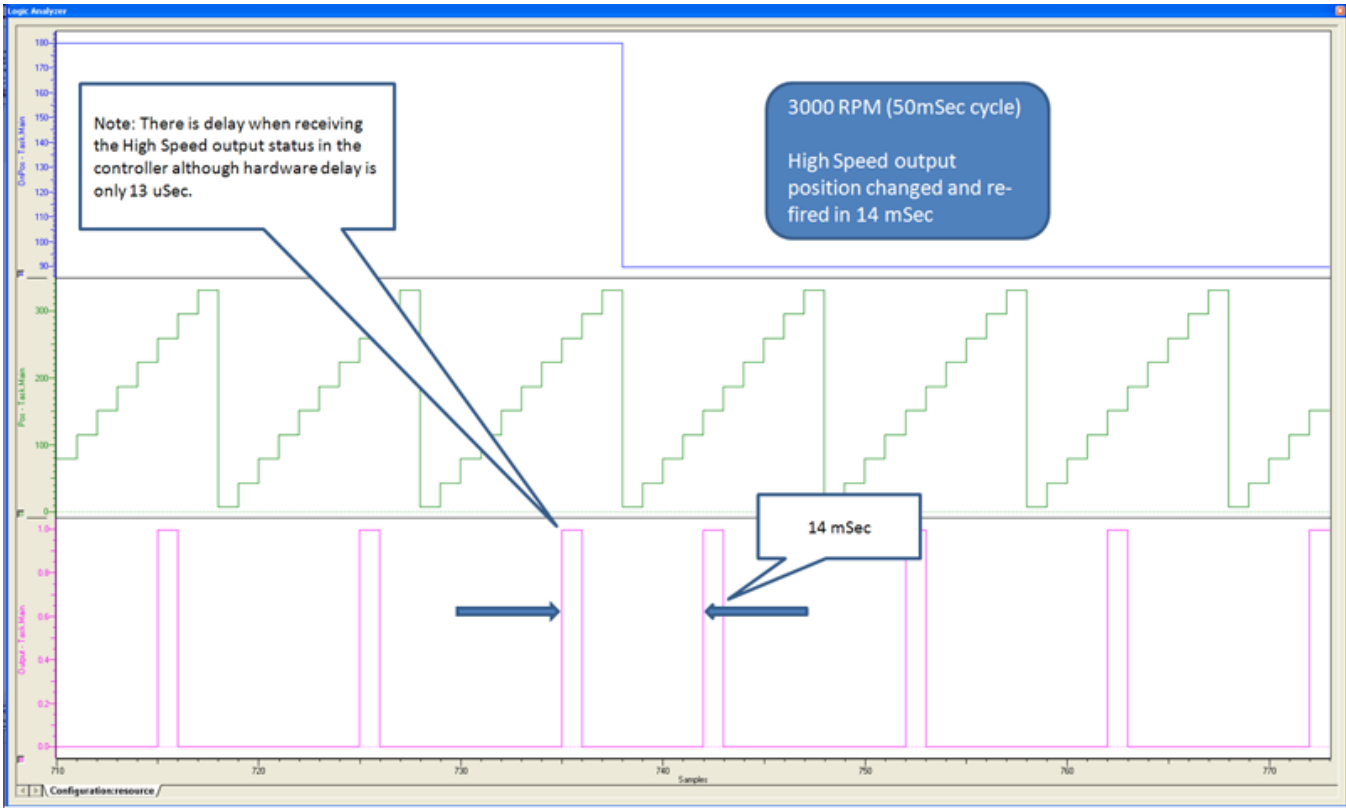
- See the [HighSpeedOutput eLearning Module](#) on Yaskawa's YouTube channel.
- The Global variable mapped to the physical output will not report the state of the output; this variable only reflects the application programs control of the output when not using the high speed output in hardware mode.. Refer to the function block's VAR_OUTPUT for the status of the high speed output

Error Description

See the [Function Block ErrorID](#) list.

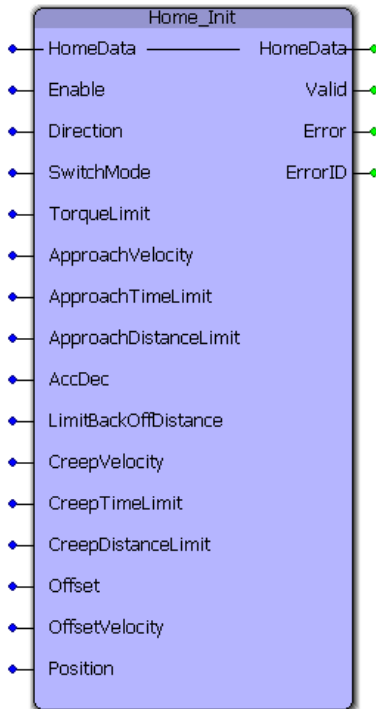
Timing Diagram







Home_Init



This function block provides a method to initialize the HomeStruct data for use with all HOME_** function blocks. It is useful for programmers who prefer to avoid structured text for initializing HomeStruct values.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	HomeData	HomeStruct	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			Default
B	Enable	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
			FALSE

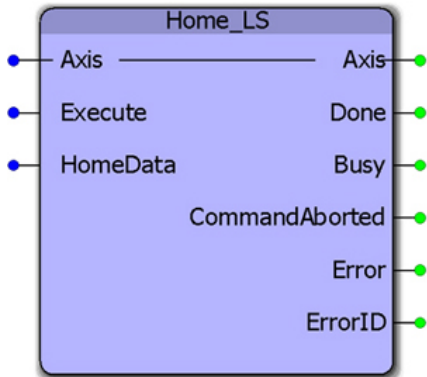
B	Direction	MC_Direction	Direction of travel for homing.	
B	SwitchMode	MC_SwitchMode	Edge On is the only mode supported.	
B	TorqueLimit	LREAL	Torque limit while attempting homing. In percentage of rated torque of the servo.	LREAL#0.0
V	ApproachVelocity	LREAL	Velocity used to approach limit switch or c channel.	LREAL#0.0
V	ApproachTimeLimit	LREAL	Time limit for the homing attempt in seconds .	LREAL#0.0
V	ApproachDistanceLimit	LREAL	Distance limit for the homing attempt.	LREAL#0.0
V	AccDec	LREAL	Acceleration/deceleration for offset moves.	LREAL#0.0
V	LimitBackOffDistance	LREAL	Distance limit for back off move after a limit switch is encountered.	LREAL#0.0
V	CreepVelocity	LREAL	Velocity to creep to theC channel.	LREAL#0.0
V	CreepTimeLimit	LREAL	Time limit for the creep attempt in seconds .	LREAL#0.0
V	CreepDistanceLimit	LREAL	Distance limit for the creep attempt	LREAL#0.0
V	Offset	LREAL	Offset distance to move after the limit switch or C channel.	LREAL#0.0
V	OffsetVelocity	LREAL	Velocity of the offset move after the limit switch or C channel.	LREAL#0.0
B	Position	LREAL	Position to be defined as the home position.	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Error Description

No Errors will be generated.



Home_LS



This function block combines the PLCopen function blocks MC_StepLimitSwitch, MC_MoveRelative, and MC_SetPosition to make a sequence that detects the limit switch, performs an offset move away from the limit, and sets a home position.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
V	HomeData	HomeStruct	User defined Data Type in the PLCopen Toolbox, contains all related homing parameters.
VAR_OUTPUT			
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.

B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

- This function is intended to operate only with a ServoPack's POT or NOT signal detection. HomeData.SwitchMode only supports "EdgeOn." Configure the ServoPack Pn 50A and 50B appropriately.

Error Description

See the [Function Block ErrorID](#) list.

Example

Use a ST POU to initialize the data required for HomeData. To save time, copy & paste the example initialization into your project.

(Copy & Paste, then search & replace the headings in the following section to speed the initialization of the homing data. **)**

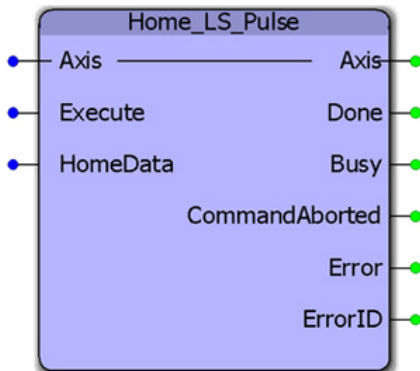
```

HomeStruct_ReplaceMe.AccDec:=LREAL#500.0; (* In User units /sec2 as set in the Hardware Configuration *)
HomeStruct_ReplaceMe.ApproachDistanceLimit:=LREAL#500.0; (* In User units as set in the Hardware Configuration *)
HomeStruct_ReplaceMe.ApproachTimeLimit:=LREAL#500.0; (* In seconds *)
HomeStruct_ReplaceMe.ApproachVelocity:=LREAL#500.0; (* In User units / sec as set in the Hardware Configuration *)
HomeStruct_ReplaceMe.CreepDistanceLimit:=LREAL#500.0; (* In User units as set in the Hardware Configuration *)
HomeStruct_ReplaceMe.CreepTimeLimit:=LREAL#500.0; (* In seconds *)
HomeStruct_ReplaceMe.CreepVelocity:=LREAL#500.0; (* In User units / sec as set in the Hardware Configuration *)
HomeStruct_ReplaceMe.Direction:=INT#0; (* MC_Direction#Positive_Direction; *)
HomeStruct_ReplaceMe.Offset:=LREAL#500.0; (* In User units as set in the Hardware Configuration *)
HomeStruct_ReplaceMe.OffsetVelocity:=LREAL#500.0; (* In User units / sec as set in the Hardware Configuration *)
HomeStruct_ReplaceMe.Position:=LREAL#500.0; (* In User units as set in the Hardware Configuration *)
HomeStruct_ReplaceMe.SwitchMode:=INT#2; (* MC_SwitchMode#EdgeOn; *)
HomeStruct_ReplaceMe.TorqueLimit:=LREAL#500.0; (* In percentage of rated torque of the servo *)

```



Home_LS_Pulse



This function block combines the PLCopen function blocks MC_StepLimitSwitch, MC_StepRefPulse, MC_MoveRelative, and MC_SetPosition to make a sequence that detects the limit switch, reverses to the C channel, performs an offset move away from the limit, and sets a home position.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
V	HomeData	HomeStruct	User defined Data Type in the PLCopen Toolbox, contains all related homing parameters.
VAR_OUTPUT			
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.

B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

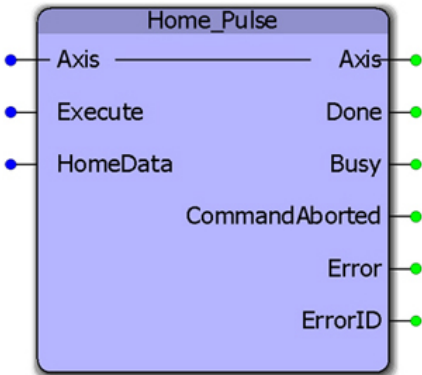
- This function is intended to operate only with a ServoPack's POT or NOT signal detection. HomeData.SwitchMode only supports "EdgeOn." Configure the ServoPack Pn 50A and 50B appropriately.
- See the [Home_LS_Pulse eLearning Module](#) on Yaskawa's YouTube channel.

Error Description

See the [Function Block ErrorID](#) list.



Home_Pulse



This function block combines the PLCopen function blocks MC_StepRefPulse, MC_MoveRelative, and MC_SetPosition to make a sequence that detects the limit switch, reverses to the C channel, performs an offset move away from the limit, and sets a home position.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
V	HomeData	HomeStruct	User defined Data Type in the PLCopen Toolbox, contains all related homing parameters.
VAR_OUTPUT			
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)

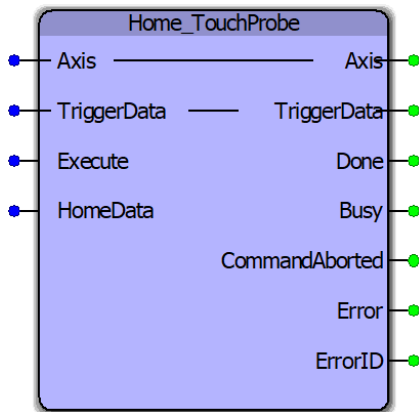
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Error Description

See the [Function Block ErrorID](#) list.



Home_TouchProbe



This function block combines the PLCopen function blocks MC_MoveRelative, MC_TouchProbe, MC_MoveAbsolute and MC_SetPosition to make a sequence that initiates motion on the axis until a signal is detected on the sensor connected to the high speed latch input of the servo. The axis then performs an offset move from the latched position, and sets a home position.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
V	TriggerData	TRIGGER_REF	Reference to the trigger signal source.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	HomeData	HomeStruct	User defined Data Type in the PLCopen Toolbox, contains all related homing parameters.	All zeros in structure
VAR_OUTPUT				

B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

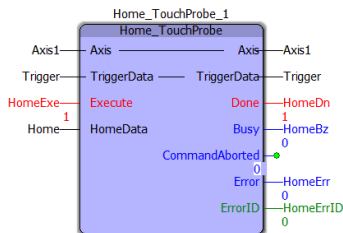
Use HomeData.ApproachDistanceLimit to set the maximum travel distance while waiting for the TouchProbe function to detect the sensor. Enter a positive or negative value, this function block does not use HomeData.Direction.

Error Description

See the [Function Block ErrorID](#) list.

Example

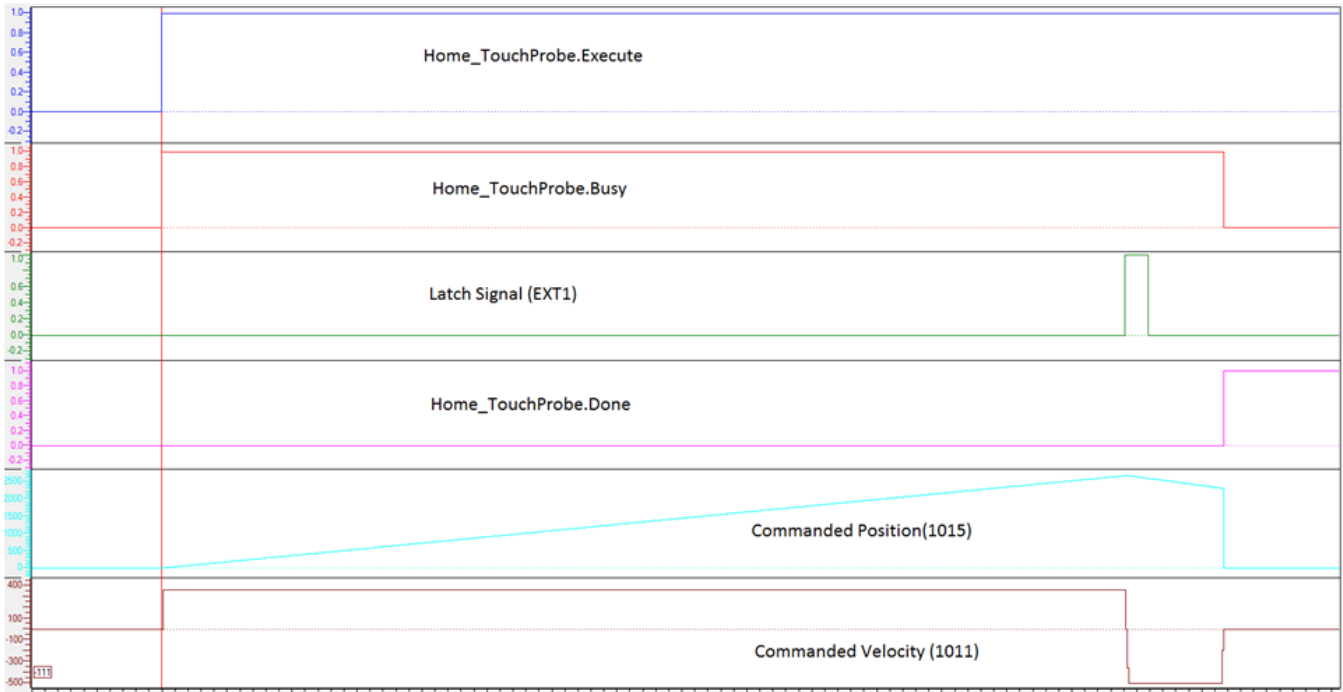
The example below illustrates how the Home_TouchProbe function block homes an axis based on the latch detected on one of the three EXT channels on the servo. Plots of the commanded speed and positions are shown to describe a negative home offset of 360 units after the latch is detected.



```

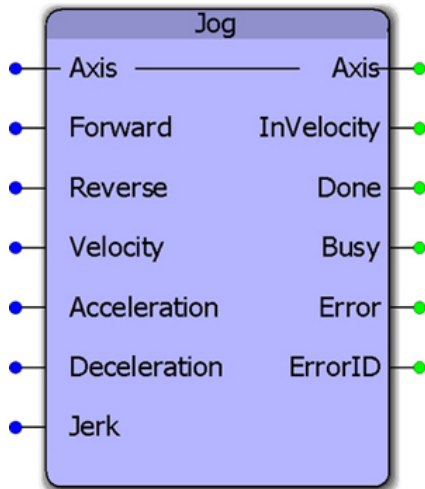
36000.0000000 | Home.AccDec:=LREAL#36000.0; (* In User units /sec2 as set in the Hardware Configuration *)
20000.0000000 | Home.ApproachDistanceLimit:=LREAL#20000.0; (* In User units as set in the Hardware Configuration *)
500.0000000 | Home.ApproachTimeLimit:=LREAL#500.0; (* In seconds *)
360.0000000 | Home.ApproachVelocity:=LREAL#360.0; (* In User units / sec as set in the Hardware Configuration *)
2000.0000000 | Home.CreepDistanceLimit:=LREAL#2000.0; (* In User units as set in the Hardware Configuration *)
500.0000000 | Home.CreepTimeLimit:=LREAL#500.0; (* In seconds *)
90.0000000 | Home.CreepVelocity:=LREAL#90.0; (* In User units / sec as set in the Hardware Configuration *)
0 | Home.Direction:=INT#0; (* MC_Direction#Positive_Direction: *)
-360.0000000 | Home.Offset:=LREAL#-360.0; (* In User units as set in the Hardware Configuration *)
500.0000000 | Home.OffsetVelocity:=LREAL#500.0; (* In User units / sec as set in the Hardware Configuration *)
0.0000000 | Home.Position:=LREAL#0.0; (* In User units as set in the Hardware Configuration *)
2 | Home.SwitchMode:=INT#2; (* MC_SwitchMode#EdgeOn *)
300.0000000 | Home.TorqueLimit:=LREAL#300.0; (* In percentage of rated torque of the servo *)

```





Jog



This function block combines the PLCopen functions MC_MoveVelocity and MC_Stop to provide a jogging feature only while the Forward or Reverse inputs are TRUE. The function will default to stopping the axis when neither (or both) are high.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT			Default	
V	Forward	BOOL	Runs the axis in a forward direction when TRUE.	FALSE
V	Reverse	BOOL	Runs the axis in a Reverse direction when TRUE.	FALSE
B	Velocity	LREAL	Absolute value of the velocity in user unit-s/second.	LREAL#0.0
B	Acceleration	LREAL	Value of the acceleration in user unit-s/second ² (acceleration is applicable with same sign of torque and velocity)	LREAL#0.0

B	Deceleration	LREAL	Value of the deceleration in user units/second ² (deceleration is applicable with opposite signs of torque and velocity.)	LREAL#0.0
E	Jerk	LREAL	<i>Not supported; reserved for future use. Value of the jerk in [user units / second³].</i>	LREAL#0.0
VAR_OUTPUT				
B	InVelocity	BOOL	Set high when the axis first reaches the specified velocity (function is complete). If the function is re executed with a new velocity, the output will remain high. It will go low when an MC_Stop block is executed. If using the Jog function block from the PLCopen Toolbox, this output will go low when both the Forward and Reverse inputs are low.	
B	Done	BOOL	Turns on for one scan when the axis comes to a stop after both Forward and Reverse inputs go FALSE.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

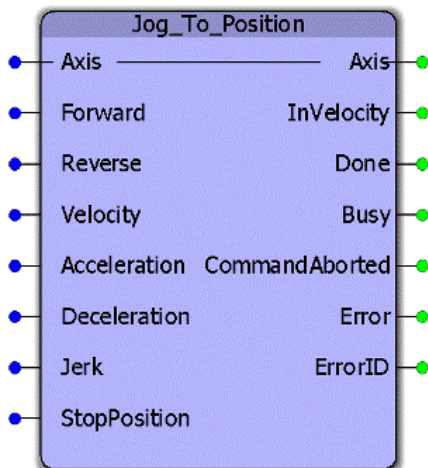
- The velocity can be changed on the fly without toggling the Forward or Reverse input. The code inside this function block will detect if the velocity has changed, and automatically re trigger the MC_MoveVelocity function block inside. Starting in PLCopen Toolbox v202, changes in Acceleration and Deceleration are detected and can be changed on the fly.
- See the [Jog eLearning Module](#) on Yaskawa's YouTube channel.

Error Description

See the [Function Block ErrorID](#) list.



Jog_To_Position



This function block combines the PLCopen functions MC_MoveVelocity and MC_MoveAbsolute to provide a jogging feature specifically for rotary axes that must stop at a specific position after an indefinite period of motion.

Library

PLCopen Toolbox

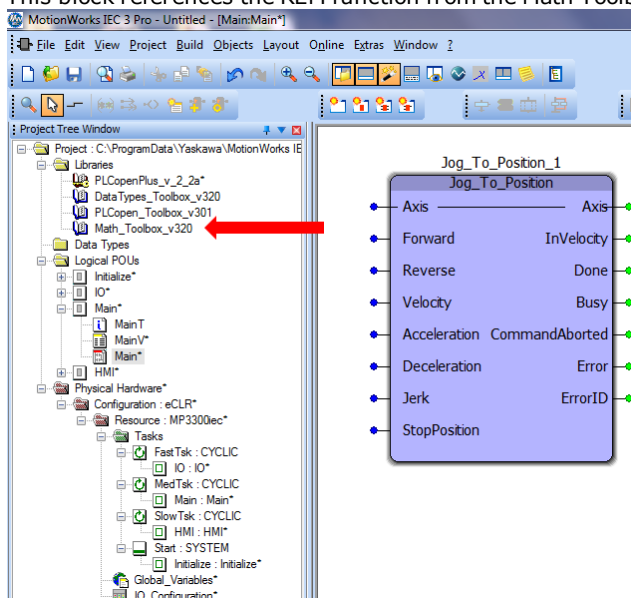
Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			
V	Forward	BOOL	Runs the axis in a forward direction when TRUE.
V	Reverse	BOOL	Runs the axis in a Reverse direction when TRUE.
B	Velocity	LREAL	Absolute value of the velocity in user units/second.
B	Acceleration	LREAL	Value of the acceleration in user units/second ² (acceleration is applicable with same sign of torque and velocity)

B	Deceleration	LREAL	Value of the deceleration in user units/second ² (deceleration is applicable with opposite signs of torque and velocity.)	LREAL#0.0
E	Jerk	LREAL	Not supported; reserved for future use. Value of the jerk in [user units / second ³].	LREAL#0.0
V	StopPosition	LREAL	Once the Forward and Reverse inputs are false, the axis will decelerate to a stop at the specified StopPosition using the specified deceleration rate	LREAL#0.0
VAR_OUTPUT				
B	InVelocity	BOOL	Set high when the axis first reaches the specified velocity (function is complete). If the function is re executed with a new velocity, the output will remain high. It will go low when an MC_Stop block is executed. If using the Jog function block from the PLCopen Toolbox, this output will go low when both the Forward and Reverse inputs are low.	
B	Done	BOOL	Turns on for one scan when the axis comes to a stop after both Forward and Reverse inputs go FALSE.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

- When using this function block, the Math Toolbox is also required because JogToPosition references the REM function.
- The velocity, acceleration, and deceleration can be changed on the fly without toggling the Forward or Reverse input. The code inside this function block will detect if the input values have changed, and automatically re trigger the MC_MoveVelocity function block inside. Starting in PLCopen Toolbox v202, changes in Acceleration and Deceleration are detected and can be changed on the fly.
- This block references the REM function from the Math Toolbox, so it must be included in the project tree.

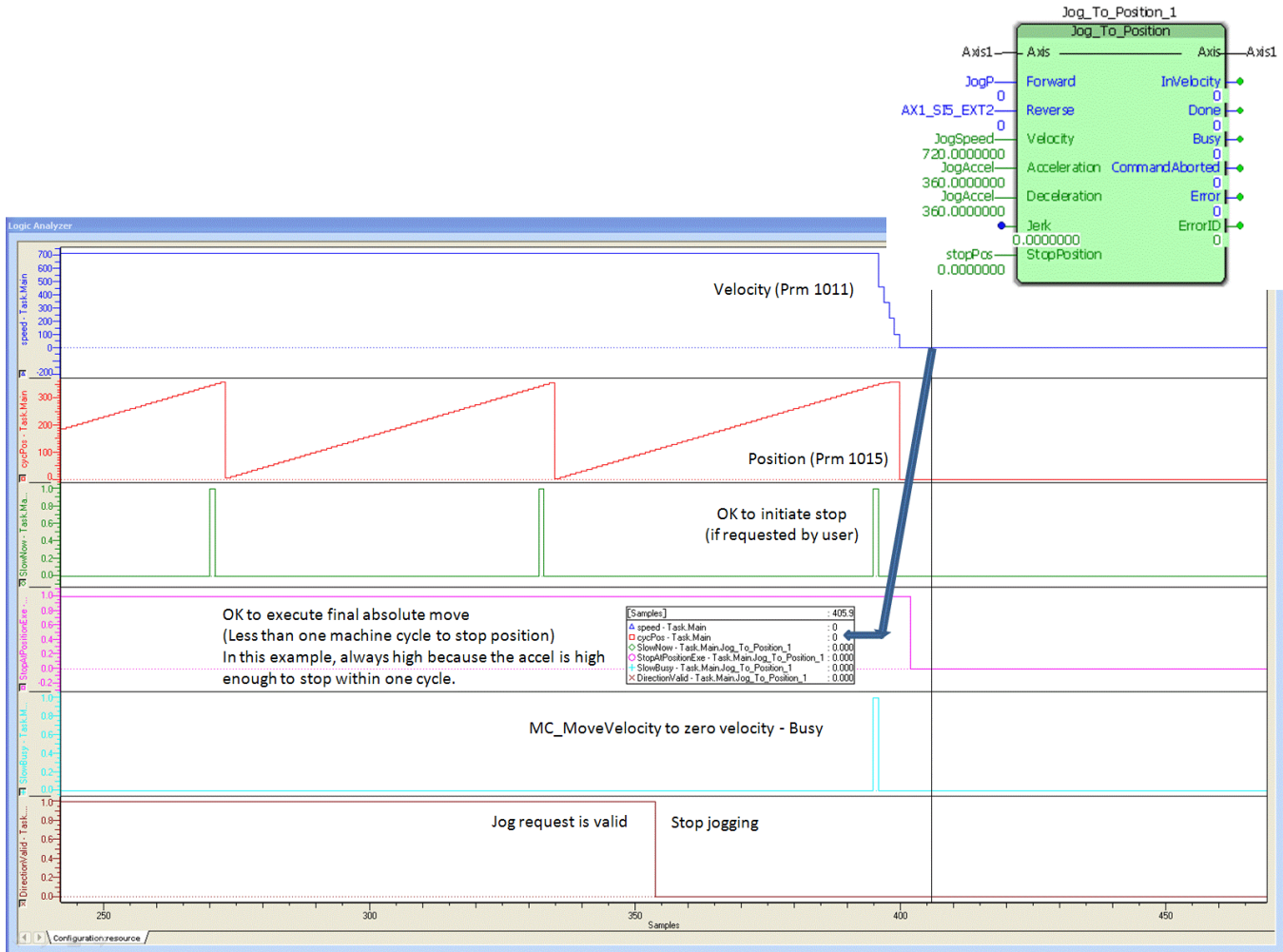


Error Description

See the [Function Block ErrorID](#) list.

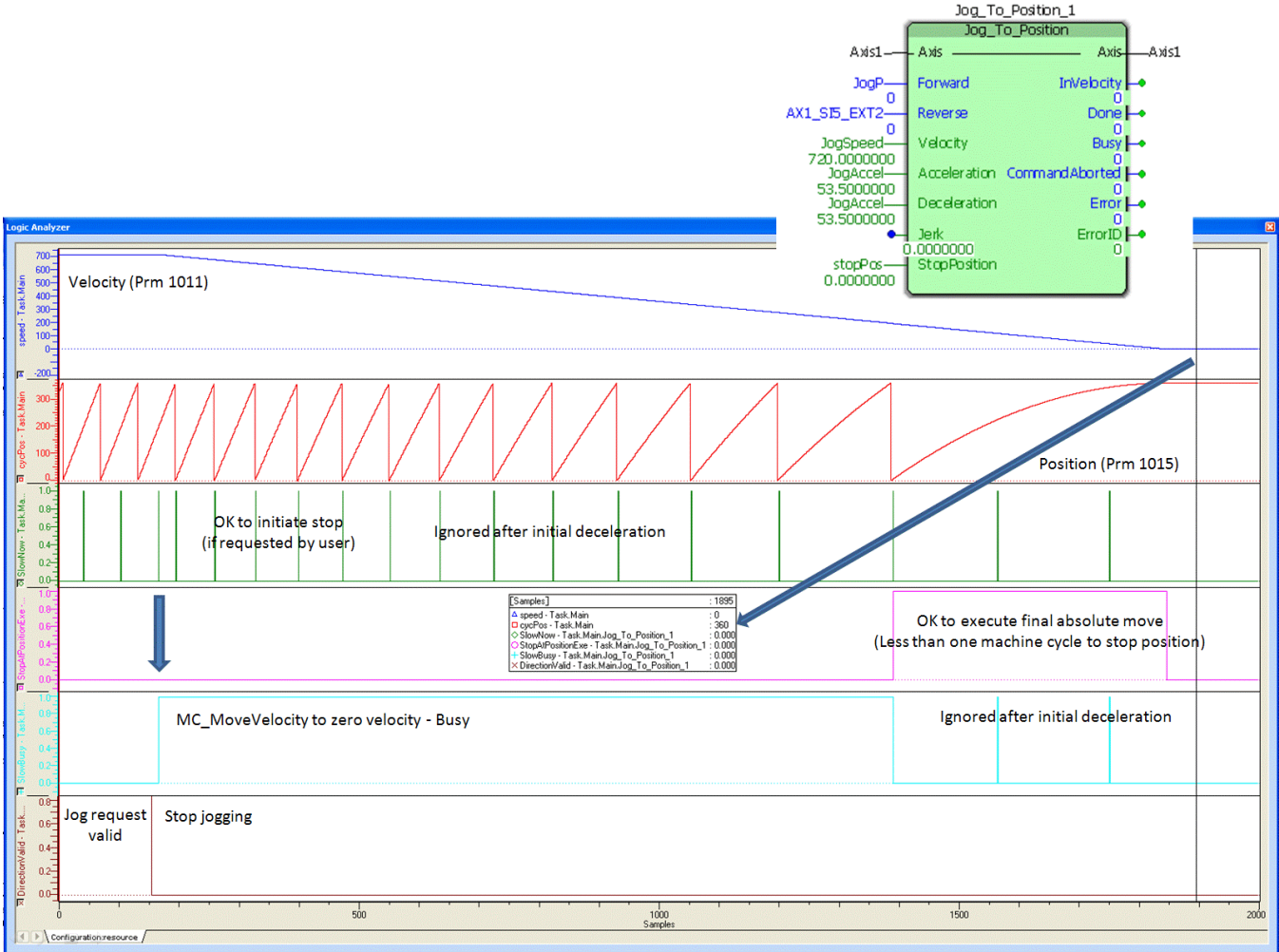
Example 1

In the first example the speed is low enough and the deceleration high enough that the axis can stop within one revolution. This is the easiest condition.



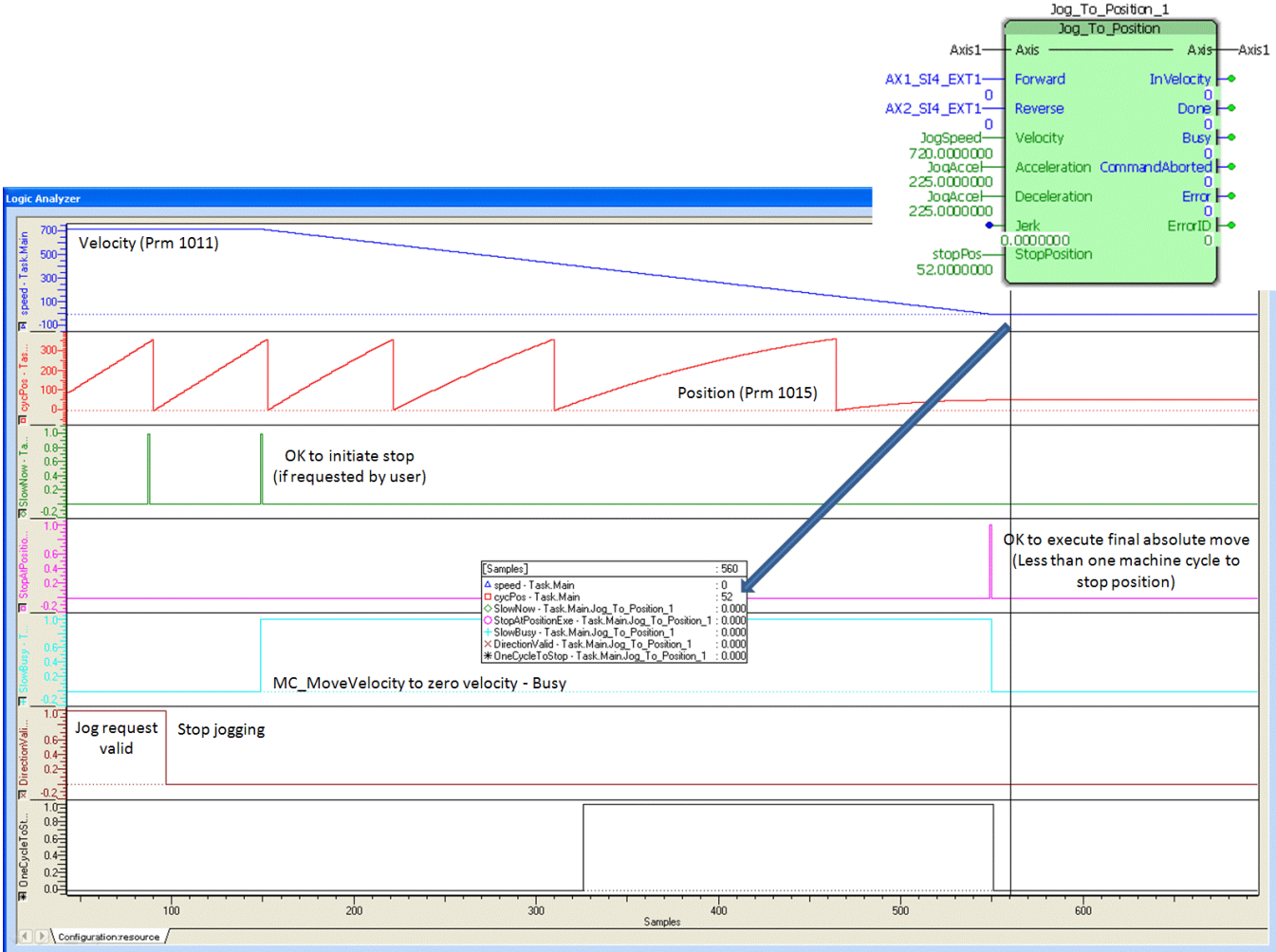
Example 2

In this example, the axis requires about 13 revolutions to come to a stop at the specified velocity and deceleration. The data "SlowNow" in green is an internal monitoring bit which results from a calculation made to determine a position that will allow the motion profile to follow the deceleration rate to the specified StopPosition. Notice there is a very brief delay between the time the Forward jog request is removed and the axis starts decelerating. This allow the axis to decelerate smoothly to the StopPosition. The pink data indicates when the MC_MoveAbsolute is active.



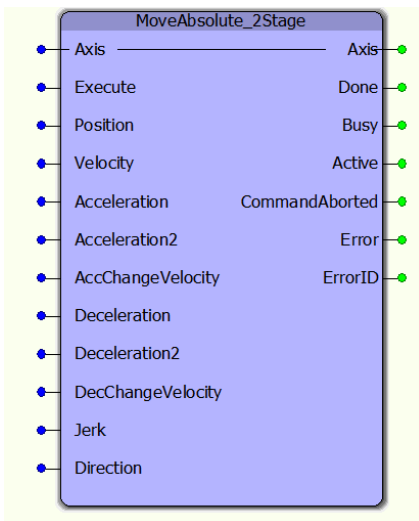
Example 3

The third example shows a deceleration to stop at 52 degrees.





MoveAbsolute_2Stage



This function block commands a move to an absolute position using a two staged acceleration and deceleration.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
B	Position	LREAL	A positive or negative value within the coordinate system in user units.	LREAL#0.0
B	Velocity	LREAL	Absolute value of the velocity in user units/second.	LREAL#0.0

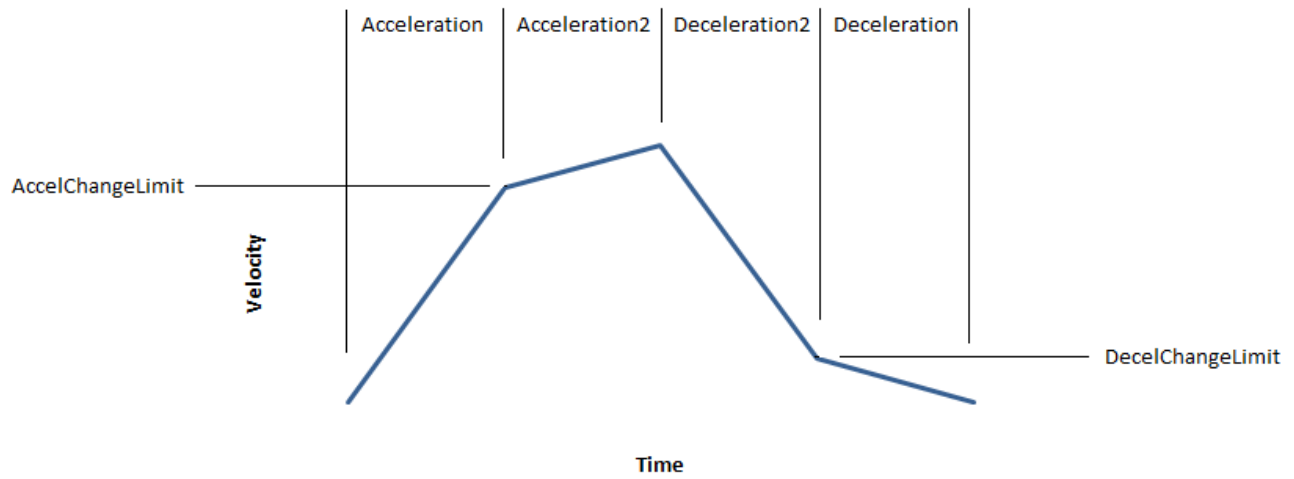
B	Acceleration	LREAL	Value of the acceleration in user unit-s/second ² (acceleration is applicable with same sign of torque and velocity) This acceleration will be applied when Velocity is less than AccChangeVelocity.	LREAL#0.0
B	Acceleration2	LREAL	Value of the acceleration in user unit-s/second ² (acceleration is applicable with same sign of torque and velocity) This acceleration will be applied when Velocity is greater than AccChangeVelocity.	LREAL#0.0
V	AccChangeVelocity	LREAL	Velocity at which motion will transition from using Acceleration to Acceleration2.	LREAL#0.0
B	Deceleration	LREAL	Value of the deceleration in user unit-s/second ² (deceleration is applicable with opposite signs of torque and velocity.) This deceleration will be used when Velocity is less than the DecChangeVelocity.	LREAL#0.0
B	Deceleration2	LREAL	Value of the deceleration in user unit-s/second ² (deceleration is applicable with opposite signs of torque and velocity.) This deceleration will be used when Velocity is greater than the DecChangeVelocity.	LREAL#0.0
V	DecChangeVelocity	LREAL	Velocity at which motion will transition from using Deceleration to Deceleration2.	LREAL#0.0
E	Jerk	LREAL	Not supported; reserved for future use. Value of the jerk in [user units / second ³].	LREAL#0.0
B	Direction	MC_Direction	Specifies the direction of motion. Allowable modes are positive_direction, shortest_way, negative_direction, current_direction.	MC_Direction#Positive_Direction
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Active	BOOL	For buffered modes, this output is set high at the moment the block takes control of the axis. For non buffered modes, the outputs Busy and Active have the same value.	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

Error Description

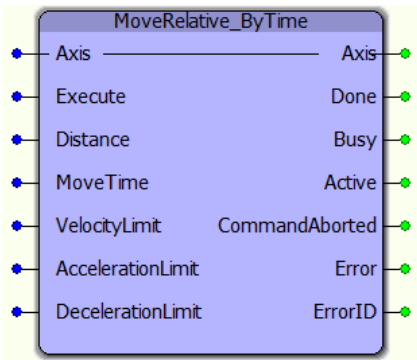
See the [Function Block ErrorID](#) list.

Example





MoveRelative_ByTime



This function block converts the MoveTime input into acceleration, velocity, and deceleration for a 1/3, 1/3, 1/3 trapezoidal move profile which will complete in the MoveTime specified. It uses the MC_MoveRelative function block.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
V	Distance	LREAL	A relative positive or negative value within the coordinate system in user units
V	MoveTime	LREAL	The time required (in seconds) for the move to complete.
V	VelocityLimit	LREAL	Maximum velocity used when moving. If left unconnected, no velocity limit will be applied during move.

V	AccelerationLimit	LREAL	Maximum acceleration used when moving. If left unconnected, no acceleration limit will be applied during move.	LREAL#0.0
V	DecelerationLimit	LREAL	Maximum deceleration used when moving. If left unconnected, no deceleration limit will be applied during move.	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Active	BOOL	For buffered modes, this output is set high at the moment the block takes control of the axis. For non buffered modes, the outputs Busy and Active have the same value.	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

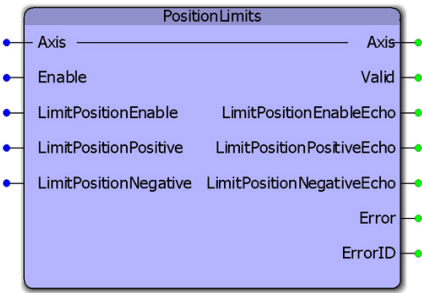
- Prior to v207, this function creates a 1/3, 1/3, 1/3 trapezoidal move, it may not be appropriate for very long moves, because the calculated commanded speed may be too high. New functionality was added for v207 which allows the function to calculate the move parameters to stay within the restraints of the new VAR_INPUTS VelocityLimit, AccelerationLimit, and DecelerationLimit.
- See the [MoveRelative_ByTime eLearning Module](#) on Yaskawa's YouTube channel.

Error Description

See the [Function Block ErrorID](#) list.



PositionLimits



This function block enables or disables the position limit function. It also allows continuous streaming of new position limits. This block uses MC_WriteBoolParameter, MC_ReadBoolParameter, MC_WriteParameter, and MC_ReadParameter.

Library

PLCopen Toolbox

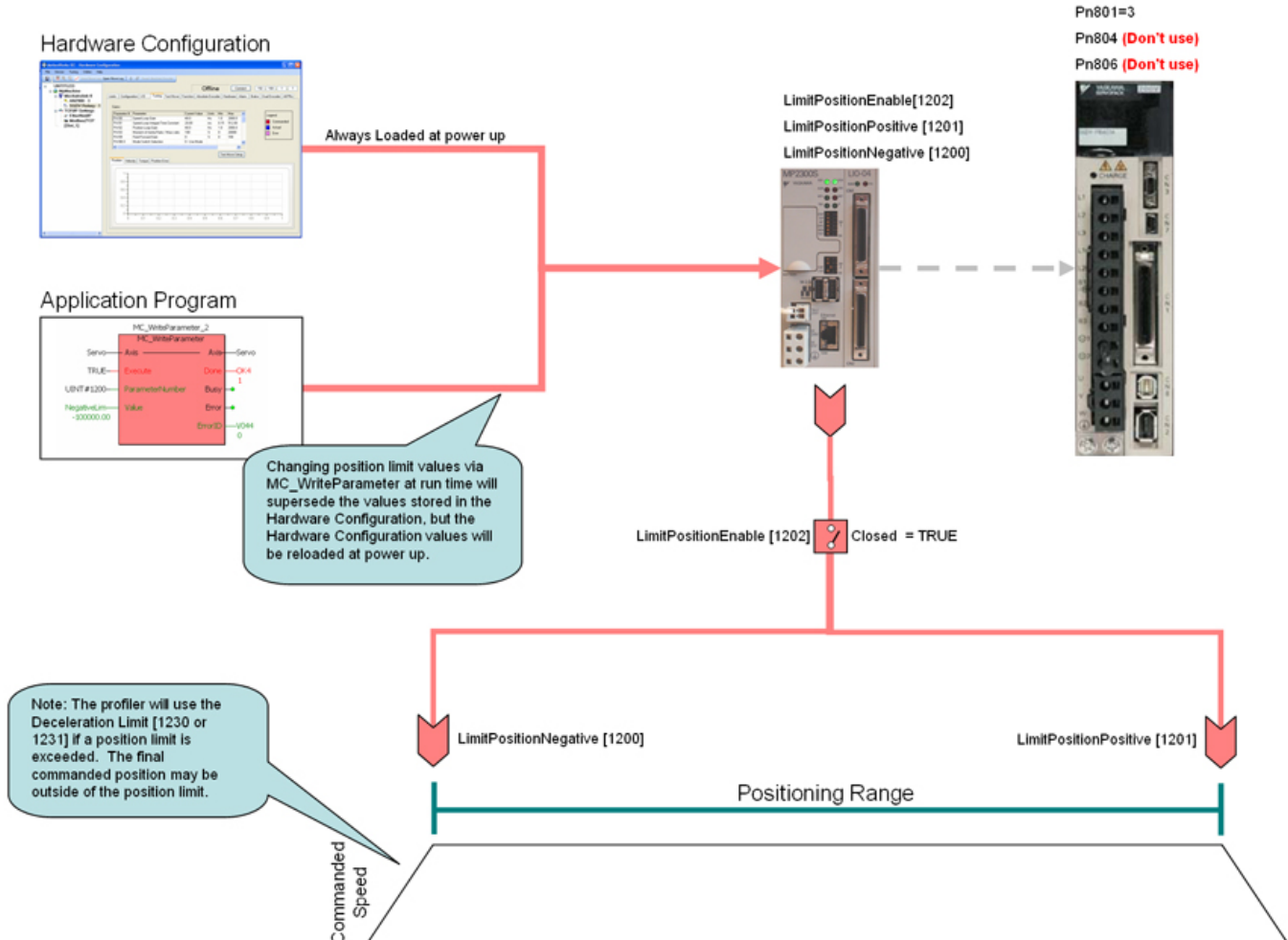
Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	LimitPositionEnable	BOOL	Enables / Disables the position limit function in the motion engine.
V	LimitPositionPositive	LREAL	The maximum commanded position allowed
V	LimitPositionNegative	LREAL	The minimum commanded position allowed
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	LimitPositionEnableEcho	BOOL	Status of the Position Limit function from the motion engine.
V	LimitPositionPositiveEcho	LREAL	Value used by the motion engine for the maximum allowed commanded position.
V	LimitPositionNegativeEcho	LREAL	Value used by the motion engine for the minimum allowed commanded position.

B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

The function block uses MC_ReadBoolParameter, MC_WriteBoolParameter, MC_ReadParameter, and MC_WriteParameter.



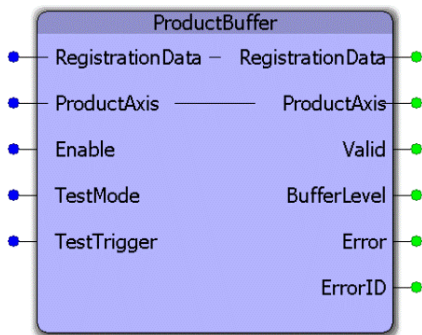
- The software position limits are managed by the MPic controller. The parameters are called LimitPositionPositive and LimitPositionNegative, with values of UINT#1201 and UINT#1200 respectively. Use the MC_WriteParameter function block for these and all controller side parameters. Position limit parameters are in user units.
- When a position limit is exceeded, a controller alarm will be generated, obtainable via the MC_ReadAxisError function block, or the web server.
- The controller alarm will be 16#3202 0001 if the positive position limit is exceeded and 16#3202 0002 if the negative position limit is exceeded.
- To disable the position limits, set LimitPositionEnable, parameter 1202 to zero.
- LimitPositionPositive must be greater than LimitPositionNegative.
- LimitPositionNegative must be lower than LimitPositionPositive.
- See the [PositionLimits eLearning Module](#) on Yaskawa's YouTube channel.

Error Description

See the [Function Block ErrorID](#) list.



ProductBuffer



This function block uses MC_TouchProbe and provides a circular buffer of recorded latch positions for the axis specified. It is tailored for use specifically for applications that process random incoming products such as rotary knives or linear flying shear. Together, the application programmer and the ProductBuffer function block manage the RegistrationData structure which contains information pertaining to the product positions and other mechanical dimensions related to the application.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	ProductAxis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
V	RegistrationData	ProductBufferStruct	Structure containing all information for the circular buffer to operate.	
VAR_INPUT				
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	Default FALSE
V	TestMode	BOOL	If TRUE, then the internal MC_TouchProbe is aborted, and the function block can be used to "dry cycle" the machine by simulating products using the TestTrigger input.	FALSE
V	TestTrigger	BOOL	If TestMode is TRUE, then on the rising edge of TestTrigger, the actual position of the ProductAxis will be stored into the RegistrationData STRUCT.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	

V	BufferLevel	INT	Indicates the number of products in the buffer by subtracting UsePointer from StorePointer.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

- The Math Toolbox v202 or higher is required when using the ProductBuffer. The Math Toolbox must be included and placed above the PLCOpen Toolbox in the Libraries folder of your MotionWorks IEC project.
- The ProductBuffer function block manages only the storing activity and only updates the StorePointer. Another part of your application must update the UsePointer after the products have been processed. If the UsePointer is not updated, this function block will eventually generate the ErrorID 10022, buffer overrun.
- The StorePointer and UsePointer are the 'Head' and the 'Tail' of the circular buffer. If more than one 'Use' of the latch data is required, use the expanded sub structure added for v206 which supports a series of Use pointer activities.
- The cyclic (modularized) and non cyclic (unmodularized) latch position are stored into the RegistrationData simultaneously.
- TestMode can be toggled on the fly without re Enabling the function block. TestMode was added in v201.
- See the [ProductBuffer eLearning Module](#) on Yaskawa's YouTube channel.

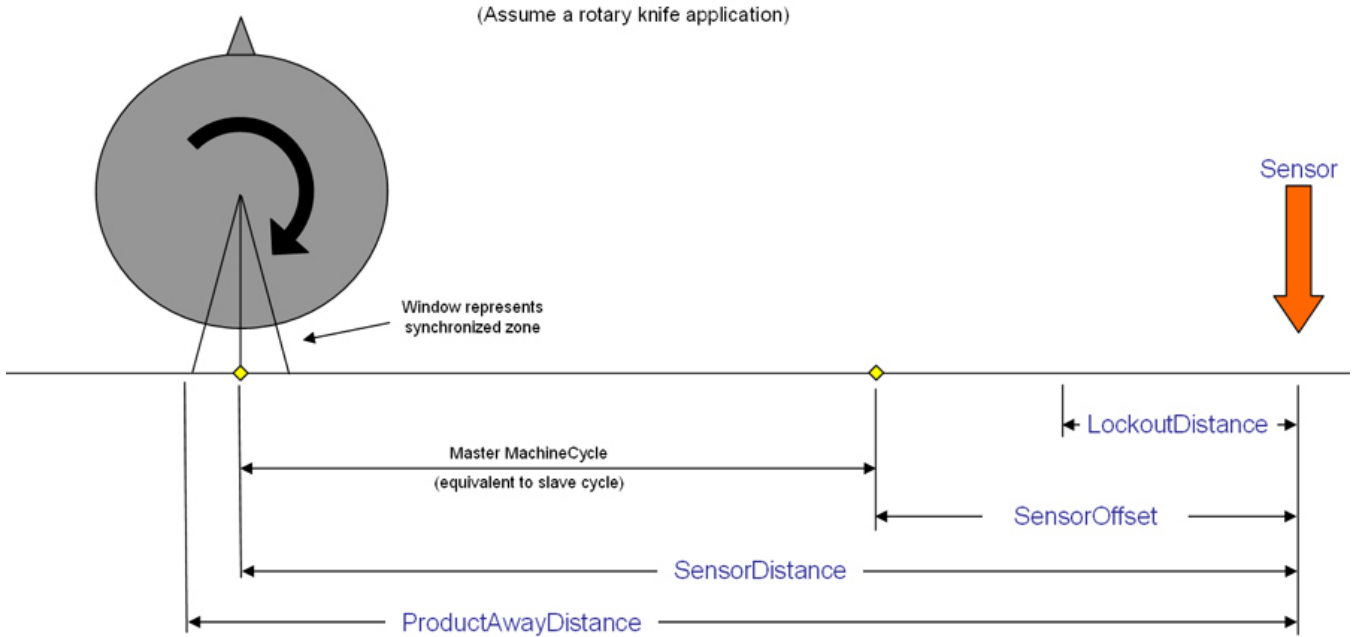
Error Description

See the [Function Block ErrorID](#) list.

Example 1:

Consider a rotary knife application as shown below.

ProductBufferStruct Definitions



(*Initialization of the ProductBufferStruct in an initialize program*)

Conveyor.Products.BufferSize: =INT#20;

Conveyor.Products.LockoutDistance: =LREAL#3.25; (* inches *)

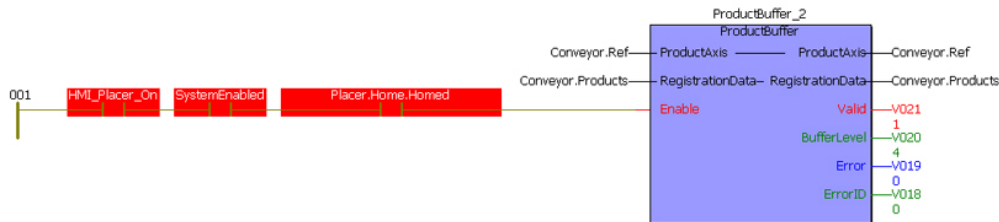
Conveyor.Products.ManualOffset: =LREAL#0.0;

Conveyor.Products.ProductAwayDistance: =LREAL#23.75;

Conveyor.Products.Sensor.Bit: =UINT#1; (* Equates to input1 on 2600 I/O, see MC_TouchProbe help for details *)

Conveyor.Products.SensorDistance: =LREAL#23.25; (* If product leads slave, increase this value *)

Conveyor.Products.SensorOffset: =REM(Conveyor.Products.SensorDistance, Conveyor.MachineCycle);

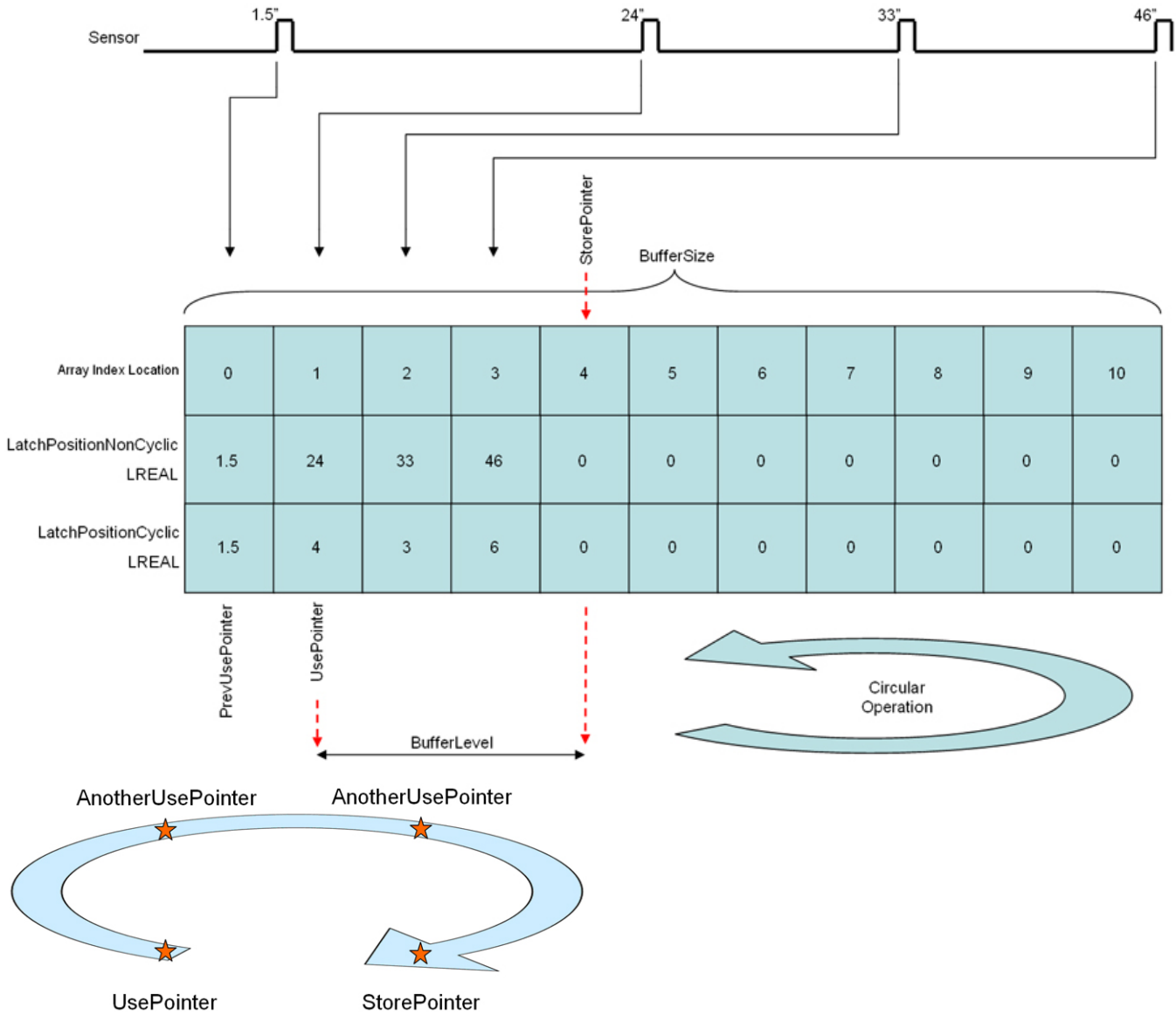


Variable	Value	Default value	Type
Conveyor			ConveyorStruct
Ref			AxIS_REF
Prm			AxisParameterStruct
Products			ProductBufferStruct
BufferSize	20		INT
BufferNonCyclic			LatchBufferArray
BufferCyclic			LatchBufferArray
Sensor			TRIGGER_REF
Input			INPUT_REF
Bit	1		UINT
Pattern	0		INT
ID	0		UINT
SensorDistance	2.3250000E+001		LREAL
SensorOffset	1.2588514E+000		LREAL
ManualOffset	0.0000000E+000		LREAL
FilterDistance	3.2500000E+000		LREAL
ProductAwayDistance	2.3750000E+001		LREAL
StorePointer	19		INT
UsePointer	16		INT
PrevUsePointer	15		INT

Variable	Value	Default value	Type
BufferNonCyclic			LatchBufferArray
[0]	7.0217149E+005		LREAL
[1]	7.0217666E+005		LREAL
[2]	7.0203970E+005		LREAL
[3]	7.0204402E+005		LREAL
[4]	7.0205855E+005		LREAL
[5]	7.0206436E+005		LREAL
[6]	7.0207238E+005		LREAL
[7]	7.0207649E+005		LREAL
[8]	7.0208167E+005		LREAL
[9]	7.0209183E+005		LREAL
[10]	7.0209664E+005		LREAL
[11]	7.0210632E+005		LREAL
[12]	7.0211436E+005		LREAL
[13]	7.0211861E+005		LREAL
[14]	7.0212569E+005		LREAL
[15]	7.0212982E+005		LREAL
[16]	7.0213470E+005		LREAL
[17]	7.0215034E+005		LREAL
[18]	7.0216219E+005		LREAL
[19]	7.0216738E+005		LREAL
[20]	0.0000000E+000		LREAL

ProductBuffer Operation

(Assume a 10" Machine Cycle)



Example 2:

The configuration shown below is for a system which is used to detect rising and falling edge triggers for a product moving along a conveyor driven by a servo. the rising edge detection signal is wired to the EXT1 terminal of the ServoPack. The falling edge signal is wired to the EXT2 terminal of the ServoPack.

```

(*ProductBufferStruct for Registration Data *)
(*=====*)
20 Products.BufferSize           := INT#20;
90.0000000 Products.LockoutDistance := LREAL#90.0;
1580.0000000 Products.SensorDistance := LREAL#1580.0;

(*Products.Sensor.Bit := UINI#1; *)
1 Products.BufferPattern[0].Bit := UINI#1;
2 Products.BufferPattern[1].Bit := UINI#2;
2 Products.BufferPatternSize := INT#2;

(*Products.ProductAwayDistance := LREAL#1730.0;*)
1730.0000000 Products.PatternAwayDistance[0] := LREAL#1730.0;
1930.0000000 Products.PatternAwayDistance[1] := LREAL#1930.0;

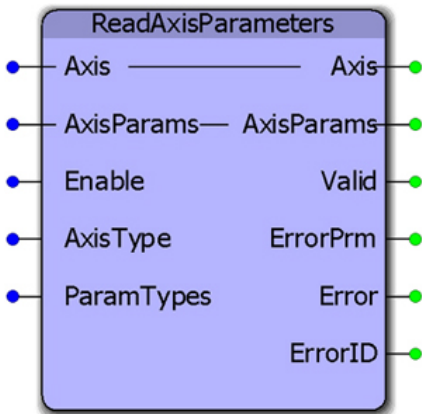
```

Products.BufferedPattern shows the trigger sequence in which latched data was captured.

+	BufferNonCyclic	
+	BufferCyclic	
-	BufferedPattern	
	[0]	1
	[1]	2
	[2]	1
	[3]	2
	[4]	1
	[5]	2
	[6]	1
	[7]	2
	[8]	1
	[9]	2
	[10]	1
	[11]	2
	[12]	1
	[13]	2
	[14]	1
	[15]	2
	[16]	1
	[17]	2
	[18]	0



ReadAxisParameters



This function block reads all axis parameters into the AxisParameterStruct. The Y_Motion firmware library must be included in a project that uses ReadAxisParameters.

Library

PLCopen Toolbox

Parameters

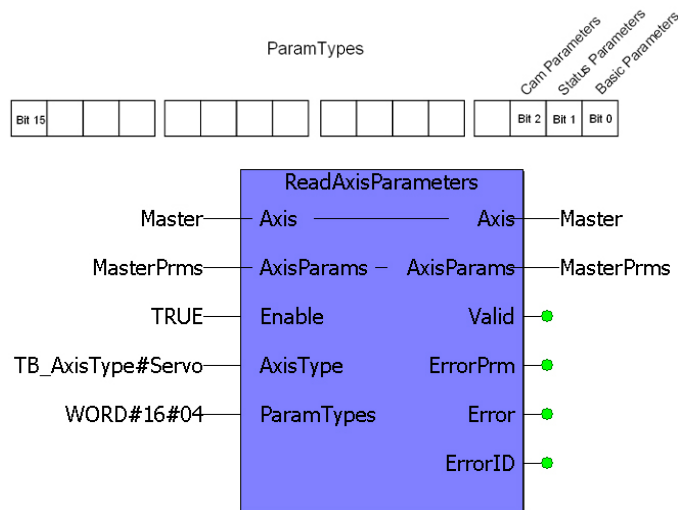
*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
V	AxisParams	AxisParameterStruct	User Defined DataType declared in the PLCopen Toolbox.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors. Inputs will be read only on the rising edge of enable.	FALSE
V	AxisType	TB_AxisType	Indicates axis type: TB_AxisType#Servo TB_AxisType#VFD TB_AxisType#Stepper TB_AxisType#Virtual TB_AxisType#External	INT#0 (TB_AxisType#Servo)

V	ParamTypes	WORD	Used to include additional parameter sets, such as camming.	WORD#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	ErrorPrm	UINT	If there was an error while attempting to read one of the parameters listed in the ParamStruct, this output will contain the offending parameter number.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

Only AxisType#Servo, AxisType#External, AxisType#Virtual are supported.

By default, the function will update all parameter types in the AxisParamStruct. For efficiency, parameters are grouped into types. Basic, Status, and Cam. For axes that are not cam slaves, there is no need to read the cam parameters. To cause the function to skip the update of a parameter group, set the corresponding bit high. For example, the following function block will not read the cam parameters:



Parameters categorized as BasicMotion are always read.

ParamType	ParameterName	Parameter #
BasicMotion	ActualPosition	1000
BasicMotion	ActualPositionCyclic	1005
BasicMotion	ActualPositionNonCyclic	1006
BasicMotion	ActualTorque	1004
BasicMotion	ActualVelocity	1001
BasicMotion	AtVelocity	1141
BasicMotion	CommandedPosition	1010
BasicMotion	CommandedPositionCyclic	1015
BasicMotion	CommandedPositionNonCyclic	1016
BasicMotion	CommandedTorque	1014
BasicMotion	CommandedVelocity	1011
BasicMotion	InPosition	1140
BasicMotion	LatchPositionNonCyclic	1031
BasicMotion	PositionError	1130
Cam	CamMasterCycle	1512
Cam	CamMasterPosition	1500
Cam	CamMasterScale	1510

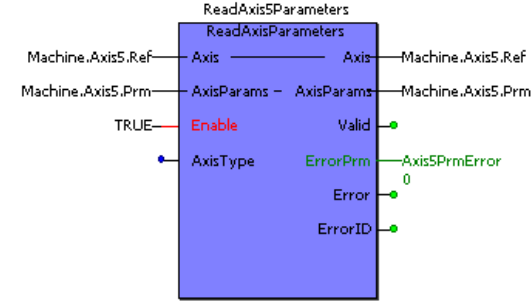
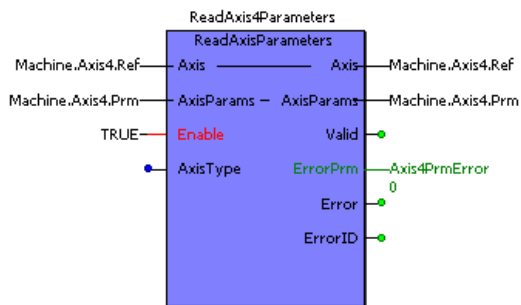
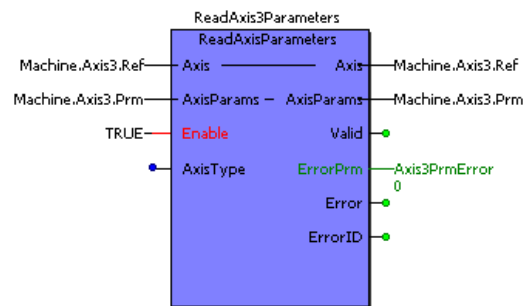
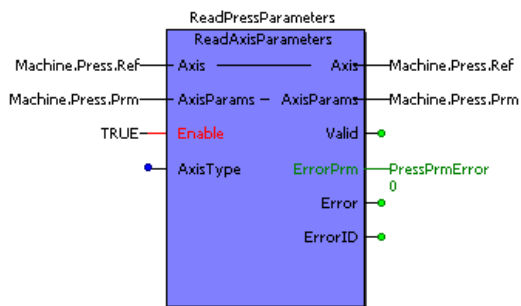
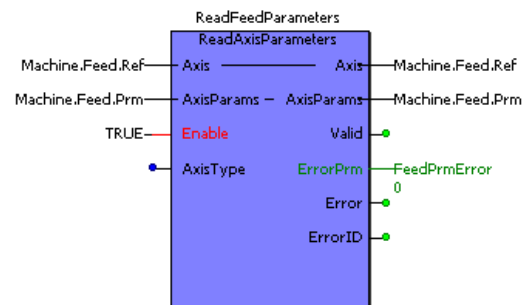
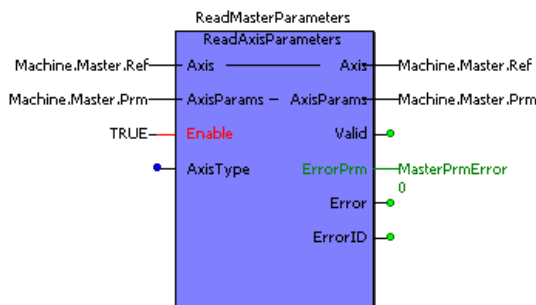
Cam	CamMasterShift	1511
Cam	CamMasterShiftedCyclic	1502
Cam	CamMasterShiftedPosition	1501
Cam	CamOffset	1531
Cam	CamScale	1530
Cam	CamShiftRemaining	1513
Cam	CamState	1540
Cam	CamTableIDEngaged	1541
Cam	CamTableOutput	1520
Status	BufferedMotionBlocks	1600
Status	CommandedAcceleration	1012
Status	PositionWindow	1120

- See the [ReadAxisParameters eLearning Module](#) on Yaskawa's YouTube channel.

Error Description

See the [Function Block ErrorID](#) list.

Example



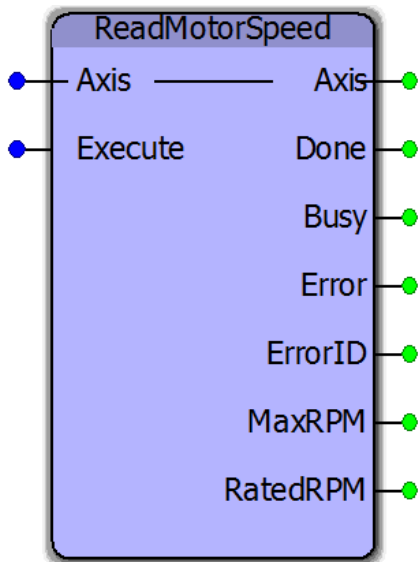


Variable	Value	Type	Instance
Machine.Master.Prm		AxisParameterStruct	Configuration.Resource.Task.Monitor.Machine.Master.Prm
ActualPosition	1467.48	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualPosition
ActualPositionCyclic	1467.48	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualPositionCyclic
ActualPositionNonCyclic	1467.48	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualPositionNonCyclic
ActualTorque	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualTorque
ActualVelocity	60.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualVelocity
AtVelocity	FALSE	BOOL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.AtVelocity
BufferedMotionBlocks	1.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.BufferedMotionBlocks
CamMasterCycle	1.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterCycle
CamMasterPosition	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterPosition
CamMasterShiftedCyclic	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterShiftedCyclic
CamMasterShiftedPosition	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterShiftedPosition
CamMasterScale	100.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterScale
CamMasterShift	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterShift
CamOffset	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamOffset
CamScale	100.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamScale
CamShiftRemaining	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamShiftRemaining
CamState	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamState
CamTableDEngaged	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamTableDEngaged
CamTableOutput	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamTableOutput
CommandedAcceleration	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CommandedAcceleration
CommandedPosition	1467.60	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CommandedPosition
CommandedPositionCyclic	1467.60	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CommandedPositionCyclic
CommandedPositionNonCyclic	1467.60	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CommandedPositionNonCyclic
CommandedTorque	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CommandedTorque
CommandedVelocity	60.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CommandedVelocity
InPosition	FALSE	BOOL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.InPosition
LatchPositionNonCyclic	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.LatchPositionNonCyclic
PositionError	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.PositionError

◀ ▶ Watch 1 / Watch 2 \ Watch 3 \ Watch 4 /



ReadMotorSpeed



This function block reads the rated and peak speeds of a Sigma-5 motor connected to the controller.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
			Default FALSE
VAR_OUTPUT			
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.

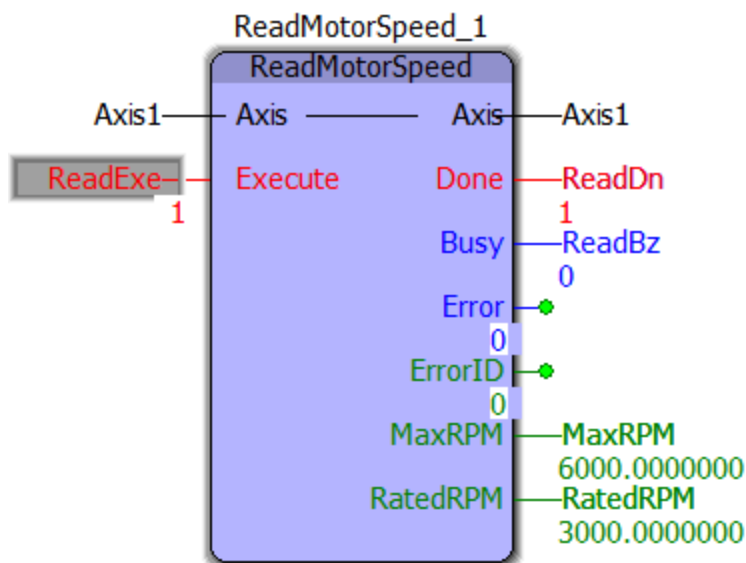
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
E	MaxRPM	LREAL	Peak speed of the motor
E	RatedRPM	LREAL	Rated speed of the motor

Error Description

See the [Function Block ErrorID](#) list.

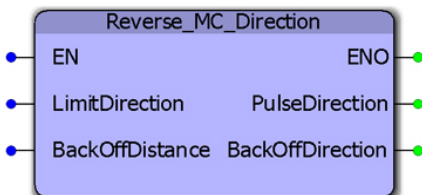
Example

The example below shows the ReadMotorSpeed function block reading the rated and peak speeds of an SGMAV motor.





Reverse_MC_Direction



This function block was designed for use with the [Home_LS_Pulse](#) function block in the PLCopen Toolbox. It changes the enumerated type MC_Direction#positive_direction to MC_Direction#negative_direction or vice versa so that the function can move the motor one direction into a limit switch with MC_StepRefLimit, and the other direction when searching for the Index Pulse with MC_StepRefPulse.

Library

PLCopen Toolbox

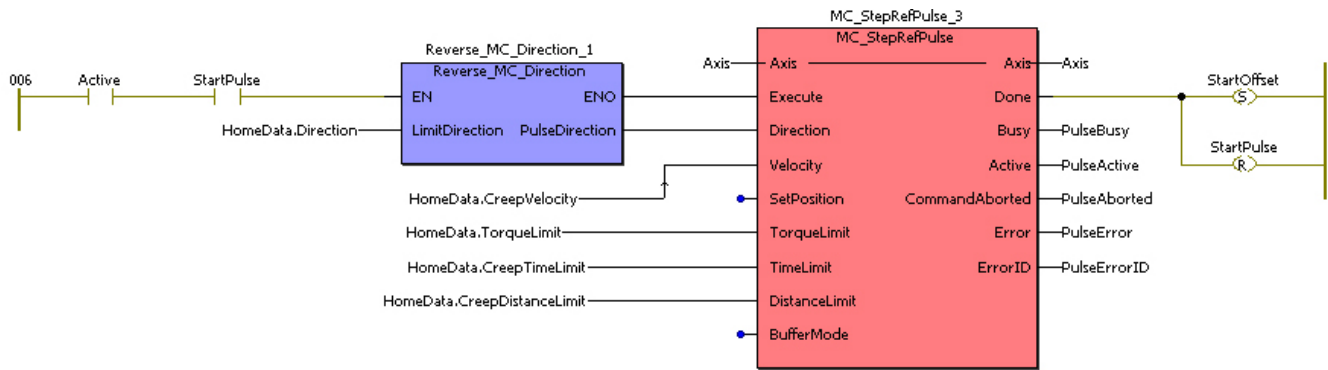
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	EN	BOOL	Enables the function.	FALSE
V	LimitDirection	INT	INT / ENUM MC_Direction#positive_direction or MC_Direction#negative_direction	
V	BackOffDistance	LREAL		INT #0
VAR_OUTPUT				
B	ENO	BOOL	High if the function is executing normally.	
V	PulseDirection	INT	INT / ENUM MC_Direction#positive_direction or MC_Direction#negative_direction	
V	BackOffDirection	LREAL		

Error Description

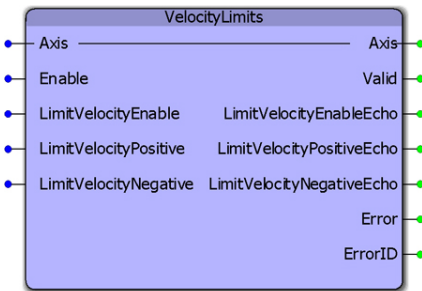
No Errors will result, but if there is a problem with the ENUM input for MC_Direction, then ENO will be FALSE.

Example





VelocityLimits



This function block enables or disables the velocity limit function. It also allows continuous streaming of new velocity limits. This block uses MC_WriteBoolParameter, MC_ReadBoolParameter, MC_WriteParameter, and MC_ReadParameter.

Library

PLCopen Toolbox

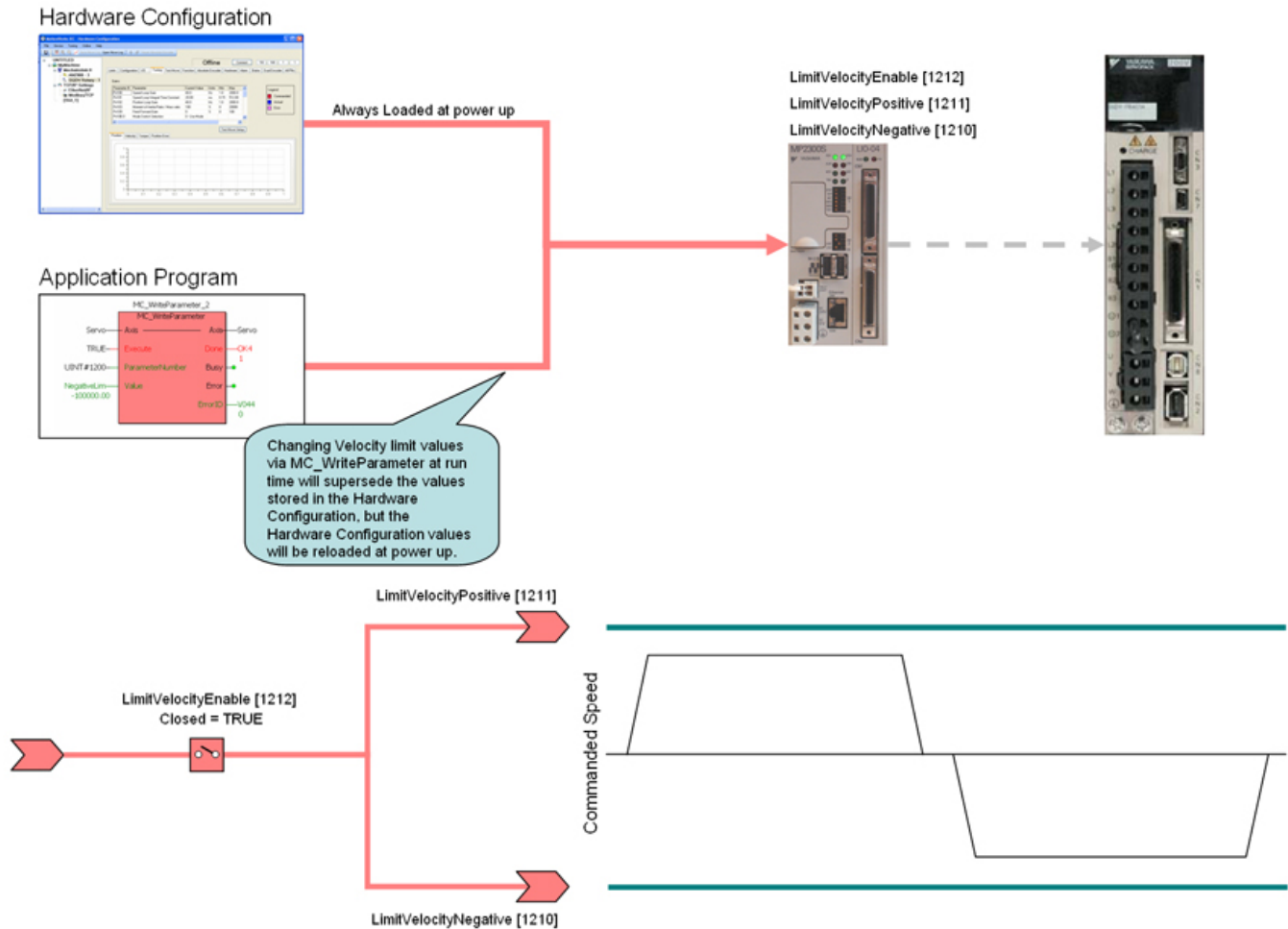
Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	LimitVelocityEnable	BOOL	Enables / Disables the velocity limit function in the motion engine.
V	LimitVelocityPositive	LREAL	The maximum commanded velocity allowed
V	LimitVelocityNegative	LREAL	The minimum commanded velocity allowed
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	LimitPositionEnableEcho	BOOL	Status of the Velocity Limit function from the motion engine.
V	LimitPositionPositiveEcho	LREAL	Value used by the motion engine for the maximum allowed commanded velocity.
V	LimitPositionNegativeEcho	LREAL	Value used by the motion engine for the minimum allowed commanded velocity.

B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

The function block uses MC_ReadBoolParameter, MC_WriteBoolParameter, MC_ReadParameter, and MC_WriteParameter.



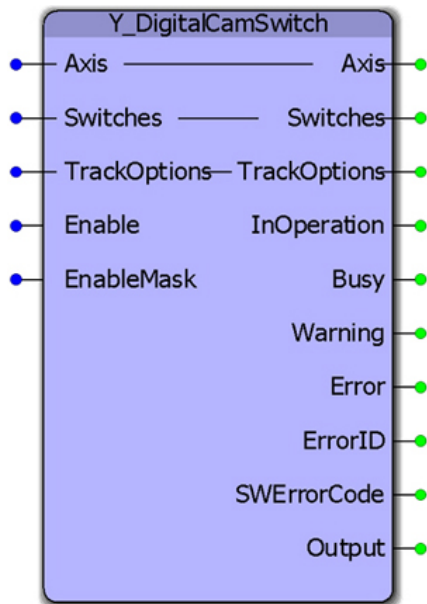
- The software velocity limits are managed by the MPieci controller. The parameters are called LimitVelocityPositive and LimitVelocityNegative, with values of UINT#1211 and UINT#1210 respectively. Use the MC_WriteParameter function block for these and all controller side parameters. Velocity limit parameters are in user units / sec.
- When a velocity limit is exceeded, a controller alarm will be generated, obtainable via the MC_ReadAxisError function block, or the web server.
- The controller alarm will be 16#3202 0003 if the positive velocity limit is exceeded and 16#3202 0004 if the negative velocity limit is exceeded.
- To disable the velocity limits, set LimitVelocityEnable, parameter 1212 to zero.
- LimitVelocityPositive must be zero or greater.
- LimitVelocityNegative must be zero or lower.

Error Description

See the [Function Block ErrorID](#) list.



Y_DigitalCamSwitch



This function block commands a group of discrete output bits analogous to a set of mechanical cam controlled switches driven by a rotating shaft. Forward and backward movements are allowed. A maximum of 32 outputs and 256 switches are supported. Starting in v340, support was added to include the FT62, a specialized high speed output capability of the Sigma 7 ServoPack.

Library

PLCopen Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).
B	Switches	CAMSWITCH_REF	Reference to the switching actions. 256 maximum switches.
E	TrackOptions	TRACK_REF	Reference to the track related properties. 32 maximum tracks.
VAR_INPUT			Default
B	Enable	BOOL	The function will continue to execute while enable is held high. FALSE

E	EnableMask	DWORD	Individually enables the tracks [0..31] per the bit pattern. Value of 1 means Enabled, 0 means disabled. Least significant bit corresponds to Track [0]. Default if not connected is All Tracks Enabled.	DWORD#0
VAR_OUTPUT				
B	InOperation	BOOL	Function Block Enable is ON and at least 1 track is enabled (EnableMask is <> 0).	
B	Busy	BOOL	Function Block Enable is ON but no tracks are enabled (EnableMask = 0).	
E	Warning	BOOL	Signals that a non-critical error has occurred within the function block. In this case, the block will continue to function.	
E	SWErrorCode	SWERROR_STRUCT	Switch Error Code Structure that identifies particular warnings with switch settings. The user can monitor this ErrorCode if Warning output comes on.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
E	Output	DWORD	Resulting CamSwitch output for each track per the bit pattern. Least significant bit corresponds to Track [0]. This Output will need to be tied to physical outputs outside of the DigitalCamSwitch FB.	

Notes

- This functionality is sometimes called PLS – Phase or Position or Programmable Limit Switch.
- Switches will be evaluated for both forward and reverse travel of the axis.
- OnCompensation and OffCompensation will only be applied when the axis is moving in the Positive Direction.
- Track Hysteresis is not supported.
- When changing TrackNumber and reverse signal of any switch, Sigma7_FT62 feature requires controller reboot.
- The function block 'Output' is not supported by Sigma7_FT62 and not output bit can be observed for this feature.
- Add the Yaskawa Toolbox to the project above the PLCopen Toolbox. This is required because some datatypes used by this function block are declared in that library, and compile errors will occur if the datatypes are not defined.

Restrictions

If the output specified in the PLS is also controlled somewhere else in the project then the last instruction wins. This would also be the case when a single output is used in two PLS blocks.

The PLS block will support a maximum of 256 switches and 32 outputs. This means that the block will react to a maximum of 512 positions (two for each switch).

If the cam-like lobes of multiple switches intersect with each other for a single track the net effect would be an OR-ing of the switches.

Example1 SW1: on at 10, off at 50, SW2: on at 20, off at 30; net effect on at 10 off at 50.

Example2 SW1: on at 10, off at 50, SW2: on at 40, off at 60; net effect on at 10 off at 60.

Operation

On the rising edge of Enable, the input data will be checked against restrictions. The busy output will remain on until at least 1 track is enabled and the FB is controlling the outputs, then the InOperation bit will be set and the busy bit reset.

While the Enable is on, the EnableMask value will be read each scan and effect the output control.

On the falling edge of Enable, all outputs will be reset (turn off), and the InOperation, Busy, and Error bits will be reset. ErrorID output will be set to 0.

Input Data that is read only on rising edge of Enable:

CAMSWITCH_STRUCT[].TrackNumber

CAMSWITCH_STRUCT[].AxisDirection
CAMSWITCH_STRUCT[].CamSwitchMode
AXIS_REF
CAMSWITCH_REF.MasterType
CAMSWITCH_REF.MachineCycle
CAMSWITCH_REF.LastSwitch

Input Data that is read continuously while Enabled:

CAMSWITCH_STRUCT[].FirstOnPosition
CAMSWITCH_STRUCT[].LastOnPosition
CAMSWITCH_STRUCT[].Duration
CAMSWITCH_STRUCT[].FirstOnPosition
TRACK_STRUCT[].OnCompensationScaler
TRACK_STRUCT[].OffCompensationScaler

Enable

EnableMask

Output Bits: Boolean Outputs are exclusive

AxisDirection must be 0, any other number will default to 0. (values 1 and 2 not supported.)

CamSwitchMode must be 0 or 1, any other number will default to 0.

Error Description

See the [Function Block ErrorID](#) list.

Example 1:

Consider the PLS requirement shown in the figure below. There are 4 tracks (0, 1, 2, 3) in the set up and a total of 5 switches (0, 1, 2, 3, 4).

Track 0 has 2 switches associated with it.

Switch 0: On Position : 2 degrees

Off Position : 10 degrees

Switch 1: On Position : 200 degrees

Off Position : 210 degrees

Track 1 has 1 switch associated with it

Switch 2: On Position : 20 degrees

Off Position : 30 degrees

Track 2 has 1 switch associated with it

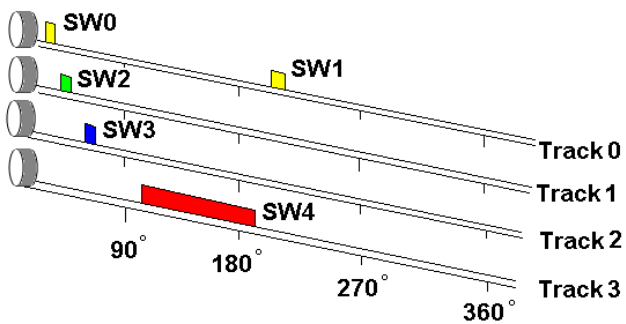
Switch 3: On Position : 50 degrees

Off Position : 60 degrees

Track 3 has 1 switch associated with it

Switch 4: On Position : 100 degrees

Off Position : 200 degrees



The switches can be defined and initialized as follows:

```

(*PLS initialization*)

4   PLS_Switches.LastSwitch :=INT#4;
360.00000000 PLS_Switches.MachineCycle :=LREAL#360.0;
0   PLS_Switches.MasterType :=INT#0;

0   PLS_Switches.Switch[INT#0].TrackNumber := 0;
0   PLS_Switches.Switch[INT#0].AxisDirection := INT#0;
0   PLS_Switches.Switch[INT#0].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0   PLS_Switches.Switch[INT#0].Duration := DINT#0;
2.00000000 PLS_Switches.Switch[0].FirstOnPosition := LREAL#2.0;
10.00000000 PLS_Switches.Switch[0].LastOnPosition := LREAL#10.0;

0   PLS_Switches.Switch[INT#1].TrackNumber := 0;
0   PLS_Switches.Switch[INT#1].AxisDirection := INT#0;
0   PLS_Switches.Switch[INT#1].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0   PLS_Switches.Switch[INT#1].Duration := DINT#0;
200.00000000 PLS_Switches.Switch[1].FirstOnPosition := LREAL#200.0;
210.00000000 PLS_Switches.Switch[1].LastOnPosition := LREAL#210.0;

1   PLS_Switches.Switch[INT#2].TrackNumber := 1;
0   PLS_Switches.Switch[INT#2].AxisDirection := INT#0;
0   PLS_Switches.Switch[INT#2].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0   PLS_Switches.Switch[INT#2].Duration := DINT#0;
20.00000000 PLS_Switches.Switch[2].FirstOnPosition := LREAL#20.0;
30.00000000 PLS_Switches.Switch[2].LastOnPosition := LREAL#30.0;

2   PLS_Switches.Switch[INT#3].TrackNumber := 2;
0   PLS_Switches.Switch[INT#3].AxisDirection := INT#0;
0   PLS_Switches.Switch[INT#3].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0   PLS_Switches.Switch[INT#3].Duration := DINT#0;
50.00000000 PLS_Switches.Switch[3].FirstOnPosition := LREAL#50.0;
60.00000000 PLS_Switches.Switch[3].LastOnPosition := LREAL#60.0;

3   PLS_Switches.Switch[INT#4].TrackNumber := 3;
0   PLS_Switches.Switch[INT#4].AxisDirection := INT#0;
0   PLS_Switches.Switch[INT#4].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0   PLS_Switches.Switch[INT#4].Duration := DINT#0;
100.00000000 PLS_Switches.Switch[4].FirstOnPosition := LREAL#100.0;
200.00000000 PLS_Switches.Switch[4].LastOnPosition := LREAL#200.0;

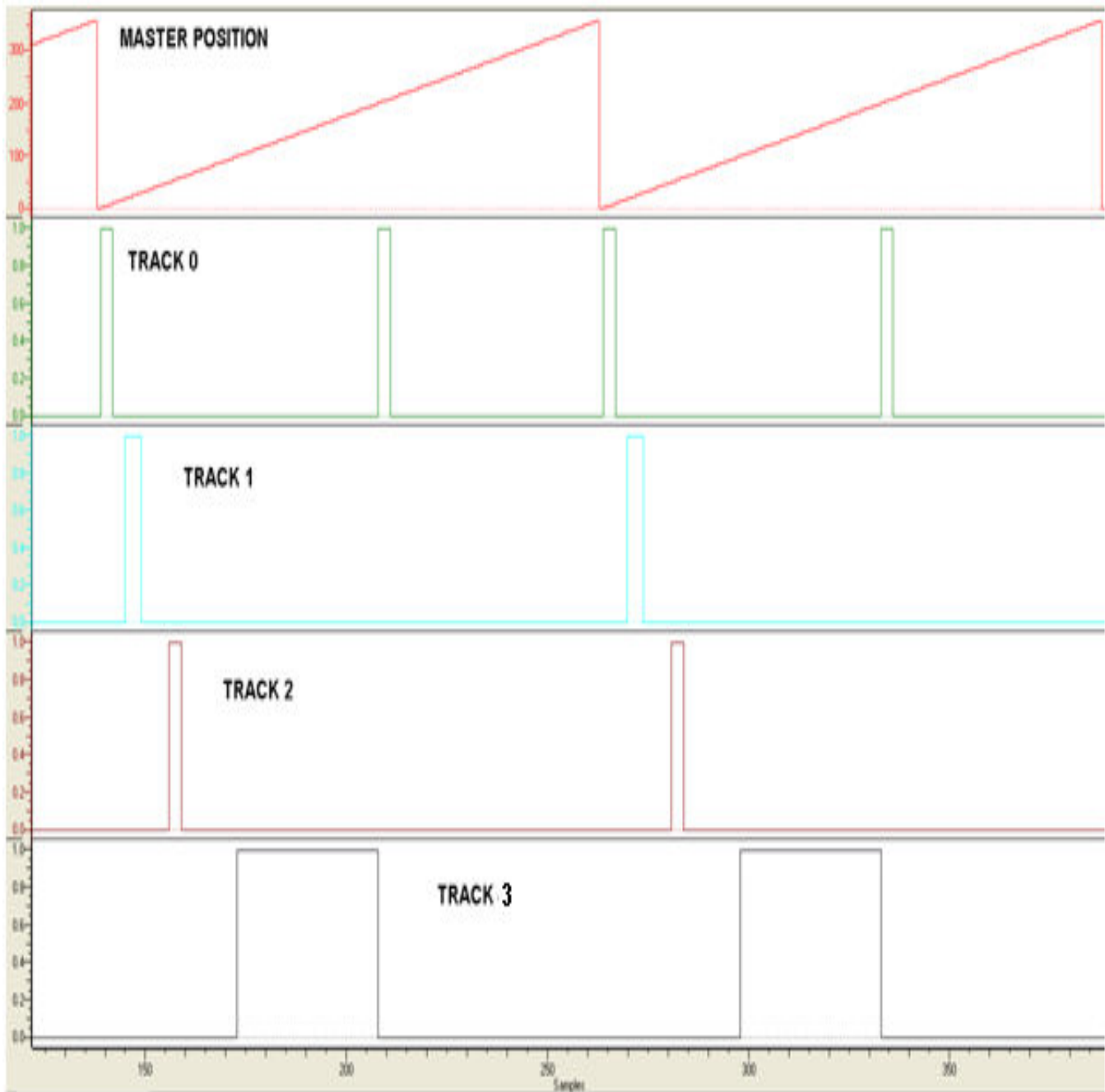
```



The image shows a 'Variable Properties' dialog box with a blue title bar. An arrow points from the first line of code in the previous block to the 'Name' field. The dialog contains the following fields:

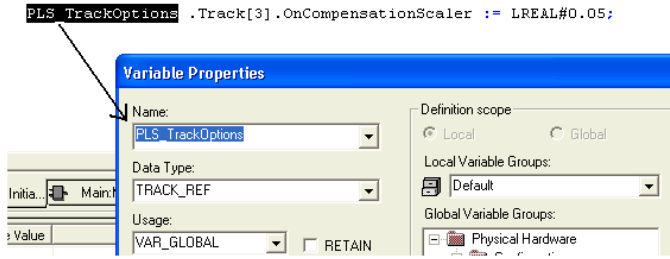
- Name:** A dropdown menu showing 'PLS_Switches'.
- Data Type:** A dropdown menu showing 'CAMSWITCH_REF'.
- Usage:** A dropdown menu showing 'VAR_GLOBAL' and a checkbox labeled 'RETAIN' which is currently unchecked.

Once the Y_DigitalCamSwitch is enabled and is in operation, the track output states will be as shown in the logic analyzer plot given below. Note that the outputs will correspond to the position of the axis.



Example 2:

If speed compensation needs to be applied to individual tracks, it can be accomplished by specifying either OnCompensationScaler or OffCompensationScaler in the TRACK_REF data type (TrackOptions in Y_DigitalCamSwitch). An example of applying a -0.06 OffCompensation on track 1 and 0.05 OnCompensation on track 3 is shown below.




```

(*PLS initialization*)
4      PLS_Switches.LastSwitch :=INT#4;
360.00000000 PLS_Switches.MachineCycle :=LREAL#360.0;
0      PLS_Switches.MasterType :=INT#0;

0      PLS_Switches.Switch[INT#0].TrackNumber := 0;
0      PLS_Switches.Switch[INT#0].AxisDirection := INT#0;
0      PLS_Switches.Switch[INT#0].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0      PLS_Switches.Switch[INT#0].Duration := DINT#0;
2.00000000 PLS_Switches.Switch[0].FirstOnPosition := LREAL#2.0;
10.00000000 PLS_Switches.Switch[0].LastOnPosition := LREAL#10.0;

0      PLS_Switches.Switch[INT#1].TrackNumber := 0;
0      PLS_Switches.Switch[INT#1].AxisDirection := INT#0;
0      PLS_Switches.Switch[INT#1].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0      PLS_Switches.Switch[INT#1].Duration := DINT#0;
200.00000000 PLS_Switches.Switch[1].FirstOnPosition := LREAL#200.0;
210.00000000 PLS_Switches.Switch[1].LastOnPosition := LREAL#210.0;

1      PLS_Switches.Switch[INT#2].TrackNumber := 1;
0      PLS_Switches.Switch[INT#2].AxisDirection := INT#0;
0      PLS_Switches.Switch[INT#2].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0      PLS_Switches.Switch[INT#2].Duration := DINT#0;
20.00000000 PLS_Switches.Switch[2].FirstOnPosition := LREAL#20.0;
30.00000000 PLS_Switches.Switch[2].LastOnPosition := LREAL#30.0;

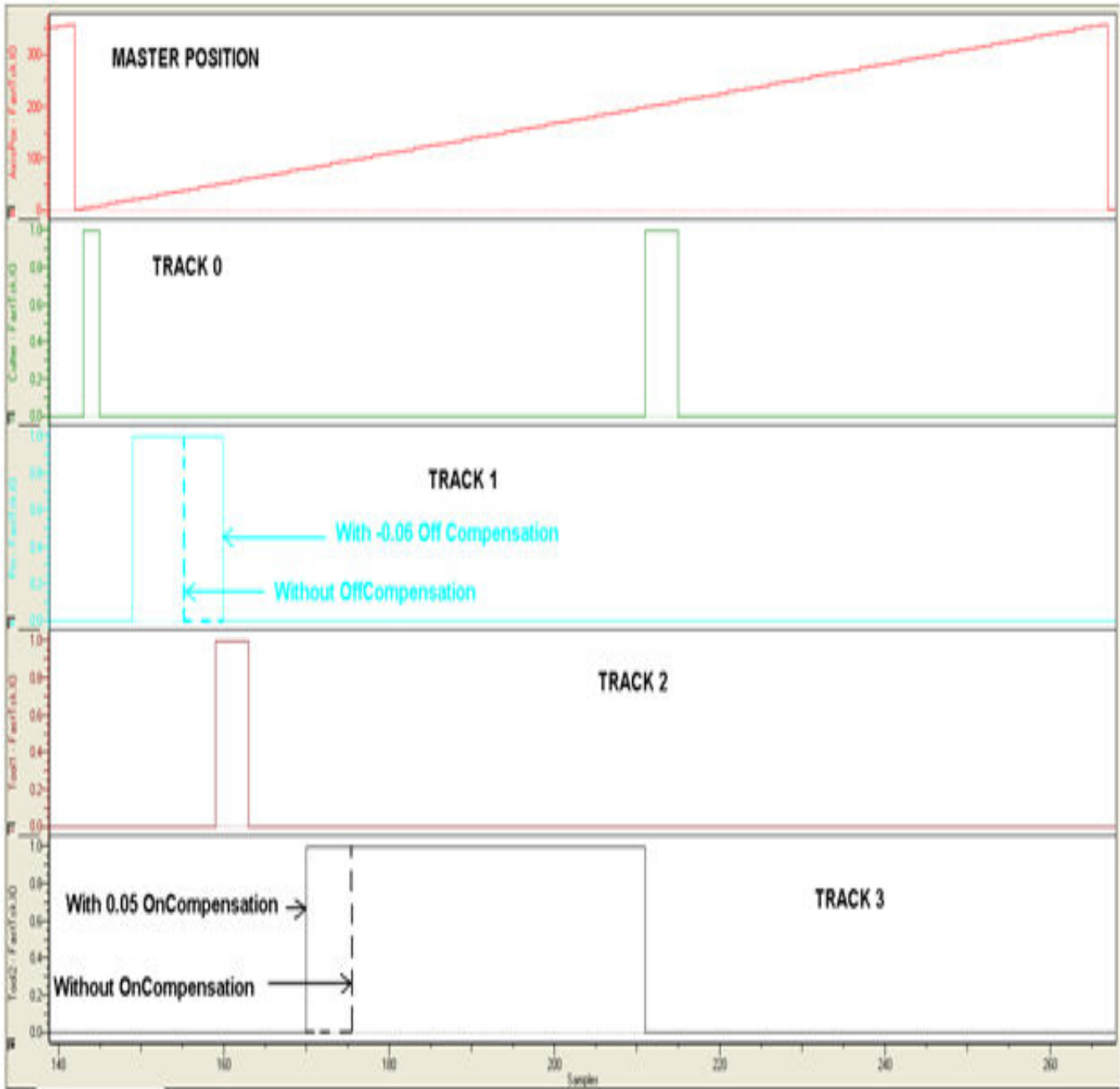
-0.06000000 PLS_TrackOptions.Track[1].OffCompensationScaler := LREAL#-0.06;

2      PLS_Switches.Switch[INT#3].TrackNumber := 2;
0      PLS_Switches.Switch[INT#3].AxisDirection := INT#0;
0      PLS_Switches.Switch[INT#3].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0      PLS_Switches.Switch[INT#3].Duration := DINT#0;
50.00000000 PLS_Switches.Switch[3].FirstOnPosition := LREAL#50.0;
60.00000000 PLS_Switches.Switch[3].LastOnPosition := LREAL#60.0;

3      PLS_Switches.Switch[INT#4].TrackNumber := 3;
0      PLS_Switches.Switch[INT#4].AxisDirection := INT#0;
0      PLS_Switches.Switch[INT#4].CamSwitchMode := INT#0; (* 0: Position, 1: Time *)
0      PLS_Switches.Switch[INT#4].Duration := DINT#0;
100.00000000 PLS_Switches.Switch[4].FirstOnPosition := LREAL#100.0;
200.00000000 PLS_Switches.Switch[4].LastOnPosition := LREAL#200.0;

0.05000000 PLS_TrackOptions.Track[3].OnCompensationScaler := LREAL#0.05;

```



Example 3(Sigma7_FT62):

Relationships between Y_DigitalCamSwitch and Sigma7-FT62 required variables:

Y_DigitalCamSwitch	Sigma7_FT62
PLS_Switches.LastSwitch	N/A
PLS_Switches.MachineCycle	N/A
PLS_Switches.MasterType	N/A
PLS_Switches.Switch[0].TrackNumber	OutputFunction
PLS_Switches.Switch[0].AxisDirection	OutputFunction
PLS_Switches.Switch[0].CamSwitchMode	Pn660
PLS_Switches.Switch[0].Duration	OutputTime
PLS_Switches.Switch[0].FirstOnPosition	OutputPosition
PLS_Switches.Switch[0].LastOnPosition	OutputPosition/OutputDistance
PLS_TrackOptions.Track[0].OnCompensationScaler	OutputPositionCompensation
PLS_TrackOptions.Track[0].OffCompensationScaler	N/A
PLS_TrackOptions.Track[0].Value	N/A



PTB_Initialize

This is not a function block but a Program POU in the PLCopen Toolbox. Its purpose is to reduce the time required to enter initialization code into your project. If you use the provided datatypes, time can be saved by copying and pasting structured text code from this POU into your Initialization POU, then replacing the string "Replace_Me" with another name meaningful to the application.

This POU is not intended to be selected for execution in a task in your application program.



Getting Started with Winding Toolbox

Requirements for v352

To use the Winding Toolbox, your project must also contain the following:

Firmware libraries:

- YMotion

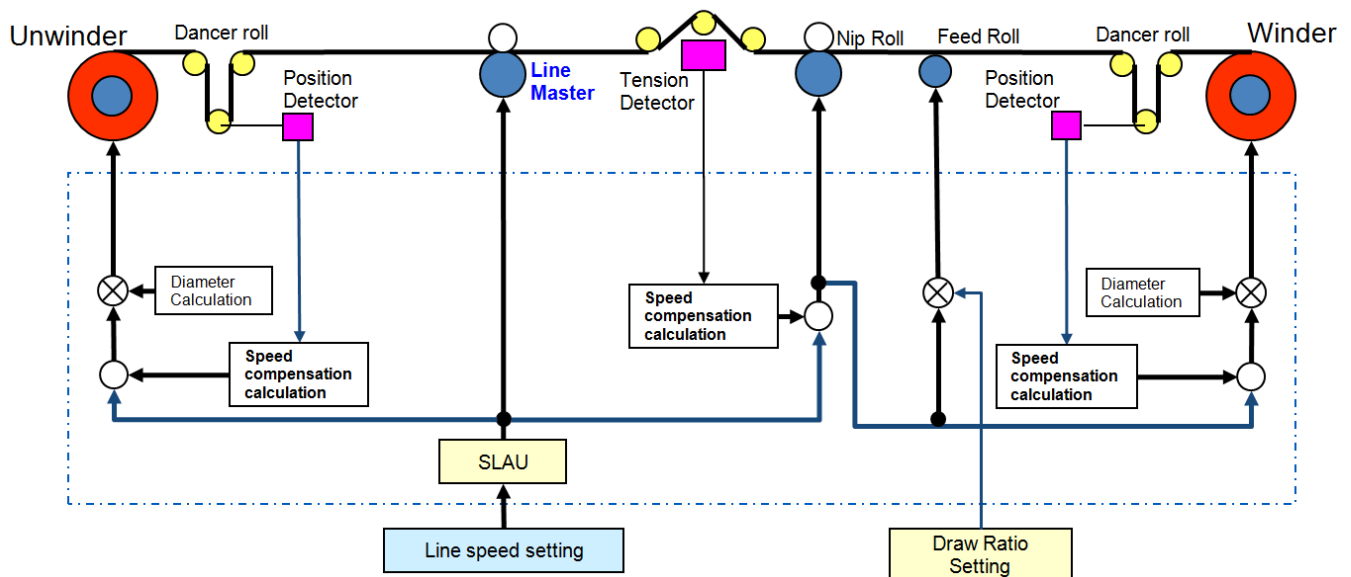
User libraries:

The following User Libraries must be listed above the Winding Toolbox and in the following order:

- DataTypes_Toolbox (v300 or higher)
- Yaskawa_Toolbox (v352 or higher)

Block Diagram

This diagram shows how a complete system can be controlled using the various functions contained in the winding toolbox.





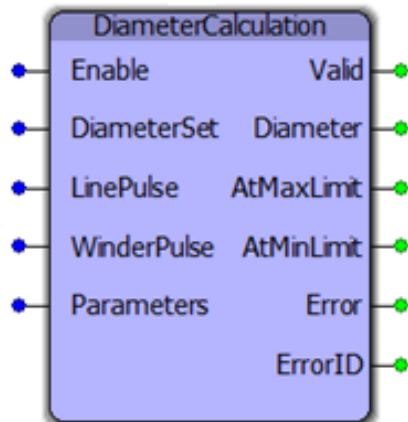
Winding Revision History

Current (first) Version:

2017-11-01 v340 created using 3.2.0 firmware



DiameterCalculation



The DiameterCalculation function block provides a diameter for a winding roll by calculating the diameter from a line pulse and the winder pulse. It updates based on the revolutions incurred at the winding or unwinding roll. As the diameter increases, the speed of the motor decreases to maintain a constant web speed across the system. Likewise as the diameter decreases, the motor speed increases to maintain a constant web speed.

Library

Winding Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	DiameterSet	BOOL	The function will be on when current diameter is set.	FALSE
V	LinePulse	LREAL	Set position feedback from encoder on the line.	LREAL#0.0
V	WinderPulse	LREAL	Feedback from winder axis.	LREAL#0.0
V	Parameters	DiameterStruct	Structure containing all the information for Diameter Calculation function block to operate.	All zeros in structure
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	Diameter	LREAL	The diameter that the DiameterCalculation FB is calculating.	

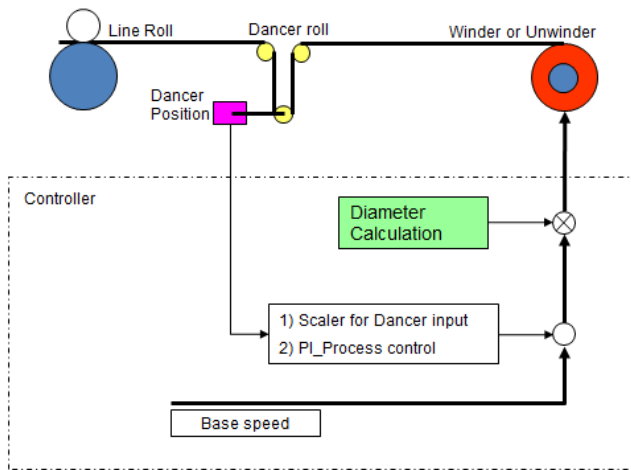
V	AtMaxLimit	BOOL	High if the maximum limit is met.
V	AtMinLimit	BOOL	High if the minimum limit is met.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

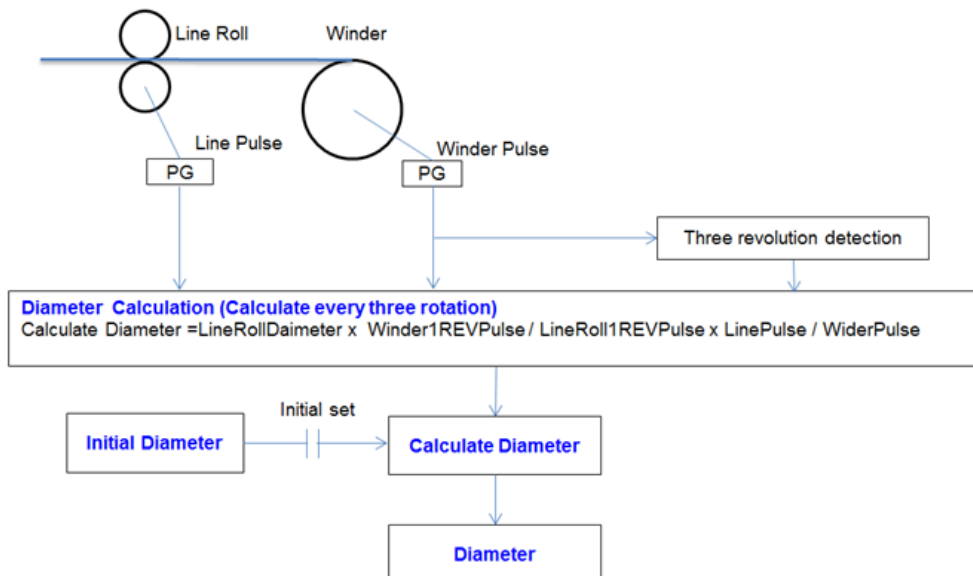
Error Description

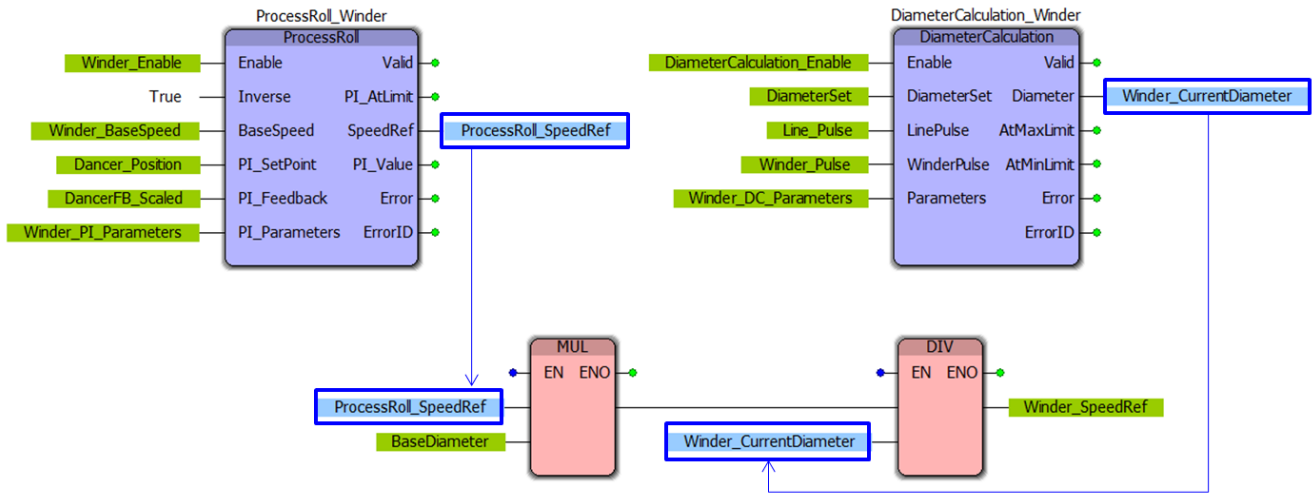
See the [Function Block ErrorID](#) list.

Example



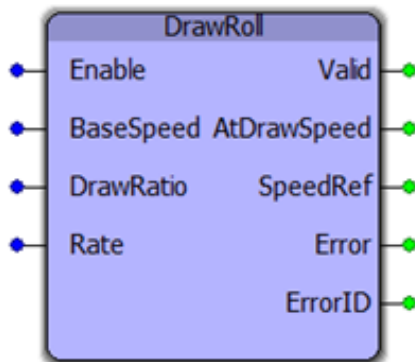
Set the initial value of the diameter before executing DiameterCalculation. An initial diameter is maintained until three rotations after an operation start, then the diameter is updated every three rotations.







DrawRoll



The DrawRoll function block provides the velocity for a draw roll with draw ratio. This method stabilizes a moving web by controlling tension. Based off of a ratio to the line master, the DrawRoll optimizes the web that is fed into a cut roll.

Library

Winding Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	BaseSpeed	LREAL	The BaseSpeed input is a base speed.	LREAL#0.0
V	DrawRatio	LREAL	The ratio for the BaseSpeed. For example 0.01 is 101% & -0.01 is 99% of base speed.	LREAL#0.0
V	Rate	LREAL	Acceleration/Deceleration per scan. The time required for the SpeedRef to become the DrawSpeed value profile depends on the Rate Input and the interval (Application task rate) at which the DrawRoll function block is being run. (User Unit/sec)	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	AtDrawSpeed	BOOL	The AtDrawSpeed output will be high once the BaseSpeed is met.	
V	SpeedRef	BOOL	This output provides the real time speed that the DrawRoll FB is moving.	

B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

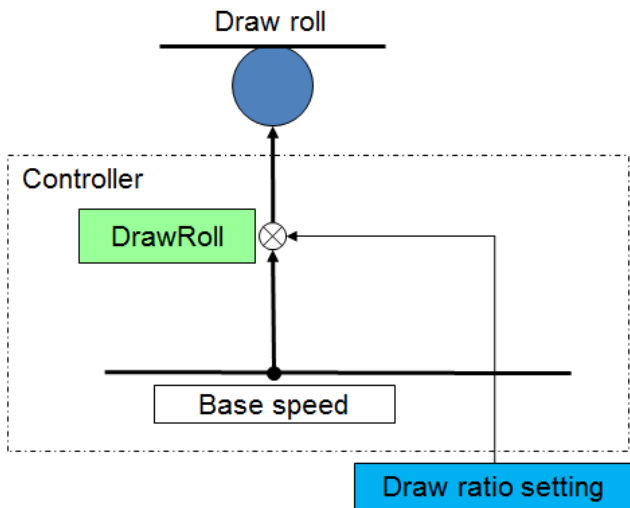
Notes

- The DrawRoll speed is calculated by multiplying the base speed by the draw ratio speed. This helps to provide a constant tension along the entire web.

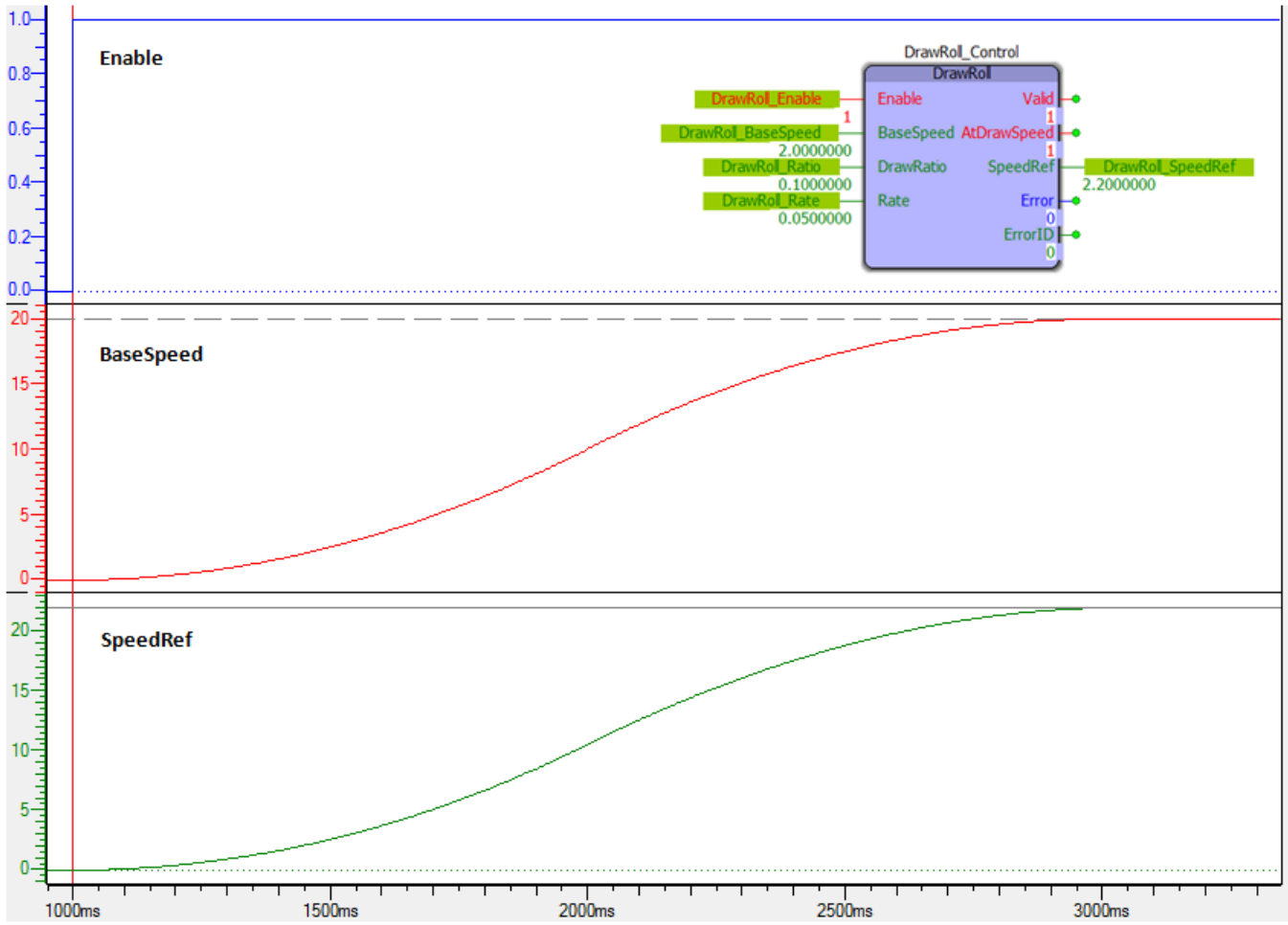
Error Description

See the [Function Block ErrorID](#) list.

Example

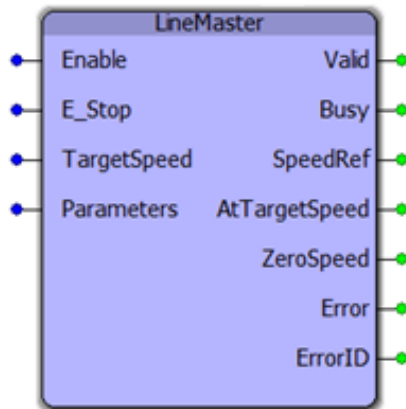


A BaseSpeed is converted to a SpeedRef that is multiplied the DrawRatio. The rate of 1 unit specifies that the output value is expected to increase at the rate of 1 unit per application scan.





LineMaster



The LineMaster function block provides the velocity for a line master with S curve acceleration and deceleration.

Library

Winding Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	E_Stop	LREAL	From the safety circuit of the machine.	LREAL#0.0
V	TargetSpeed	LREAL	Commanded speed of the Line Master in user units.	LREAL#0.0
V	Parameters	LineMasterStruct	The LineMasterStruct parameters are used for initialization of the LineMaster FB.	All zeros in structure
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
V	SpeedRef	LREAL	This is the actual speed of the Line Master.	

V	AtTargetSpeed	BOOL	This output turns on once the Line Master has reached the TargetSpeed input.
V	ZeroSpeed	BOOL	This output is used for starting up the machine.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

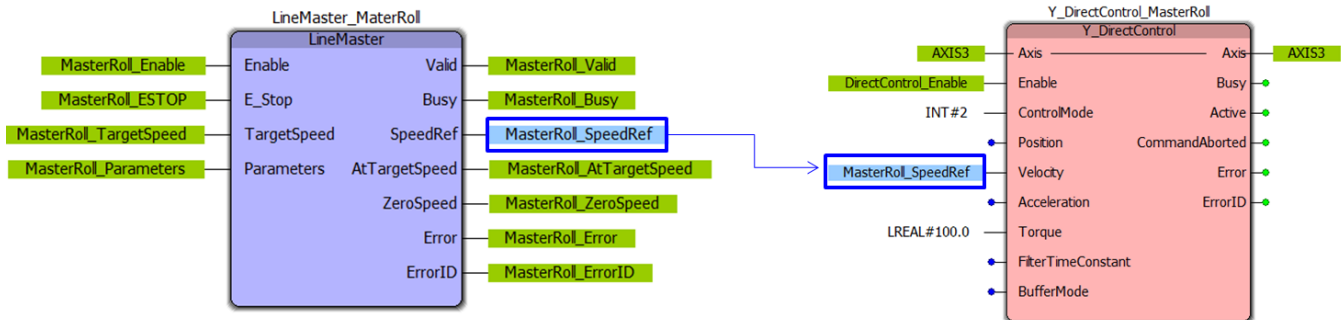
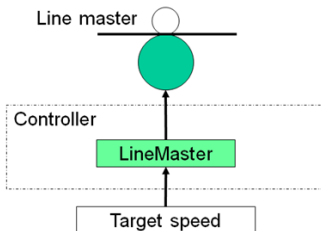
Notes

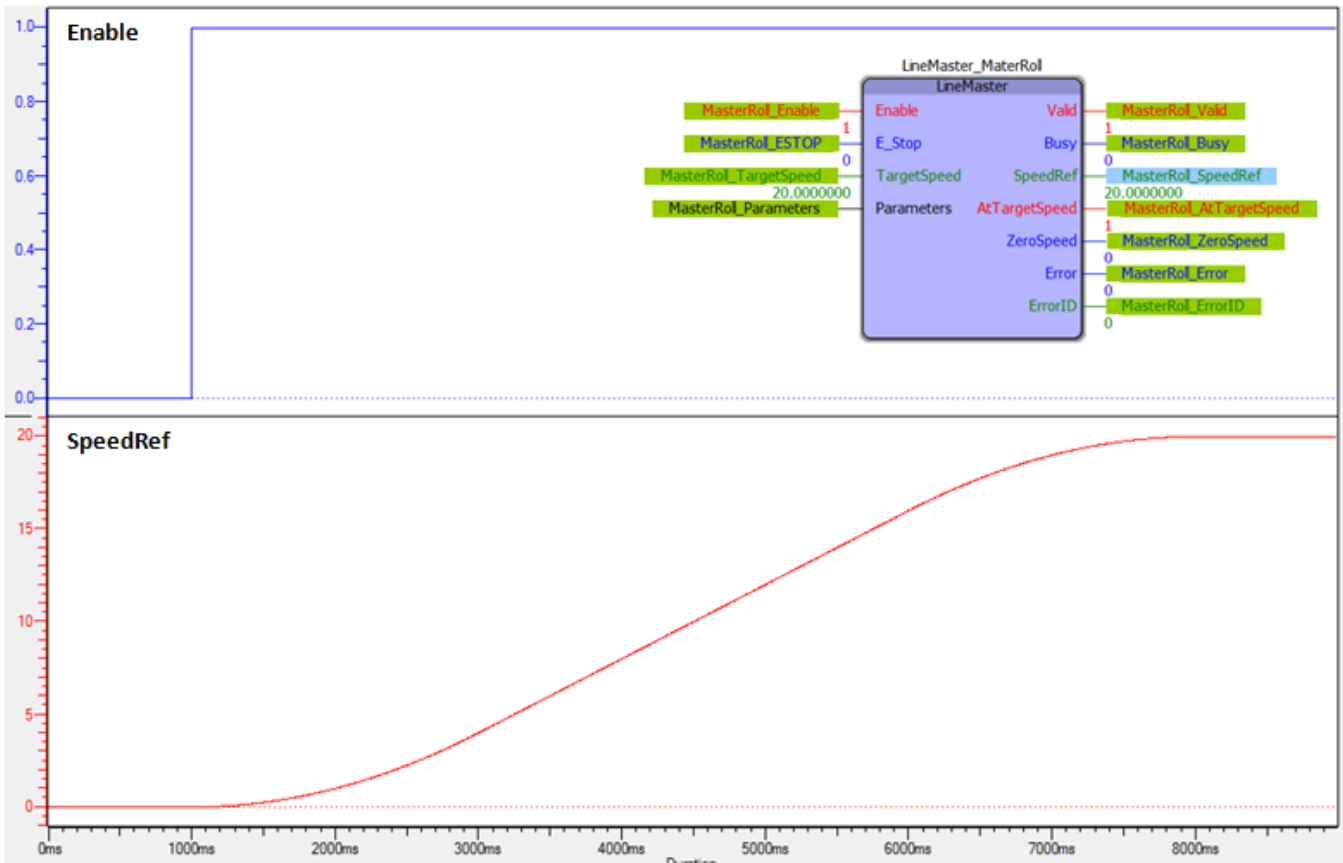
- The line master speed becomes a standard speed of the entire line. The speed reference for all axes are based on the line master speed.
- An [SLAU](#) function block from the Yaskawa Toolbox is used in the LineMaster function block. See the SLAU function block help for how to set Normal_Rate and Shape_Time.

Error Description

See the [Function Block ErrorID](#) list.

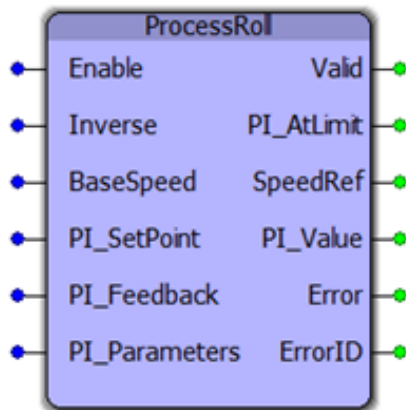
Example







ProcessRoll



The ProcessRoll function block provides the velocity for a tension cut roll with PI control.

Library

Winding Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Inverse	BOOL	This input is dependent upon the PI output of the feedback device.	FALSE
V	BaseSpeed	LREAL	The BaseSpeed input is the Line Master Speed after calculating the control gains.	LREAL#0.0
V	PI_SetPoint	LREAL	Defines the process support module baseline value.	LREAL#0.0
V	PI_FeedBack	LREAL	Scaled input from the process support module, usually a dancer or loadcell.>	LREAL#0.0
V	PI_Parameters	PI_Struct	Proportional and Integral gain parameters used for the roll.	All zeros in Structure
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	PI_AtLimit	BOOL	Indicates that the axis has reached the designated PI Limit.	
V	SpeedRef	LREAL	Real time speed that the process roll is moving.	

V	PI_Value	LREAL	Provides the PI value from the support module, usually a dancer or loadcell.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

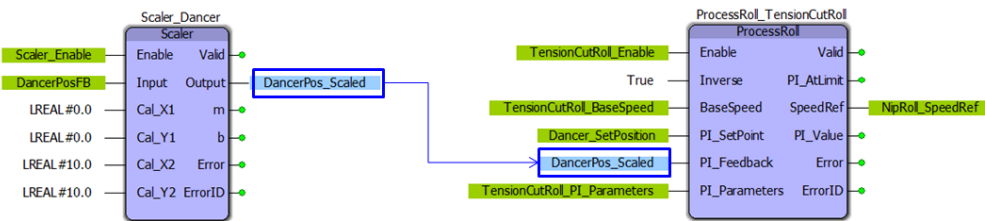
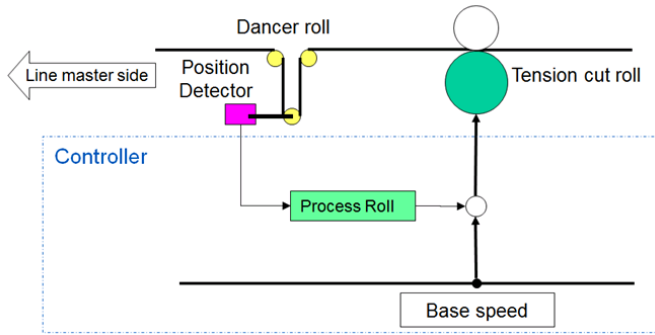
Notes

- The dancer roll is used to adjust the speed reference used by the Tension cut roll. In this manner, the LineMaster speed remains constant, while the Dancer roll accumulates any slack and adjusts the Tension cut roll speed. This ProcessRoll function block can also be used for a load cell feedback.

Error Description

See the [Function Block ErrorID](#) list.

Example





Getting Started with Yaskawa Toolbox

Requirements

To use the Yaskawa Toolbox, the project must also contain the following:

Firmware libraries:

- YDeviceComm
- PROCONOS

User libraries:

- None



Yaskawa Revision History

Current Version:

2023-09-37 v375 released

- 1) Blink FB - Stopped blinking when PLC Tick Count rolled over. DCR 7286.

Previous Versions:

2022-05-08 v374 released

- 1) TemperatureControl FB - Did not allow rewriting TC_Prms for config file version 20181217. DCR 6543.

2021-04-09 v372 released

- 1) DataSort FB - Fixed Watchdog caused when X variables are equal to each other. DCR 5175.

2020-11-02 v371 released

- 1) No changes, same as v370.

2020-02-06 v370 released

- 1) TemperatureControl FB - Should not output a control value when not Enabled. DCR 2533.
- 2) TemperatureControl FB - Fixed possible stuck in Autotuning condition after 1st cycle in oven application. DCR 2675.
- 3) CTU Counter Enhancement - Added CTU_Retained, CTD_Retained, and CTUD_Retained to Yaskawa toolbox. DCR 2838.
- 4) RangeCheck - Input "maximim" spelling error was corrected. DCR 2882.
- 5) IsNumeric - Updated FB to handle exponents and commas. DCR 2987.
- 6) TemperatureControl / CalculateGains - CycleError occurs with Un-initialized TempControl_Prms. DCR 3005.
- 7) SLAU - Improved to work in both positive and negative direction. DCR 3054.
- 8) TemperatureControl FB - CONCAT internal FB did not work on MP2300. DCR 3065.
- 9) TemperatureControl FB - Improve validation of User Inputs. DCR 3079.
- 10) TemperatureControl FB - Description for AutoTuning and Busy outputs were incorrect. DCR 3082.
- 11) TemperatureControl FB - FileRemove error results if multiple FB instances are using the same DeviceName. DCR 3085.
- 12) TemperatureControl FB - Report an error condition if it doesn't detect change in feedback when applying a control output. DCR 3111.
- 13) IsNumeric / IsInteger FBs were not returning correct value if InputString was zero length. DCR 3643
- 14) IsNumeric / IsInteger FBs were not reporting string like "5." correctly. DCR 3684

2019-02-11 v352 released

- 1) Temperature Control - New function block added. DCR 1934.

- 2) IsNumeric - New FB added. DCR 2242.
- 3) IsInteger - New FB added. DCR 2356.
- 4) GetTaskInterval - Changed from Enable style behavior to Execute. DCR 2357.
- 5) PWM - New function block added. DCR 2579.

2018-03-08 v350 released

- 1) ChangeCase - New FB added. DCR 1348.
- 2) SLAU - Added Start Value input. DCR 892

2017-12-08 v341 released (filename is still 340, using new project version control numbering)

- 1) StripSpaces - New FB added. DCR 1304.

2017-08-14 v340 released

- 1) STRING_TO_BOOL - New FB added. DCR 1068. converts false, FALSE, true, TRUE, 0, 1, BOOL#1, BOOL#0 to BOOL datatype
- 2) GetTaskInterval - New FB added. DCR 1196.
- 3) Blink - DCR 1221. Add handling for Rollover of uSec DINT value from +2147483647 to -2147483648. At 2msec MLink, rollover occurs every approx 70 minutes. Without this fix, the output freezes.

2016-06-21 v301 released

- 1) Blink - Support Mechatrolink rates less than one millesecond. DCR 825.
- 2) PLCuSec - New function block to provide a microsecond counter.

2015-01-31 v300 released

- 1) Identical to v205, but recompiled specifically for MotionWorks IEC v3.x.

2014-11-17: v205 released. Requires firmware 2.1.0 or higher

- 1) DEC_TO_HEX and HEX_TO_DEC - Added new function blocks.
- 2) CheckSumCalculate and CheckSumValidate - Updated to include a new method for two's compliment in hexadecimal.
- 3) XYLookup - Added support for decreasing X values.
- 4) YaskawaDatatypes - Added YTB_DINT32 as an Array of 32 DINT.

2013-09-01: v204 released. Requires firmware 2.1.0

- 1) More string and byte array datatypes added to be used across the Toolbox family
- 2) LAU - new function block added. Creates a linear profile from current value to target value based on rate/scan input
- 3) SLAU - new function block added. Creates an s-curve (moving average profile) from a current value to target value.
- 4) PIControl - new function block added. Subset of PID block
- 5) Removed references to the Math Toolbox to simplify usage. NOTE: This change makes version 204 and higher incompatible with MP2600iec firmware versions 2.0, 2.1, and 2.2!
- 6) RateCalculator - new function block added.

2012-08-16: v203 released. Requires firmware 2.1.0

- 1) CheckSumValidate_BYTE - Removed the Result output and added the Method input to select a calculation method to use. There will now be a function block error if the checksum is not valid.
- 2) CheckSumCalculate_BYTE - Added the Method input.

2012-06-19: v202 released. Requires firmware 2.1.0

- 1) Sweep function improved by adding Trigger and Stream inputs.
- 2) Explicit_Message - new function block added. Y_DeviceComm firmware library added
- 3) CheckSumCalculate_BYTE - new function block added.
- 4) CheckSumValidate_BYTE - new function block added.
- 4) Blink function - resolution improved.

2011-11-16: v201 released

- 1) Reduced the size of the DataType definition for MovingAverageArray back down to 1000 as it was in v008. 30000 is too large, and causing "Data Area Exceeded" error for some users.

2011-07-29: v200 released

- 1) Built from v010beta for MotionWorks IEC 2.0.
- 2) Upgraded to Math Toolbox v200
- 3) Changed Scaler FB to allow negative slope
- 4) Fixed bug in XY Lookup (Min and Max were not getting reset for each scan.)

2011-04-27: v010 beta created

- 1) Updated to Math_Toolbox_v004
- 2) Removed spaces in filename and replaced with underscores
- 3) Changed MovingAverage to always divide by the number of samples specified by the user. Old methods divided by the number of actual samples until the entire buffer had been filled.
- 4) Changed the Blink functions frequency input to REAL datatype and the value now accepts a frequency. (Before it was TIME datatype)
- 5) Added RTCString as output of RealTimeClock FB
- 6) Added error checking to WindowCheck FB to ensure Window value is greater than zero.

2011-03-25: v009 released

- 1) Added Error logic to PIDControl
- 2) Improved MovingAverage to not require a FOR LOOP to initialize the buffer at rising edge of ENABLE
- 3) Moved Math Functions to Math Toolbox

- 4) Included ProConOS firmware library to use the Real Time Clock function, provided FB to convert RTC from STRING TO STRUCT
- 5) Added DateCompare FB, STILL UNDER TEST in v009.
- 6) Moved REM function to the Math Toolbox v002.
- 7) Added XYLookup, which is equivalent to the FGN function in the standard MP series
- 8) Added DataSort, to arrange the data for use with XYLookup if it has been collected out of order.
- 9) Added DataRecord to capture XY data by either streaming or when the Trigger input goes high.
- 10) Fixed MovingAverage - it was not properly subtracting old and adding new values.

2010-02-03: v008 released

Added REM function to return the remainder of LREAL division.

Added Pack & Unpack of Byte and Word.

Added RangeCheck function block.

Added WindowCheck function.

Added Sweep function, useful for testing a range of values.

2009-10-29: v007 released

Added ErrorID and outputs to MovingAverage.

Removed ErrorWatchDog functions.

Improved templates with new, reduced logic that does not use SET or RESET coils.

Added template functions for Enable in ST and LD.

Changed functions for MP2600 compatibility by removing EN / ENO and adding MOVE_UINT.

Added Valid output to PID function.

2009-04-03: v006 released

Added CommHeartbeat Function

2009-03-27: v005 released

Added MovingAverage Function

2009-02-06: v004 released

Added the Blink function for toggling an output at a TIME interval.

Added FB_Error_Capture, FB_Error_WatchDog, FB_Error_Clear for trapping function block errors

Corrected and improved PIDControl FB based on Eric Kelley's modifications

Under Construction! - FBError trapping functions blocks, Timestamp not implemented.

2008-10-17: v002 released

Added PIDControl Function Block and associated DataType structure

2008-09-26 v001 released

Execute_FB_Template:

Shell code with all logic to replicate the behavior of PLCopen FB with Execute, Busy, Done, Error, & ErrorID outputs
Behavior and variables match the ST version.

Execute_ST_Template:

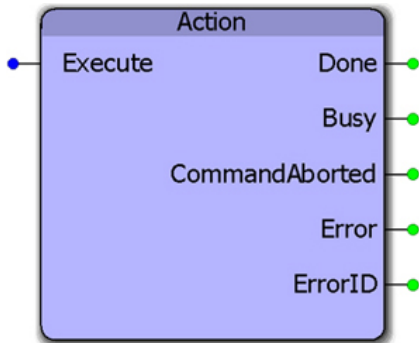
Shell code with all logic to replicate the behavior of PLCopen FB with Execute, Busy, Done, Error, & ErrorID outputs
Behavior and variables match the FB version.

Action:

Dummy FB to show simulation of the template function blocks.



Action



This function block is only for demonstration purposes. It is applied in the Enable_F_Template, Enable_ST_Template, Execute_FB_Template, and Execute_ST_Template function blocks to show how the inputs and outputs of nested functions can be interlocked to apply the PLCopen standards for I/O behavior.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Error Description

This function provides no Errors.

Example

See the Enable_F_Template, Enable_ST_Template, Execute_FB_Template, and Execute_ST_Template function blocks.



Blink



This function block will toggle the Output at the frequency specified at the input. If Frequency is set to 1.0, then the output will be on for 500 mSec and off for 500 mSec. Note that the actual frequency may be affected by the application scan rate in which this function block is placed.

Library

Yaskawa Toolbox

Parameters

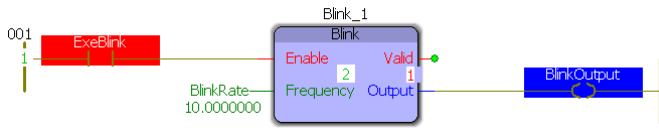
	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Frequency	REAL	The cycle rate in Hertz.	REAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates if the function is operating.	
V	Output	BOOL	Toggled at the specified frequency when the function is enabled.	

Error Description

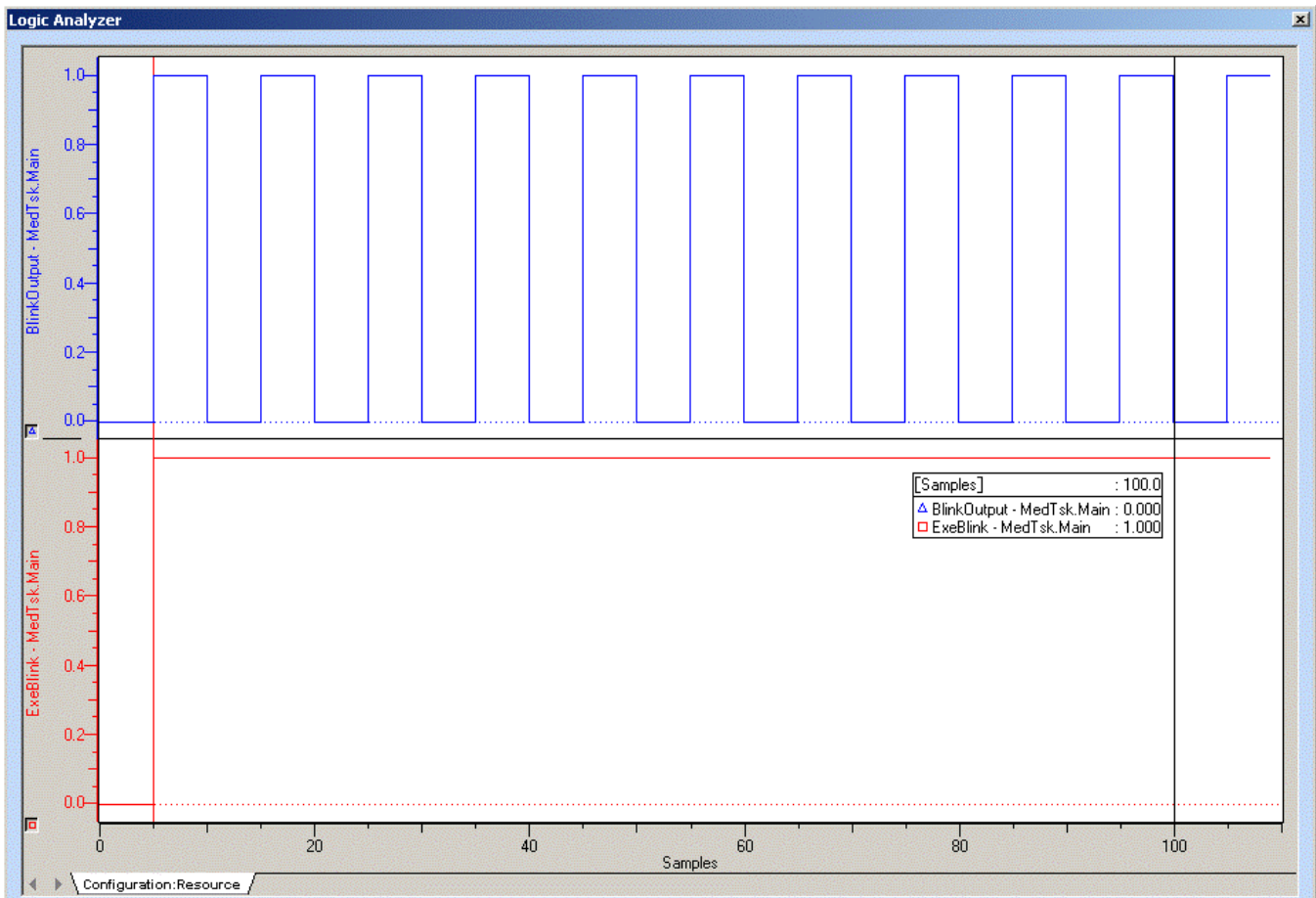
The valid output will be high if the function is operating. If Enable is held high and the Frequency is not greater than zero, the valid output will be low.

Example

Blink_1 was placed in a 10 mSec task so the expected output is 50 mSec on and 50 mSec off which corresponds to 10 cycles.



Logic Analyzer output:





ByteSwap



This function block swaps the upper and lower byte of a word.

Library

Yaskawa Toolbox

Parameters

	Parameter	Data Type	Description	
VAR_INPUT	Default			
V	WordIn	WORD	Input word	WORD#0
VAR_OUTPUT				
V	WordOut	WORD	Output word	

Error Description

This block will not produce any errors.

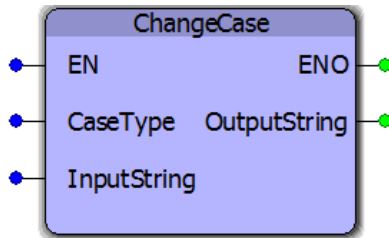
Example:



ChangeCase



ChangeCase



This function will convert the InputString to the OutputString but modify the characters to be lower case, upper case, or title case, which will capitalize the first letter of each word.

Library

Yaskawa Toolbox

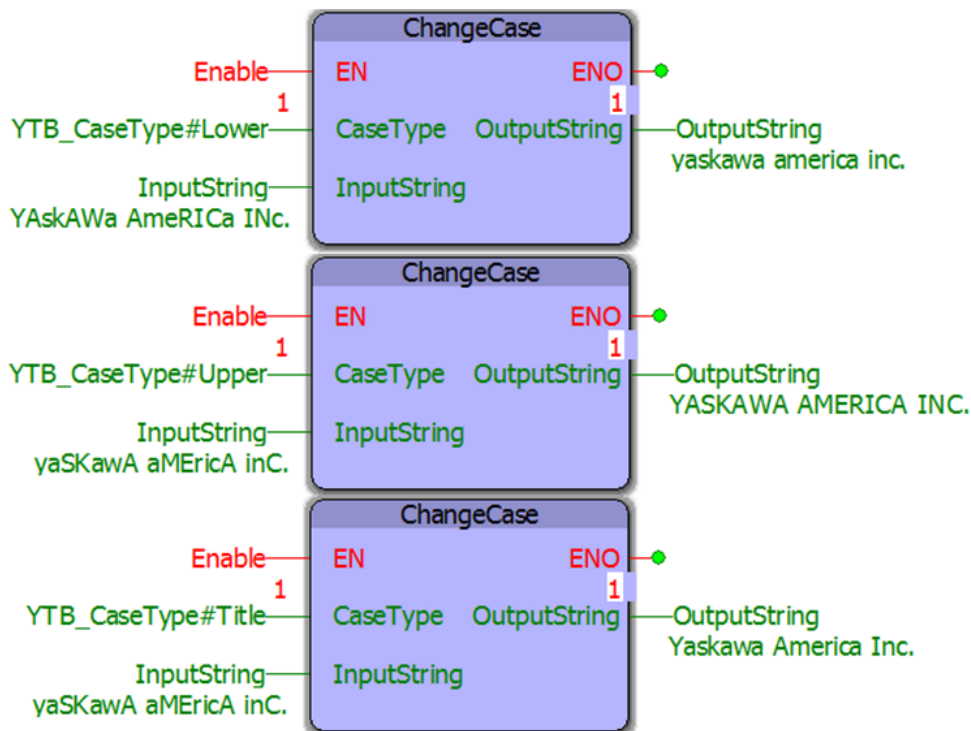
Parameters

	Parameter	Data Type	Description	
VAR_INPUT				Default
B	EN	BOOL	This function will continue to update while EN is held high.	FALSE
V	CaseType	INT	Enumeration to choose the conversion type. Select 0 for lower case, 1 for UPPER CASE, and 2 for Title Case. Title Case will capitalize the first letter of each word. Enumerations values can also be used. (YTB_CaseType#Lower, YTB_CaseType#Upper, YTB_CaseType#Title)	INT#0 (YTB_CaseType#Lower)
V	InputString	STRING	The String to be modified.	' '
VAR_OUTPUT				
B	ENO	BOOL	High if the function is executing normally.	
V	OutputString	STRING	Modified output of the string that was input to the function block.	

Notes

- EN must be set high for this function to operate.
- The OutputString is only valid if ENO is high. If for some reason the conversion had an error, ENO will not be set.
- The PROCONOS firmware library must be added to the project, otherwise, the compiler error "A function block POU "STRING_TO_BUF" is referencing a sub function, but the library containing the function is not included."

Example





CommWatchDog



This function block allows the application program to monitor data being transmitted from a master device. If the data does not change within the TimeOut period, then the OK output goes off to indicate that the communications is not being updated by the master.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	HeartBeat	DINT	Value that the master changes and sends to the MPiec controller.	DINT#0
V	WatchDog	DINT	The HeartBeat input must change value within the WatchDog period for communications to be considered OK.	DINT#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	OK	BOOL	Indicates if the HeartBeat input has changed within the TimeOut period.	

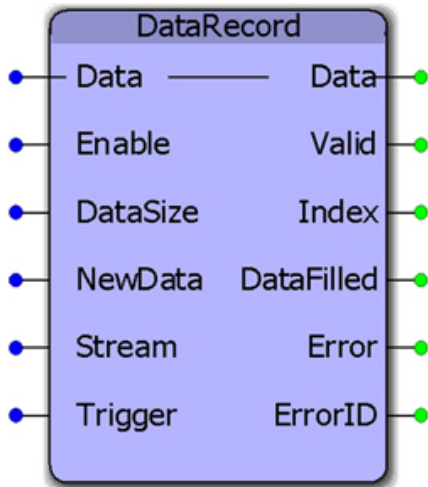
Error Description

The Valid Output will be high when the function is executing. If the WatchDog value is not greater than zero, the function will not operate.

Example



DataRecord



This function block will record Data into the array. Data can be stored continuously or intermittently. The default datatype for Data to be recorded can be customized by the user to satisfy other recording needs.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Data	XYDataStruct	Structure where recorded data is stored.	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	DataSize	INT	The maximum amount of data to be stored, which must be less than or equal to the datatype definition for Data.	INT#0
V	NewData	XYData	Structure containing a single pair of X and Y data to be added to the XYDataStruct.	n/a
V	Stream	BOOL	If TRUE, the function will store NewData every application scan.	FALSE
V	Trigger	BOOL	If Stream is FALSE, then the function will store new Data only upon the rising edge of Trigger.	FALSE
VAR_OUTPUT				

*	Parameter	Data Type	Description
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	Index	INT	Indicates the last array index recorded.
V	DataFilled	BOOL	Indicates when the Data recording has reached the DataSize.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

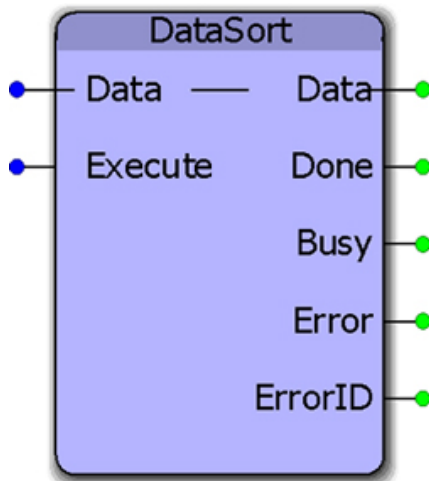
Error Description

see the [Function Block ErrorID](#) list.

Example



DataSort



This function block will sort data from the lowest to highest value of X data. This generic function can be customized for other sorting needs.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	Data	XYDataStruct	Structure where recorded data is stored.
VAR_INPUT			Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.
VAR_OUTPUT			
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)

*	Parameter	Data Type	Description
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

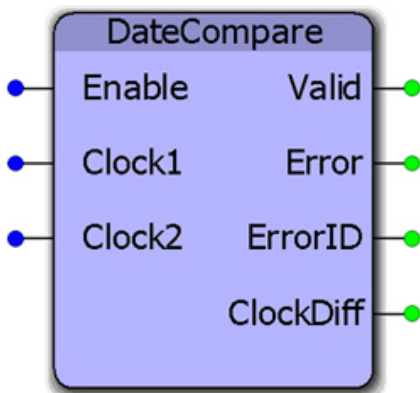
This function is designed to sort by the X data in ascending order only.

Error Description

The default version of this block produces no errors (customizing this block may add errors depending on what functions are used internally).



DateCompare



This function block will calculate the difference between two real time clock values and provide the difference as a real time clock value. The clock values may be obtained using the RealTimeClock function block.

Library

Yaskawa Toolbox

Parameters

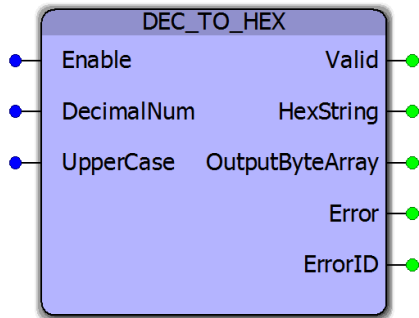
*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Clock1	RTCStruct	The more recent real time clock value.	All zeros in structure
V	Clock2	RTCStruct	The older real time clock value.	All zeros in structure
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	ClockDiff	RTCStruct	Outputs the time difference between Clock1 - Clock2.	

Error Description

There will be no Errors reported.



DEC_TO_HEX



This function block converts DINT numeric input into a hexadecimal STRING output.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	DecimalNum	DINT	Input value to be converted.
V	UpperCase	BOOL	If True, the output string will contain upper case ASCII characters. If False, the output string will contain lower case ASCII characters.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	HexString	STRING	String containing the hexadecimal representation of DecimalNum.
V	OutputByteArray	YTB_ByteArray8	Byte array containing the same ASCII values as in the OutputString.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

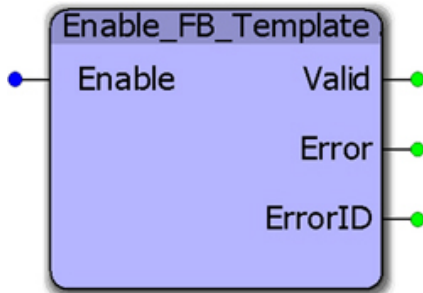
Notes

Error Description

This function block may output ErrorIDs from the BUF_TO_STRING function used internally.



Enable_FB_Template



This function block is a template which can be used when developing functions which adhere to the PLCopen output behavior.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Error Description

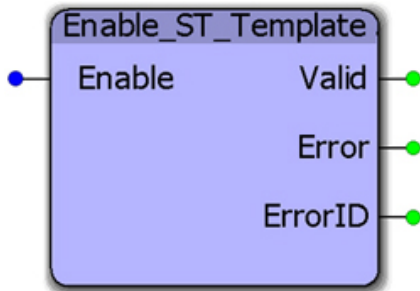
This is an example function block template with no specific errors of its own.

Example

For full documentation about how to create PLCopen compliant function blocks, see this [application note](#) (AN.MWIEC.01) on www.yaskawa.com.



Enable_ST_Template



This function block is a template which can be used when developing functions which adhere to the PLCopen output behavior.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_INPUT			Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Error Description

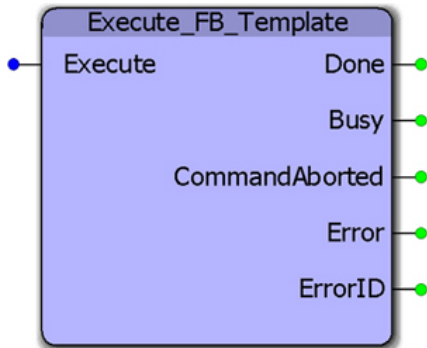
This is an example function block template with no specific errors of its own.

Example

For full documentation about how to create PLCopen compliant function blocks, see this [application note](#) (AN.MWIEC.01) on www.yaskawa.com.



Execute_FB_Template



This function block is a template which can be used when developing functions which adhere to the PLCopen output behavior.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

Depending on the exact usage, there may be outputs in the template that will not apply, such as CommandAborted. Determine what outputs are necessary for your situation and make modifications accordingly.

Error Description

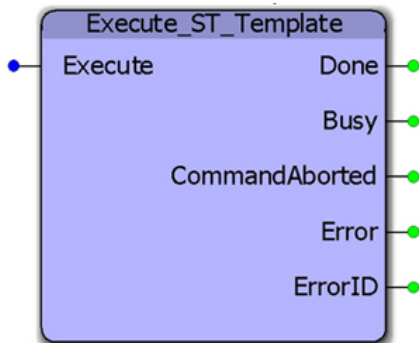
This is an example function block template with no specific errors of its own.

Example

For full documentation about how to create PLCopen compliant function blocks, see this [application note](#) (AN.MWIEC.01) on www.yaskawa.com.



Execute_ST_Template



This function block is a template which can be used when developing functions which adhere to the PLCopen output behavior.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior as the Done output.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

This template contains supporting code for:

- Initialization
- Main code body
- Output status updates

Depending on the exact usage, there may be outputs in the template that will not apply, such as CommandAborted. Determine what outputs are necessary for your situation and make modifications accordingly.

Error Description

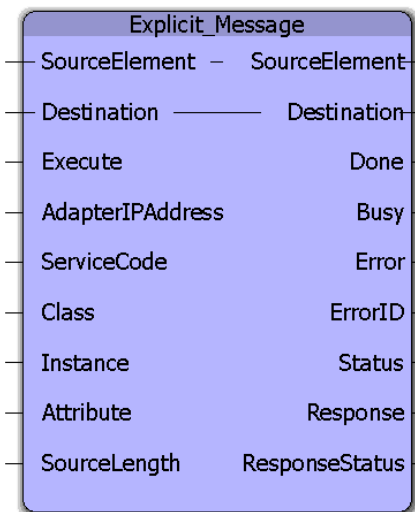
This is an example function block template with no specific errors of its own.

Example

For full documentation about how to create PLCopen compliant function blocks, see this [application note](#) (AN.MWIEC.01) on www.yaskawa.com.



Explicit_Message



This function block will write/read a block of data to/from an Ethernet/IP Target (Adapter) device via Explicit Messaging. Unlike Implicit Messaging (a built in feature of the MPiec Series Controllers) which uses the UDP protocol, Explicit Messaging uses TCP/IP.

This function block emulates the MSG function block in the AB RSLogix platform. The Explicit_Message function block is best suited when an application requires unscheduled and less frequent updates like recipe transfer, cam table transfer, job transfer etc. Explicit Messaging makes use of a request/response format for communication.

Library

Yaskawa Toolbox

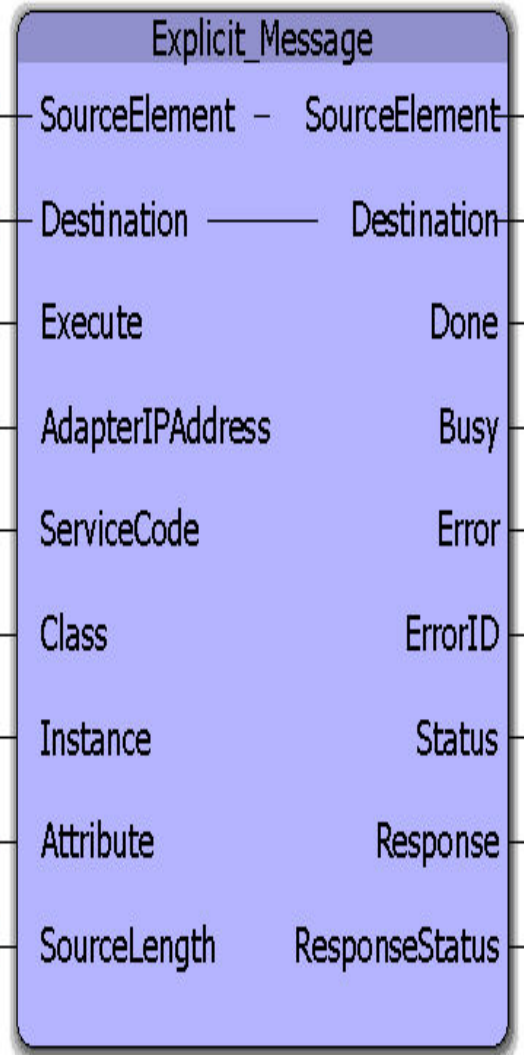
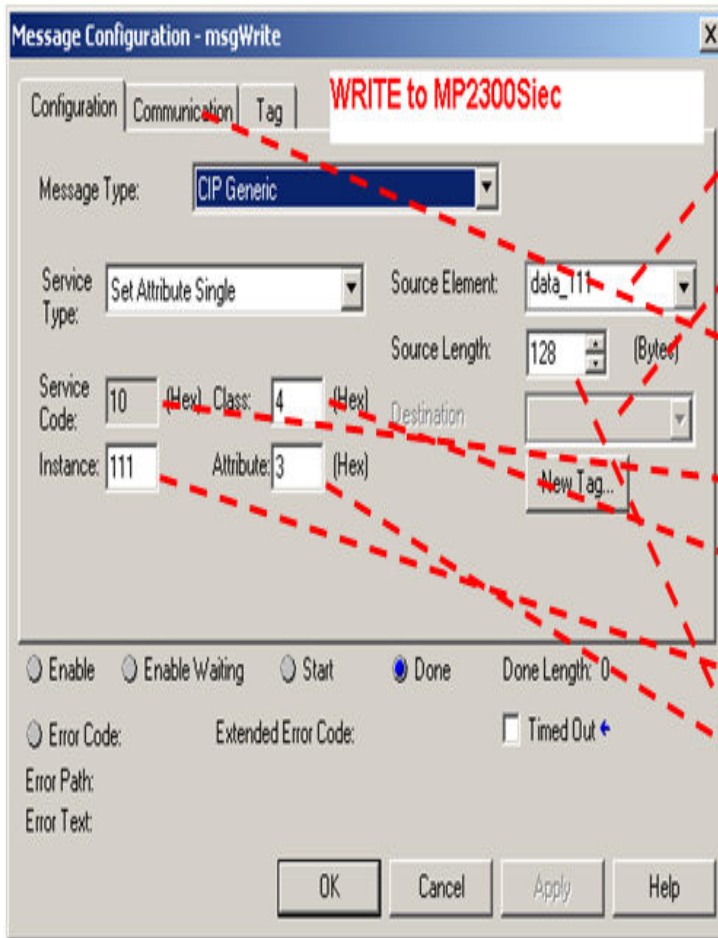
Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	SourceElement	ExplicitData	When writing a message to the Target (Adapter), SourceElement is the data (as an array of bytes) that the Scanner (MPiec Controller) will send to the Target.
V	Destination	ExplicitData	When reading a message from the Target (Adapter), the Destination Element is the data (as an array of bytes) where the Scanner (MPiec Controller) will copy the data from the Target.
VAR_INPUT			Default

*	Parameter	Data Type	Description	
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	AdapterIPAddress	STRING	IP Address of the Target device.	STRING#"
V	ServiceCode	BYTE	Code for the particular service type as defined for a CIP message. The value can be obtained from the Target's (Adapter's) documentation.	BYTE#0
V	Class	BYTE	Class parameter of a CIP Generic message. The value can be obtained from the Target's (Adapter's) documentation.	BYTE#0
V	Instance	BYTE	Instance parameter of a CIP Generic message. The value can be obtained from the Target's (Adapter's) documentation.	BYTE#0
V	Attribute	BYTE	Attribute parameter of a CIP Generic message. The value can be obtained from the Target's (Adapter's) documentation.	BYTE#0
V	SourceLength	INT	The number of bytes to be written to the Target. This is the actual data size required, not the full size of the SourceData DataType.	BYTE#0
VAR_OUTPUT				
B	Done	BOOL	The done bit is set high when the last packet of the message is successfully transferred.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	Status	DWORD	Indicates if the Target was able to execute the requested command. A value of zero indicates successful execution of the command by the remote device.	
V	Response	WORD	Response from the Target.	
V	ResponseStatus	WORD	Status of the response from the Target.	

Notes

- The Explicit_Message function block uses the Y_DeviceComm firmware library. This firmware library must be added to your project. Y_DeviceComm was incorporated into firmware version 2.1.0 and has been included as a firmware library starting in MotionWorks IEC v2.1.0.
- Enter parameters as entered in Message Configuration for the MSG function block in AB RSLogix software.
- See Yaskawa's Youtube webinar - [EtherNet/IP Explicit Messaging](#) for more info.



Error Description

See the [Function Block ErrorID](#) list.

Example 1

Set single attribute.

Variable Properties

Name: SrcEle
 Data Type: ExplicitData
 Usage: VAR RETAIN
 Initial value:

Explicit_Message_6

Explicit_Message

SrcEle - SourceElement - SourceElement - SrcEle
 D1 - Destination - Destination - D1
 Execute - Done - SetAttributeDone
 AdapterIPAddress - Busy - SetAttributeBusy
 ServiceCode - Error - SetAttributeError
 Class - ErrorID - SetAttributeErrorID
 Instance - Status - Stat
 Attribute - Response - Resp
 SourceLength - ResponseStatus - RespStat

Watch Window

Variable	Value
SrcEle	
[0]	0
[1]	0
[2]	0
[3]	0
[4]	0
[5]	0
[6]	0
[7]	0
[8]	0
[9]	0
[10]	0
[11]	0
[12]	0
[13]	0

Variable Properties

Name: D1
 Data Type: ExplicitData
 Usage: VAR RETAIN
 Initial value:

STRING#'192.168.207.88'

(*)Write instance 111 to MP2000iec slave*)

Example 2

Get single attribute.

Variable Properties

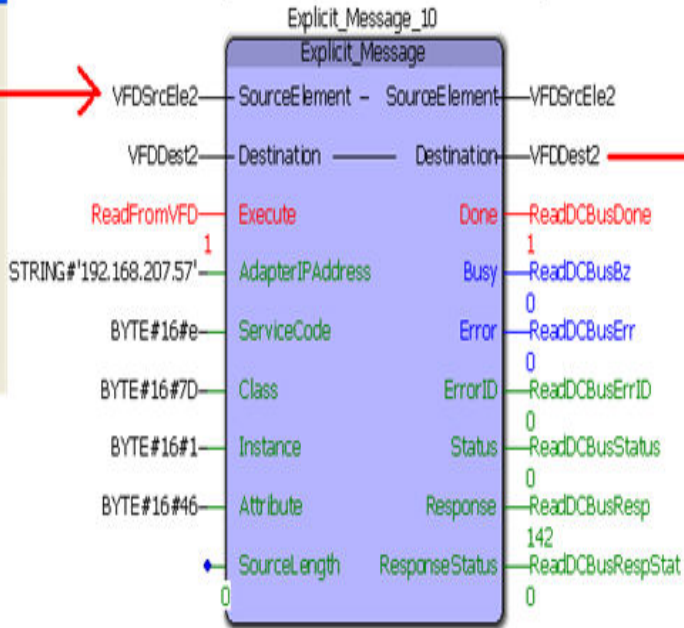
Name: VFDSrcEle2

Data Type: ExplicitData

Usage: VAR RETAIN

Initial value:

(*Read DC bus from V1000*)



Variable Properties

Name: VFDDest2

Data Type: ExplicitData

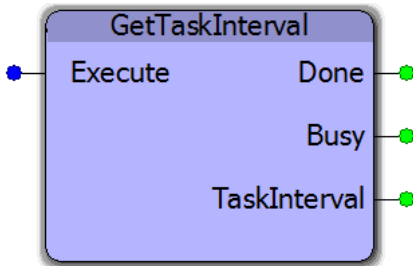
Usage: VAR RETAIN

Initial value:

VFDDest2	
[0]	44
[1]	1
[2]	0
[3]	0
[4]	0
[5]	0
[6]	0
[7]	0
[8]	0
[9]	0
[10]	0
[11]	0
[12]	0
[13]	0



GetTaskInterval



This function block reports the task interval for the CYCLIC task in which it is executing. It requires two scans to complete. It counts the number of Mechatrolink ticks that occurred between two scans by referencing global variables PLC_SYS_TICK_CNT and PLC_TICKS_PER_SEC. Note: This block was changed in v352 from an Enable input to Execute style interface.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	Default
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when Execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
V	TaskInterval	REAL	In milliseconds.	

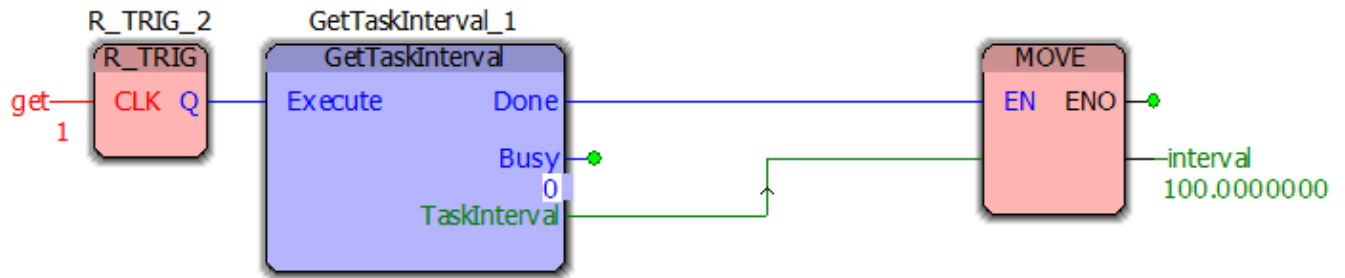
Notes

This block may be useful in applications which use Y_DirectControl, or other applications that require information about the task interval to operate successfully.

Error Description

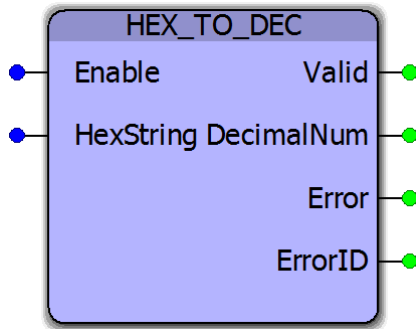
No errors will be generated.

Example





HEX_TO_DEC



This function block converts a hexadecimal STRING into a base 10 output value as a DINT.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	HexString	STRING	Input hexadecimal string. Can only contain values 0-9, A-F, and a-f. A maximum of 8 characters is allowed, because this would represent the maximum value of a DINT as STRING#'FFFFFFF'	STRING#''
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	DecimalNum	DINT	Output value from hexadecimal conversion.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

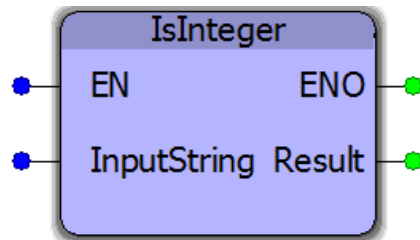
- Converts both upper and lower case ASCII characters to a DINT value.

Error Description

This function may output ErrorIDs from the STRING_TO_BUF Status output used internally.



IsInteger



This function block checks if InputString is an integer (Any non floating point value such as BYTE, SINT, INT, UINT, DINT, UDINT). If InputString is an integer, Result will be set high. This function block is useful when preparing to copy string data into other datatypes using the STRING_TO_* functions. If the data is corrupt or not an integer when using the STRING_TO_* functions, a String Conversion Exception will occur. This function block helps to eliminate that risk.

Library

Yaskawa Toolbox

Parameters

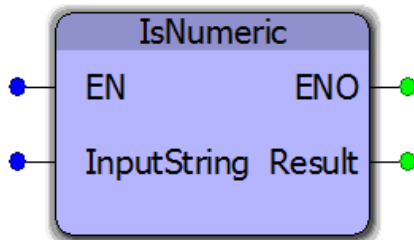
*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	EN	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	InputString	STRING	Input string to be tested.	STRING#"
VAR_OUTPUT				
B	ENO	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	Result	BOOL	Result will be TRUE if the string is convertible to an integer.	

Notes

The PROCONOS firmware library must be added to the project, otherwise, the compiler error "A function block POU "STRING_TO_BUF" is referencing a sub function, but the library containing the function is not included."



IsNumeric



If InputString is numeric, Result will be set high. This function block is useful when preparing to copy string data into other datatypes using the STRING_TO_* functions. If the data is corrupt or not a valid number when using the STRING_TO_ functions, a String Conversion Exception will occur. This function block helps to eliminate that risk. Valid strings may include a '+', '-' or '.' character as the first character, otherwise every character must be between or including '0 -9'. If there is an 'E' for scientific notation, another '+' or '-' may immediately follow the 'E'.

Library

Yaskawa Toolbox

Parameters

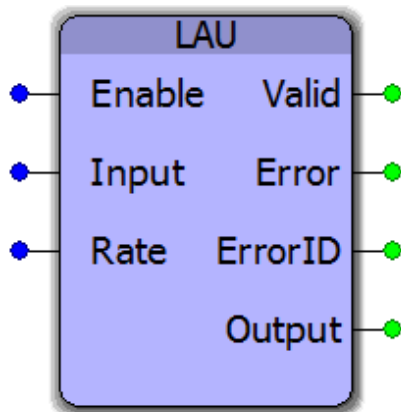
*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	EN	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	InputString	STRING	Input string to be tested.	STRING#"
VAR_OUTPUT				
B	ENO	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	Result	BOOL	Result will be TRUE if the string is convertible to a number.	

Notes

The PROCONOS firmware library must be added to the project, otherwise, the compiler error "A function block POU "STRING_TO_BUF" is referencing a sub function, but the library containing the function is not included."



LAU



This function block generates a linear ramp from a current value to the target value with a slope based on the Rate input on the function block. The input can be continuously update on the fly.

Library

Yaskawa Toolbox

Parameters

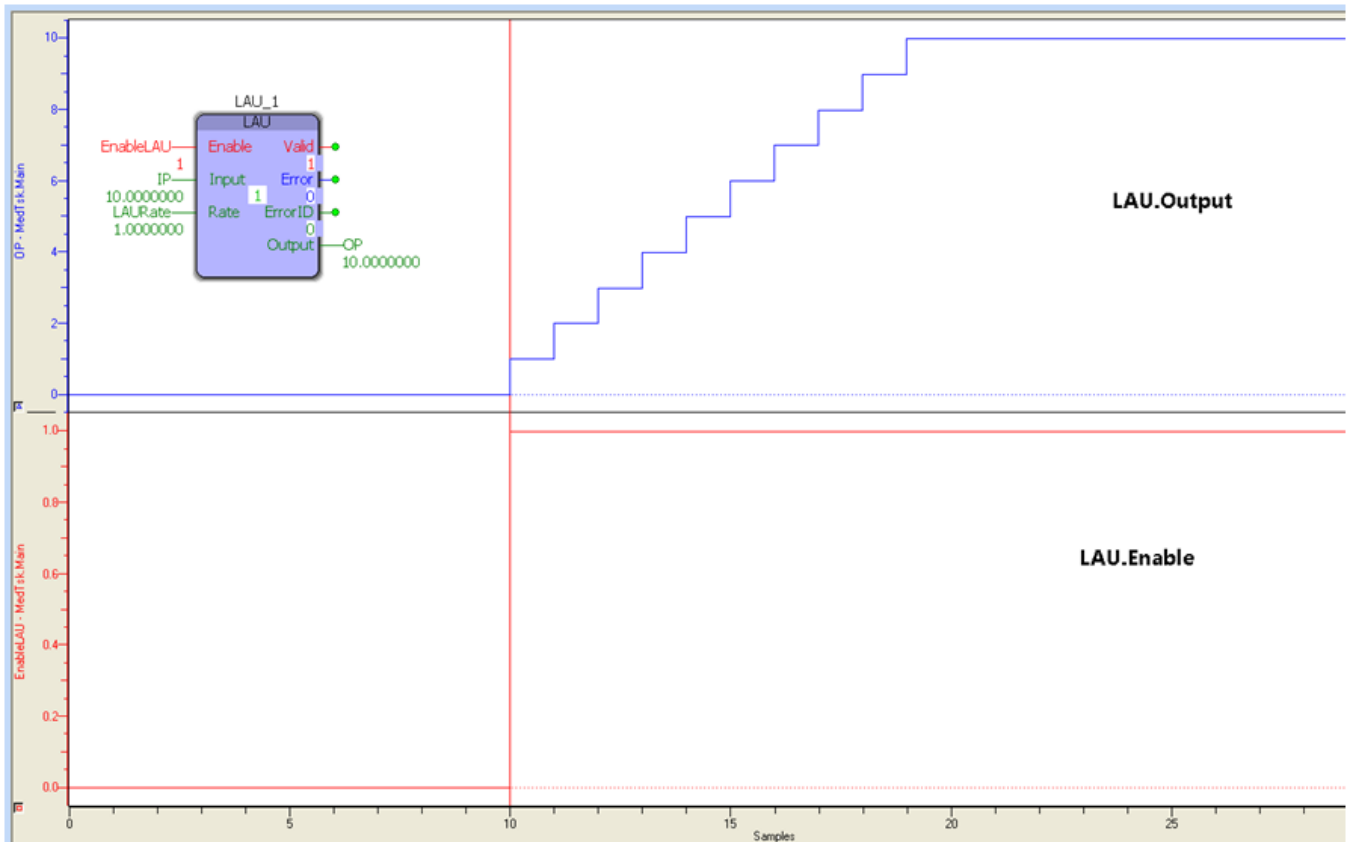
*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Input	LREAL	Target value.	LREAL#0.0
V	Rate	LREAL	Acceleration/Deceleration per scan. The time required for the Output to become the Input value profile depends on the Rate Input and the interval (Application task rate) at which the LAU function block is being run.	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	Output	LREAL	Output value.	

Error Description

See the [Function Block ErrorID](#) list.

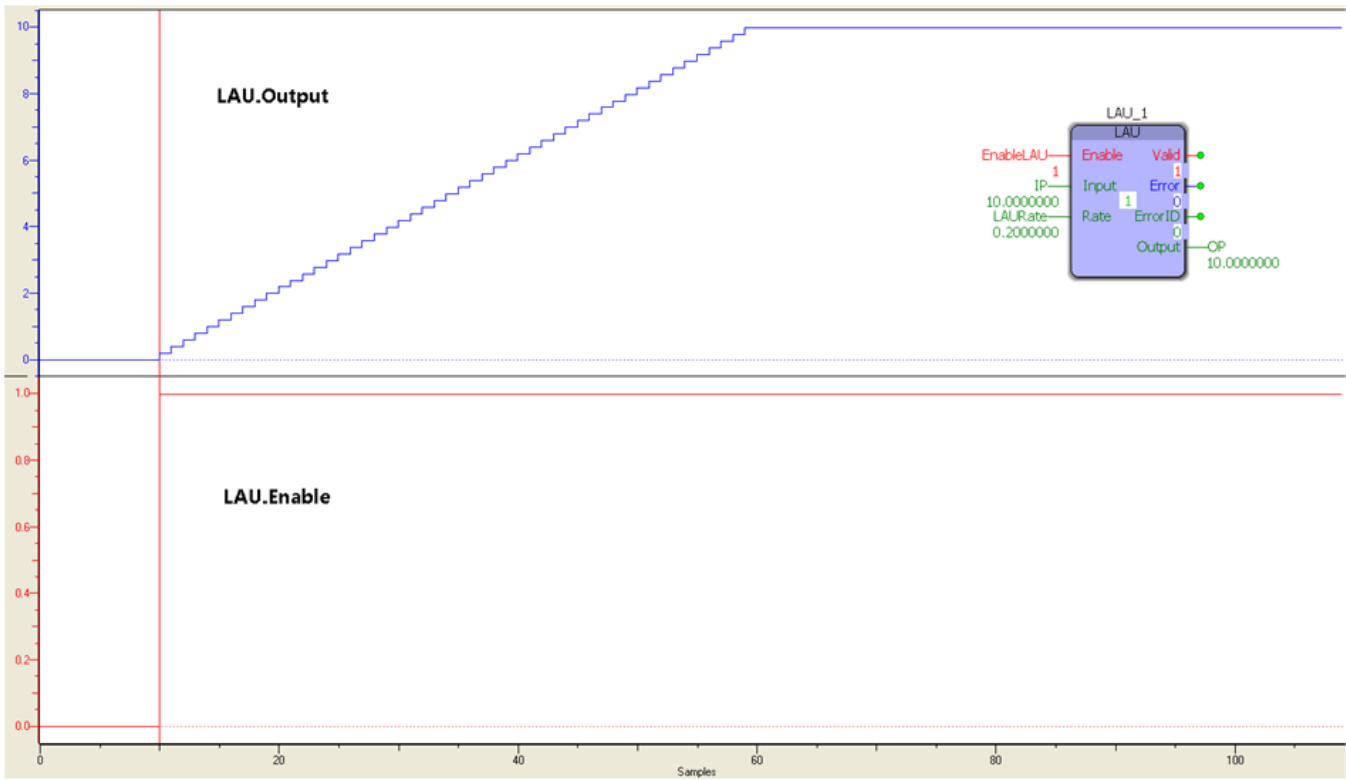
Example 1

The figure shown below illustrates how the LAU block can generate a ramp output even if the Input value instantaneously changes from 0 to 10. The rate of 1 unit specifies that the output value is expected to increase at the rate of 1 unit per application scan.



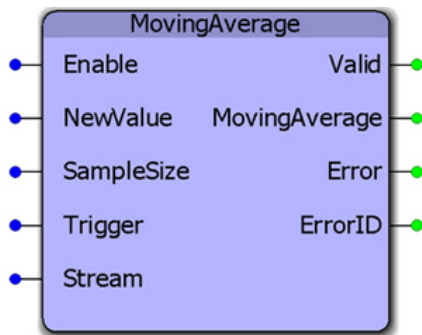
Example 2

The figure shown below shows the effect of a ramp of 0.2 units per scan. The signal takes 50 application scans to reach 10 units from an initial value of 0.





MovingAverage



This function block will provide the MovingAverage of a series of samples. The NewValue can either be streamed continuously or updated only when the Trigger input goes high.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	NewValue	LREAL	The new value to be added to the total	LREAL#0.0
V	SampleSize	UINT	The total number of values to total	UINT#0
V	Trigger	BOOL	To indicate when a NewValue should be added to the total	FALSE
V	Stream	BOOL	To indicate if the NewValues should be added to the total every scan.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	MovingAverage	LREAL	The moving average of all the samples.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

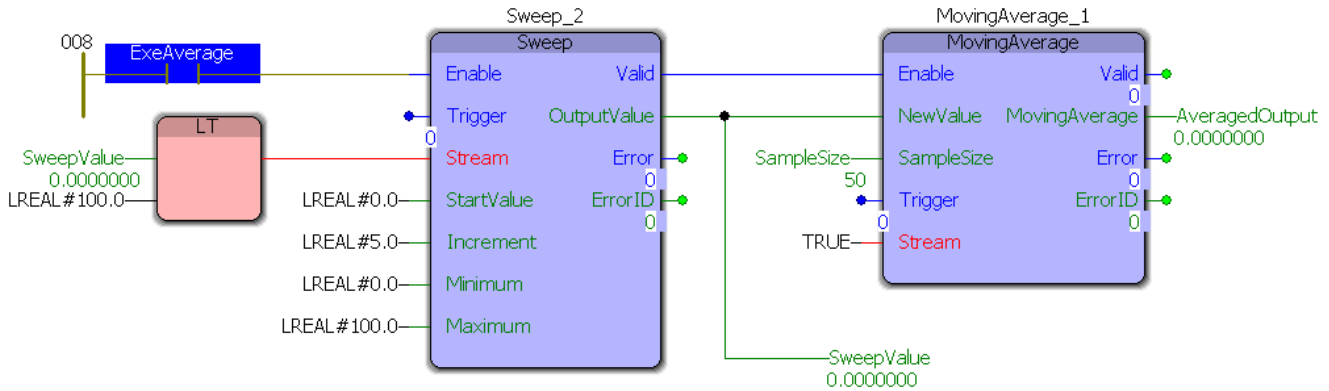
See Yaskawa's YouTube webinar - [MPiec Web Tension Control Applications](#) for more info on using this function block.

Error Description

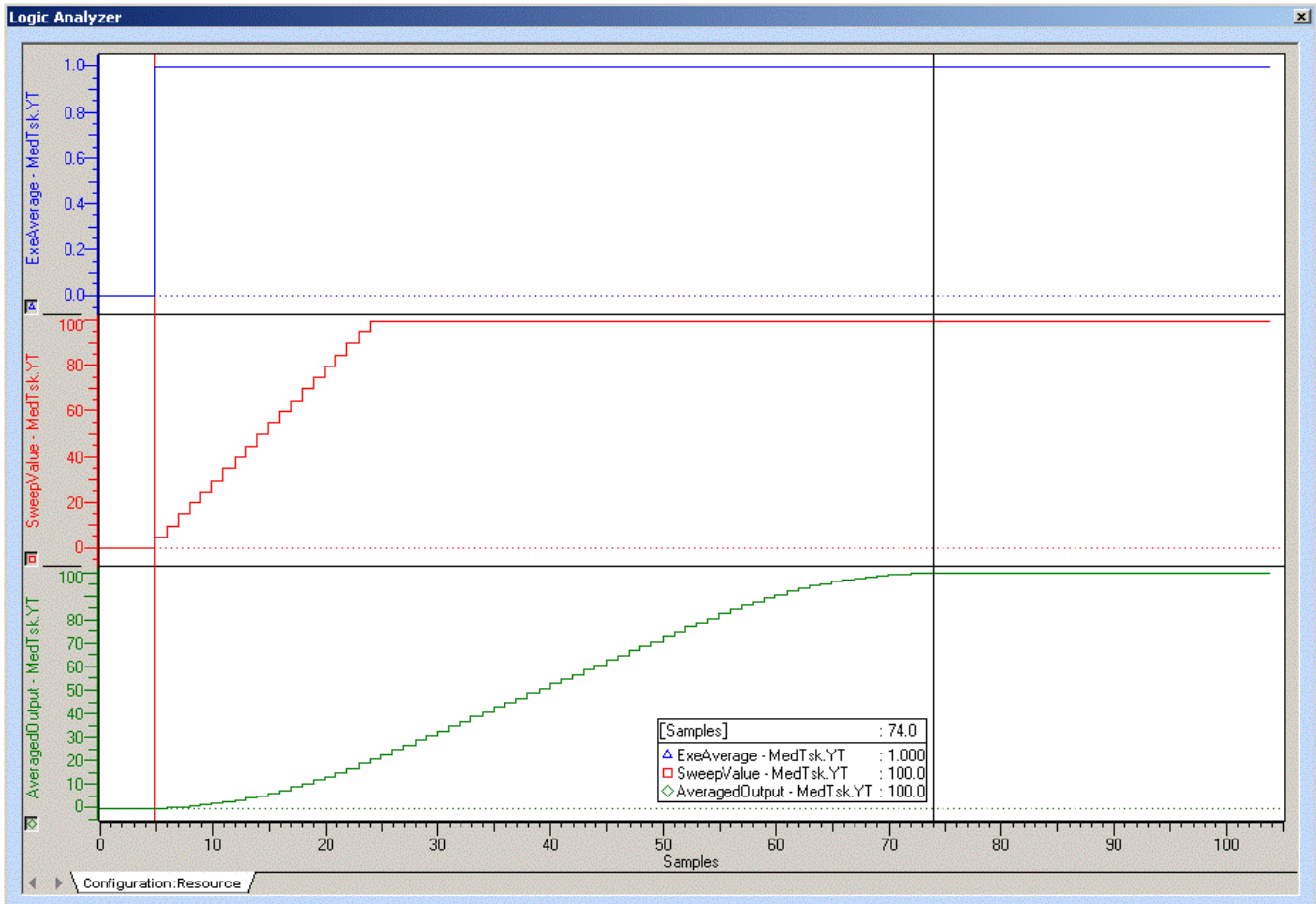
See the [Function Block ErrorID](#) list.

Example

The MovingAverage function acts as a smoothing filter. In this example, the Sweep function will increment by 5 each cycle. The Sweep function will continue to increment the OutputValue until it has reached 100.

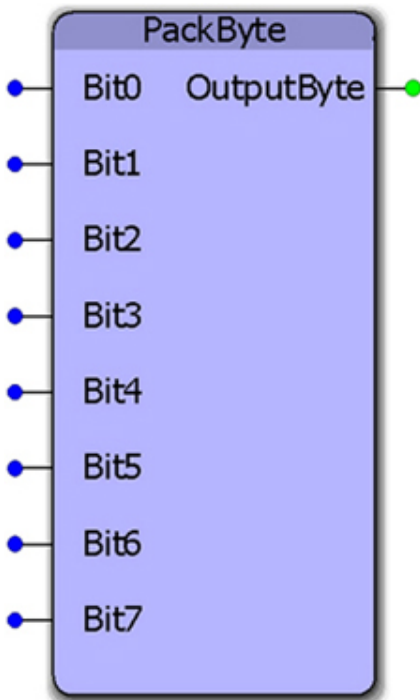


The Moving average function has a sample size of 50 which means that if Sweep reaches its maximum value after 19 cycles, MovingAverage will output the maximum value after 69 cycles. By looking at the Logic Analyzer plot below, notice there is a 5 cycle pre-record that must be taken in to account: $74 - 5 = 69$ cycles.





PackByte



This function block converts 8 BOOL inputs to a single BYTE output.

Library

Yaskawa Toolbox

Parameters

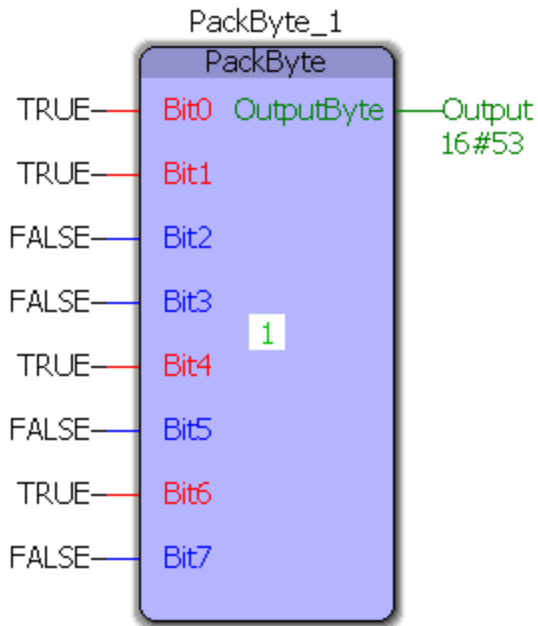
*	Parameter	Data Type	Description	
VAR_INPUT				Default
V	Bit0	BOOL	Bit 0 of the BYTE to be output	FALSE
V	Bit1	BOOL	Bit 1 of the BYTE to be output	FALSE
V	Bit2	BOOL	Bit 2 of the BYTE to be output	FALSE
V	Bit3	BOOL	Bit 3 of the BYTE to be output	FALSE
V	Bit4	BOOL	Bit 4 of the BYTE to be output	FALSE
V	Bit5	BOOL	Bit 5 of the BYTE to be output	FALSE
V	Bit6	BOOL	Bit 6 of the BYTE to be output	FALSE
V	Bit7	BOOL	Bit 7 of the BYTE to be output	FALSE

*	Parameter	Data Type	Description
VAR_OUTPUT			
V	OutputByte	BYTE	Resulting byte of the input bits

Error Description

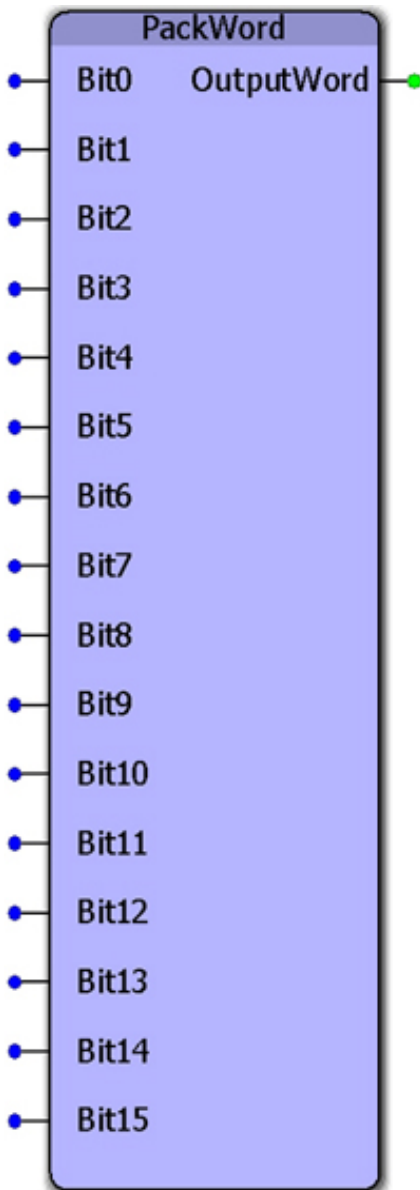
No errors will be generated

Example





PackWord



This function block converts 16 Boolean inputs to a single WORD output.

Library

Yaskawa Toolbox

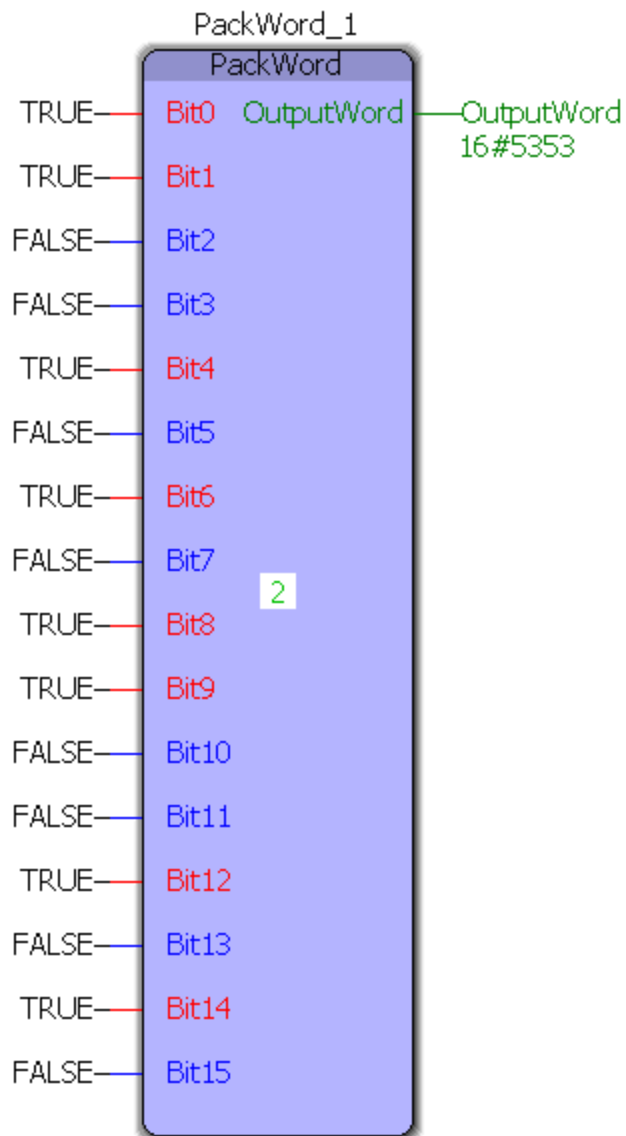
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
V	Bit0	BOOL	Bit 0 of the WORD to be output	
V	Bit1	BOOL	Bit 1 of the WORD to be output	
V	Bit2	BOOL	Bit 2 of the WORD to be output	
V	Bit3	BOOL	Bit 3 of the WORD to be output	
V	Bit4	BOOL	Bit 4 of the WORD to be output	
V	Bit5	BOOL	Bit 5 of the WORD to be output	
V	Bit6	BOOL	Bit 6 of the WORD to be output	
V	Bit7	BOOL	Bit 7 of the WORD to be output	
V	Bit8	BOOL	Bit 8 of the WORD to be output	
V	Bit9	BOOL	Bit 9 of the WORD to be output	
V	Bit10	BOOL	Bit A of the WORD to be output	
V	Bit11	BOOL	Bit B of the WORD to be output	
V	Bit12	BOOL	Bit C of the WORD to be output	
V	Bit13	BOOL	Bit D of the WORD to be output	
V	Bit14	BOOL	Bit E of the WORD to be output	
V	Bit15	BOOL	Bit F of the WORD to be output	
VAR_OUTPUT				
V	OutputWord	WORD	The resulting WORD of the input bits	

Error Description

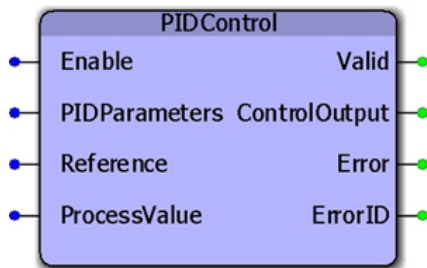
No errors will be generated

Example





PIDControl



This function block can be used as a generic control loop feedback mechanism. A PID controller calculates an "error" value as the difference between a measured process variable and a desired set point, or reference. PIDParameters must be adjusted to allow the process to provide the proper ControlOutput for a given error situation.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	PIDParameters	PIDStruct	Structure containing all the information for PID control block to operate	N/A
V	Reference	LREAL	Setpoint for the PID control loop.	LREAL#0.0
V	ProcessValue	LREAL	Real world value to be compared with the Reference in the control loop	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	ControlOutput	LREAL	Output value from the PID control block. The range of values will be governed by the PID parameters, especially the upper and lower limit.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

- All time parameters in the PIDStruct (Ts, Td1, and Td2) must be in the same units, i.e seconds or ms.
- See Yaskawa's Youtube webinar - [MPiec Web Tension Control Applications](#) for more info on using this function block.

Example

Initialization of the PIDStruct:

```
PIDPrm.Ts      := LREAL#0.004; (* Set to the same value as the cyclic application task *)
PIDPrm.Kp      := LREAL#40.0;  (* Proportional gain *)
PIDPrm.Ki      := LREAL#0.0;   (* Integral gain *)
PIDPrm.Kd      := LREAL#0.0;   (* Derivative gain *)
PidPrm.Td1     := LREAL#4.0;   (* Divergence differentiation time *)
PidPrm.Td2     := LREAL#4.0;   (* Convergence differentiation time *)
PIDPrm.Ti      := LREAL#4.0;   (* Integration time *)
PIDPrm.ILL     := LREAL#-10.0; (* The smallest integration value *)
PIDPrm.IUL     := LREAL#10.0;  (* The largest integration value *)
PIDPrm.LowerLimit := LREAL#-2000.0; (* The smallest ControlOutput that will be output *)
PIDPrm.UpperLimit := LREAL#2000.0; (* The largest ControlOutput that will be output *)
PIDPrm.DeadBand  := LREAL#0.00001; (* Maximum absolute error value that will result in a ControlOutput of zero *)
```

Symbol	Specification
Ts	Scan time set value
Kp	Proportional gain
Ki	Integral gain
Kd	Derivative gain
Td1	Divergence differentiation time
Td2	Convergence differentiation time
Ti	Integration time
IUL	Upper integration limit
ILL	Lower integration limit
LowerLimit	Lower PID Limit
UpperLimit	Upper PID limit
Deadband	Width of the deadband for the P+I+D correction value

Here, the PID operation is expressed as follows:

$$\frac{Y}{X} \cdot Kp + \frac{Ki}{Ti \cdot S} = Kd \cdot Td \cdot S$$

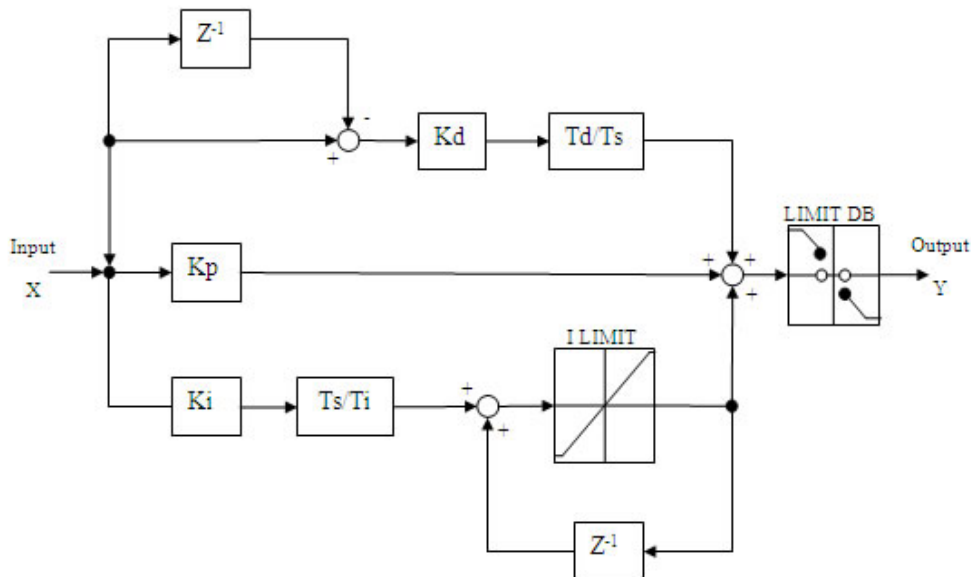
$$\frac{Y}{X} = Kp + Kd \cdot Td \cdot S$$

X: deviation input value; Y: output value

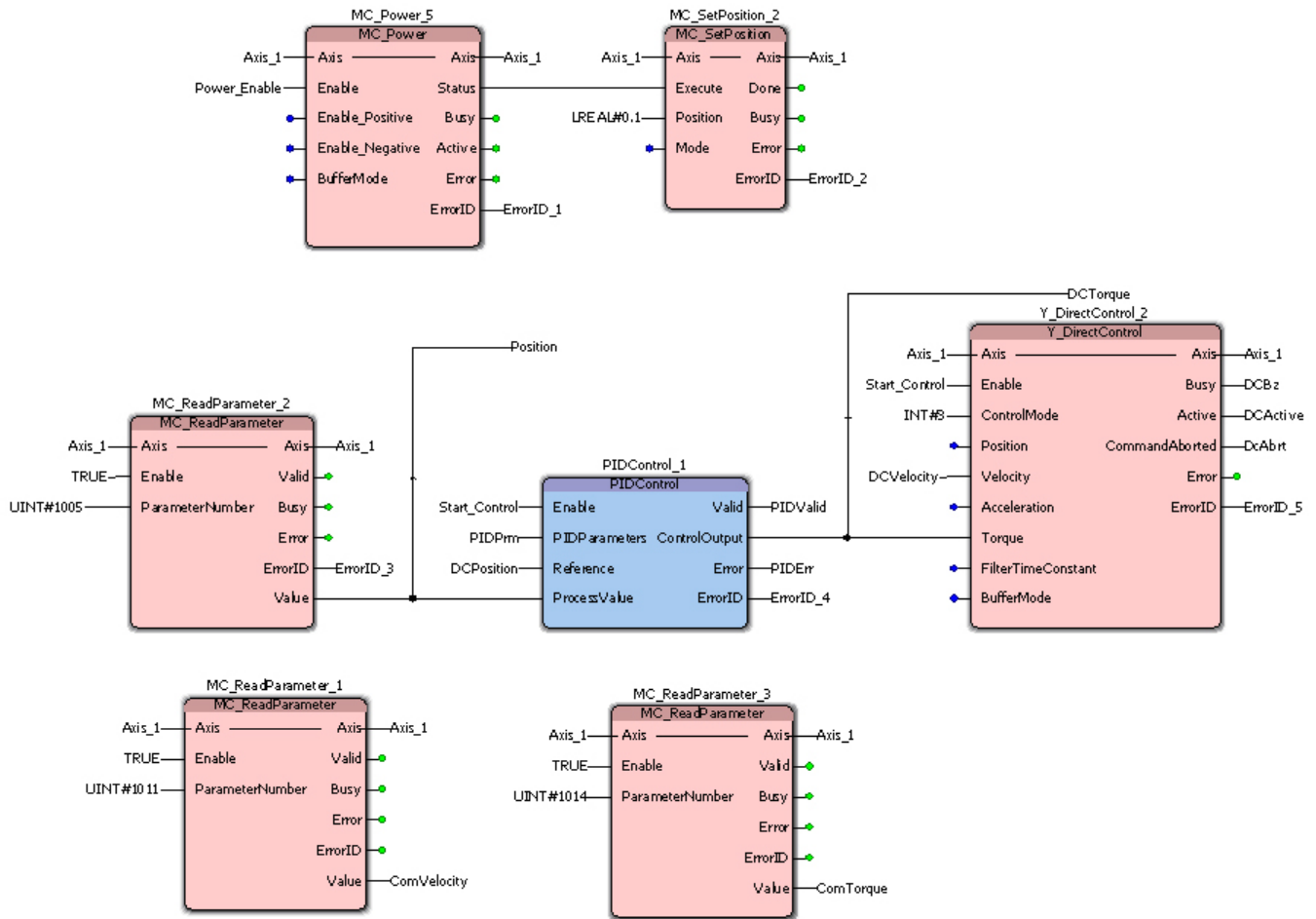
The following operation is performed within the PID instruction:

$$Y = Kp \cdot X + \left\{ \frac{Ki \cdot X + IREM}{\frac{Ti}{Ts}} + Yi' \right\} + Kd \cdot (X - X') \cdot \frac{Td}{Ts}$$

X': previous input value; Yi': previous I output value; Ts: scan time set value

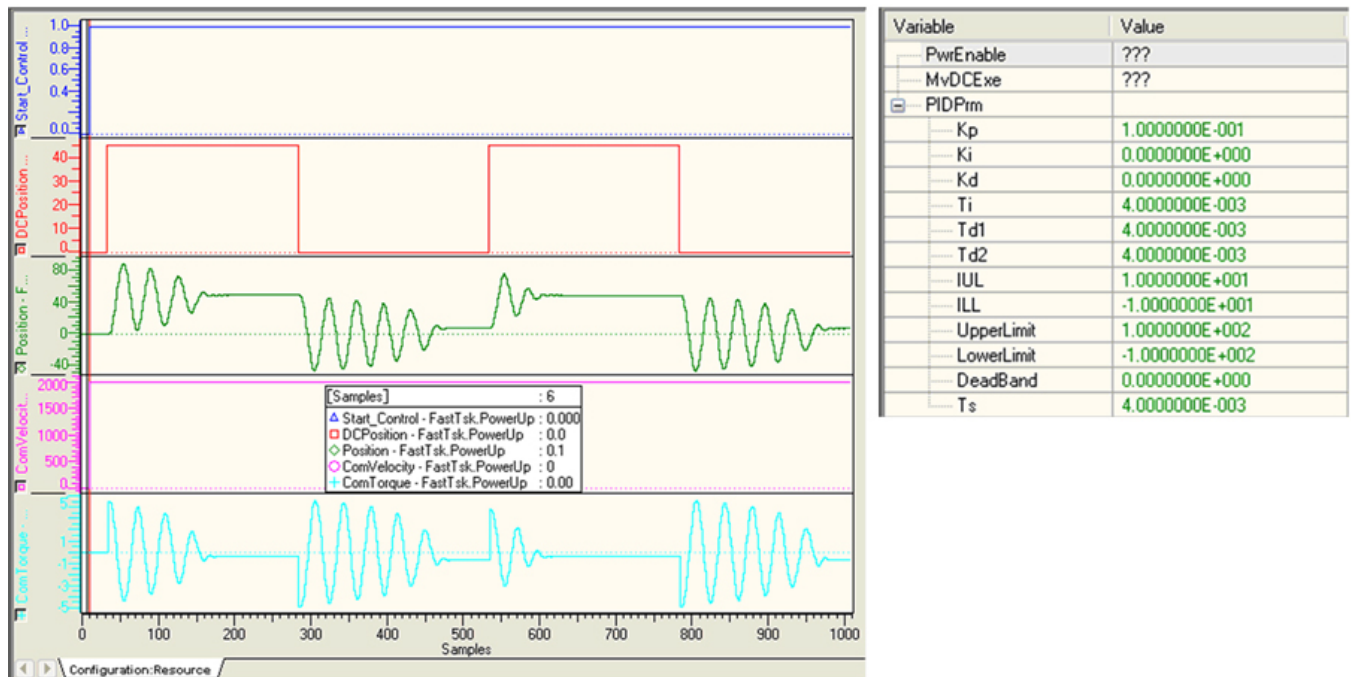


1. An example controlling a servo in torque mode:

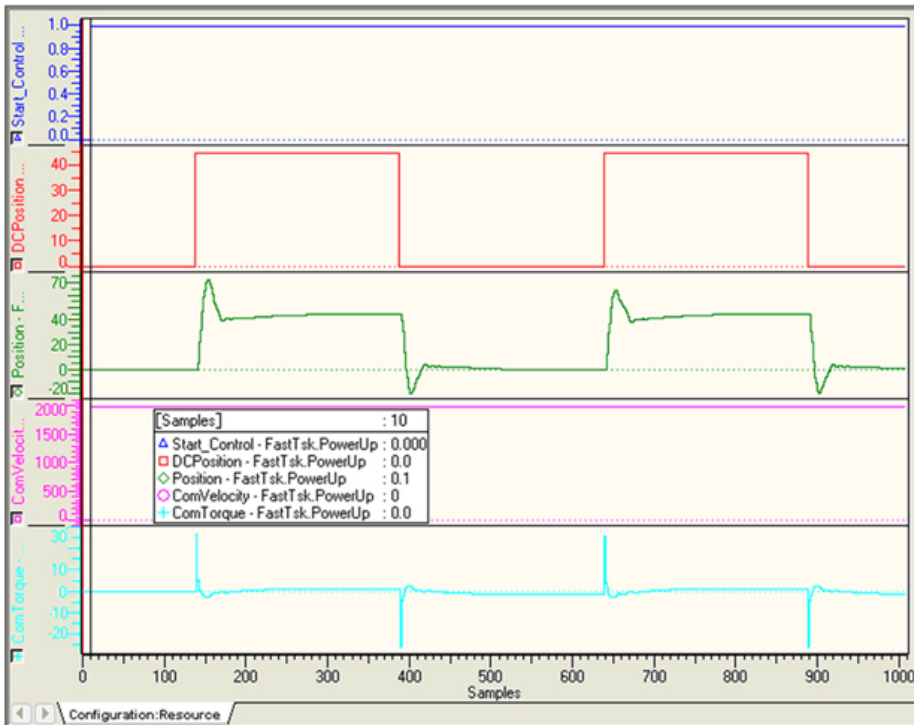


The following series of graphs show changes made to the PID gains to minimize error:

a. Proportional Control Only. Severe oscillation:

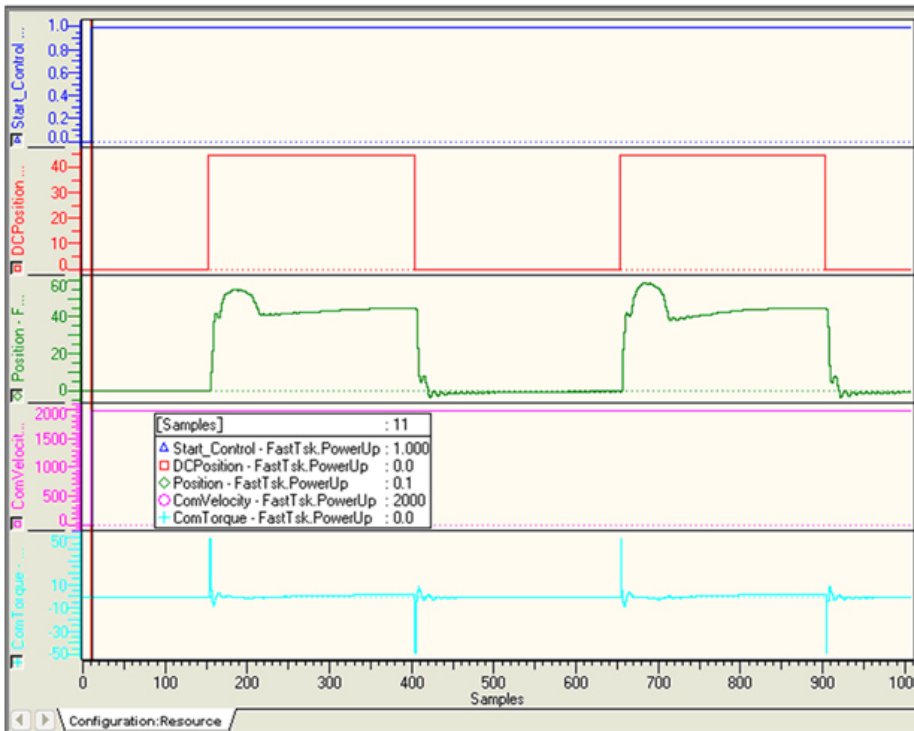


b. PID Control. Derivative helps to control oscillation:



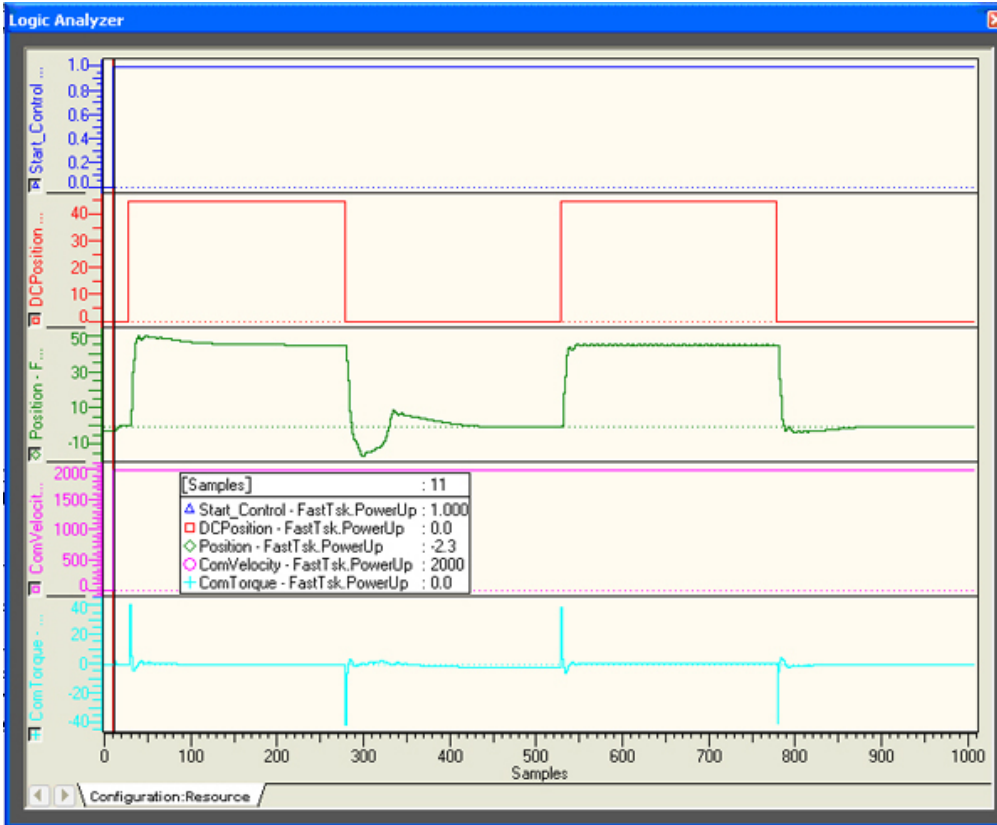
Variable	Value
PwrEnable	???
MvDCExe	???
PIDPrm	
Kp	1.000000E-001
Ki	1.000000E-002
Kd	5.000000E-001
Ti	4.000000E-003
Td1	4.000000E-003
Td2	4.000000E-003
IUL	1.000000E+001
ILL	-1.000000E+001
UpperLimit	1.000000E+002
LowerLimit	-1.000000E+002
DeadBand	0.000000E+000
Ts	4.000000E-003

c. PID Control - Increasing the derivative gain:



Variable	Value
PwrEnable	???
MvDCExe	???
PIDPrm	
Kp	1.000000E-001
Ki	1.000000E-002
Kd	1.000000E+000 *
Ti	4.000000E-003
Td1	4.000000E-003
Td2	4.000000E-003
IUL	1.000000E+001
ILL	-1.000000E+001
UpperLimit	1.000000E+002
LowerLimit	-1.000000E+002
DeadBand	0.000000E+000
Ts	4.000000E-003

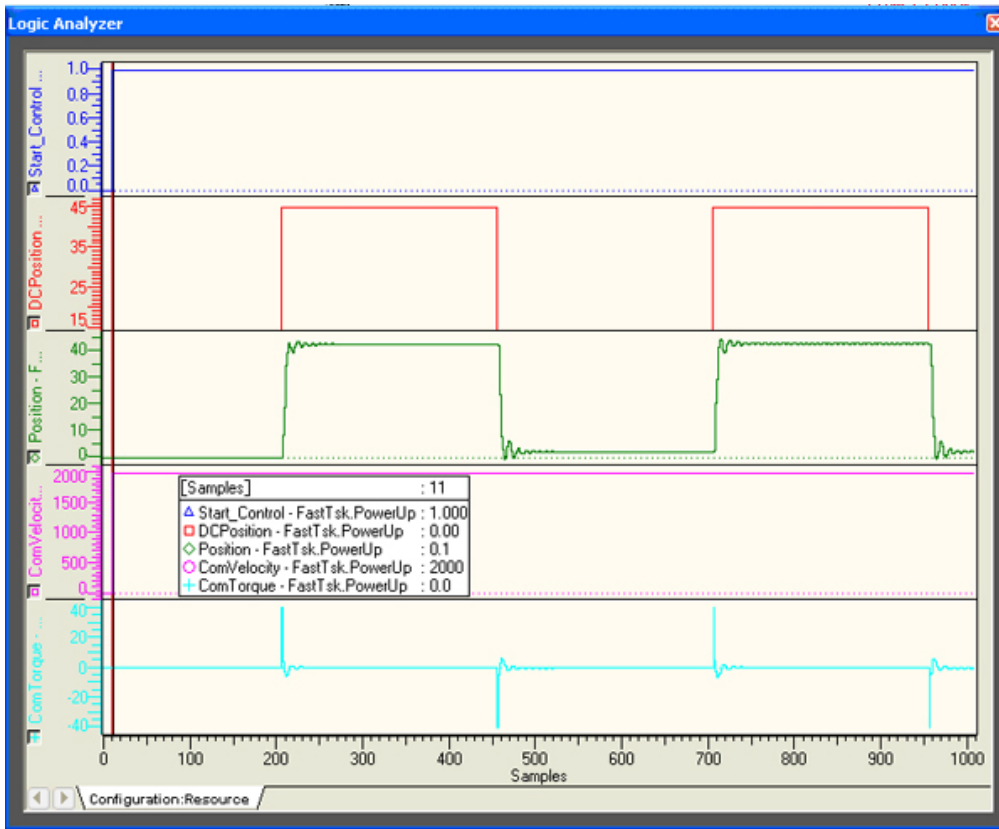
d. Further increase in the derivative gain:



The Watch Window displays a table of variables and their values. The variables are listed in the first column, the values in the second column, and the default values in the third column. The type of each variable is listed in the fourth column.

Variable	Value	Default value	Type
PwEnable	???		
MvDCExe	???		
PIDPtm			PIDStruct
Kp	1.0000000E-001		LREAL
Ki	1.0000000E-002		LREAL
Kd	8.0000000E-001		LREAL
Ti	4.0000000E-003		LREAL
Td1	4.0000000E-003		LREAL
Td2	4.0000000E-003		LREAL
IUL	1.0000000E+001		LREAL
ILL	-1.0000000E+001		LREAL
UpperLimit	1.0000000E+002		LREAL
LowerLimit	-1.0000000E+002		LREAL
DeadBand	0.0000000E+000		LREAL
Ts	4.0000000E-003		LREAL

e. PD Control – Integral gain is set to zero, which is best suited for this example.



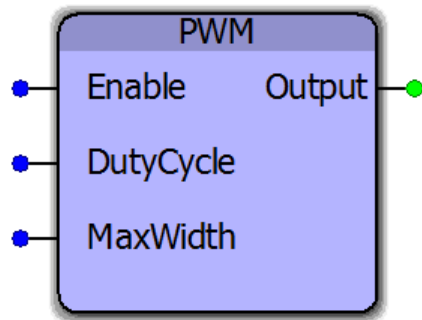
Watch Window

Variable	Value	Default value	Type
PwEnable	???		
MvDCExe	???		
PIDPrm			PIDStruct
Kp	1.0000000E-001		LREAL
Ki	0.0000000E+000		LREAL
Kd	8.0000000E-001		LREAL
Ti	4.0000000E-003		LREAL
Td1	4.0000000E-003		LREAL
Td2	4.0000000E-003		LREAL
IUL	1.0000000E+001		LREAL
ILL	-1.0000000E+001		LREAL
UpperLimit	1.0000000E+002		LREAL
LowerLimit	-1.0000000E+002		LREAL
DeadBand	0.0000000E+000		LREAL
Ts	4.0000000E-003		LREAL

Watch 1 Watch 2 Watch 3 Watch 4 Watch 5 Watch 6 Watch 7 Watch 8 Watch 9



PWM



This function block can be used for generally lower performance PWM requirements. For higher performance PWM applications, consider a SLIO PWM remote I/O module.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	DutyCycle	UINT	DutyCycle / MaxWidth indicates the percentage of time the output will be on.	UNIT #0
V	MaxWidth	UINT	The maximum DutyCycle that can be applied. The task interval in which this function block is executed will dictate the cycle period.	UINT #0
VAR_OUTPUT				
V	Output	BOOL	The output value.	

Notes

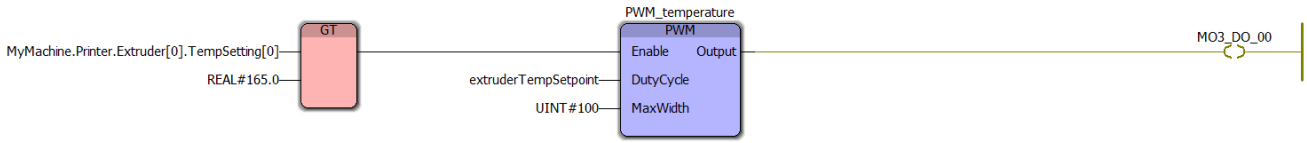
This functions performance is governed by the task interval in which it is executed. For example, if this function block is executed in a 50 mSec task and MaxWidth is set to 250, the total cycle duration is $50 \text{ mSec} * 250 = 12.5 \text{ seconds}$ with a resolution of 50 mSec. If this function block is executed in a fast task, such as 1 mSec with a MaxWidth of 100, the output could be controlled with a resolution of 1% and updated 10 times per second.

Errors

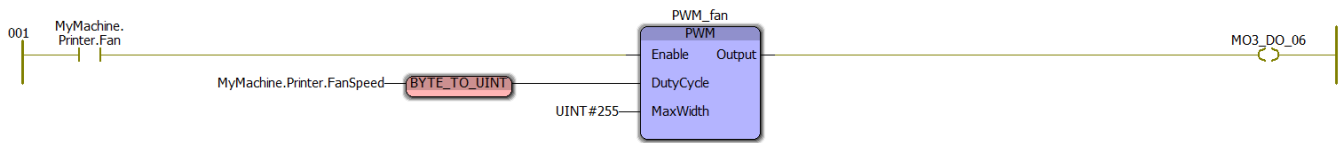
This function will not produce any errors.

Example

(*Extruder PWM Control*)

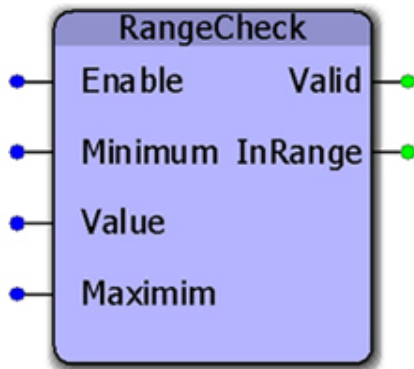


(*Fan PWM Control*)





RangeCheck



This function block will set the output 'InRange' if the Value input is between the Minimum and Maximum. The check is inclusive, meaning that if Value=Minimum or Value=Maximum, then the InRange output will be on.

Library

Yaskawa Toolbox

Parameters

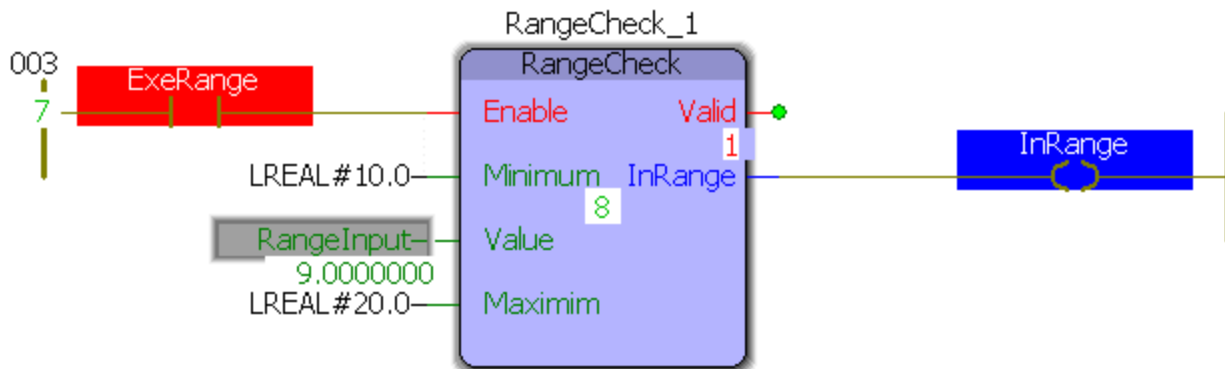
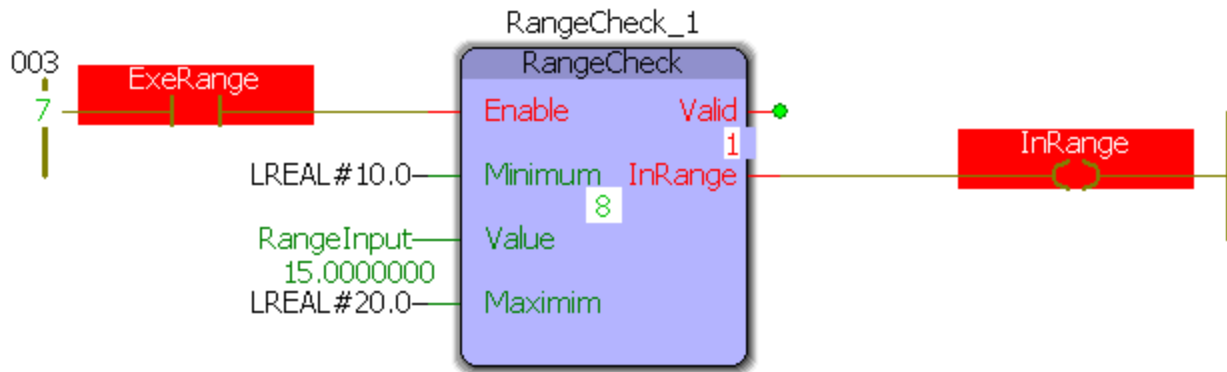
*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Minimum	LREAL	The smallest 'Value' that will set the InRange output high.	LREAL#0.0
V	Value	LREAL	The data to be tested against the Minimum and Maximum.	LREAL#0.0
V	Maximum	LREAL	The largest 'Value' that will set the InRange output high.	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	InRange	BOOL	Indicates if the Value is between the Minimum and Maximum. (Inclusive)	

Error Description

No errors will be generated.

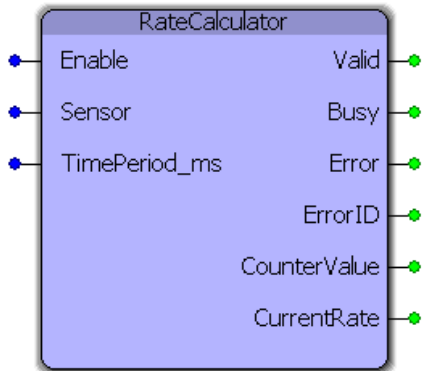
Example

ExeRange does not need to be toggled if Value is changed, as demonstrated below:





RateCalculator



This function block determines the frequency and number of occurrences of an event, such as determining the part output rate of a machine. RateCalculator counts the number of times an input 'Sensor' signal produces a rising edge and determines the frequency of that signal with respect to a chosen time period. It can account for real-time changes to the time period.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_INPUT			Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	Sensor	BOOL	Periodic signal to be measured. Commonly a "part-complete" sensor.
V	TimePeriod_ms	DINT	Sensor is measured with respect to this time window (milliseconds) to determine the current real-time rate.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.

*	Parameter	Data Type	Description
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	CounterValue	LREAL	Number of times 'Sensor' has measured a rising edge since the function block has been enabled.
V	CurrentRate	LREAL	The current frequency of the 'Sensor' input with respect to the chosen time period.

Notes

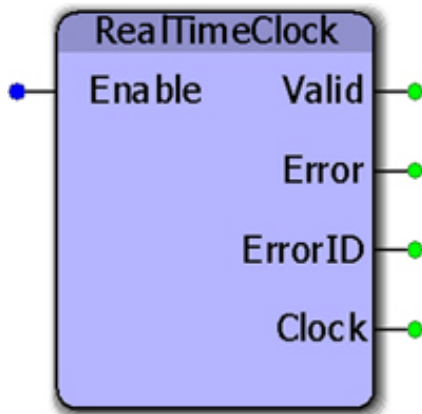
Upon enabling or a change of the time period, the 'Busy' signal remains active until the specified time period elapses, whereupon 'Busy' will go low and 'Valid' will go high. This is to receive a complete initial measurement of the rate 'Sensor' / 'TimePeriod_ms'.

Error Description

No errors will be generated.



RealTimeClock



This function block provides the controllers real time clock as an RTCStruct containing year, month, day, hour, minute, second, and millisecond. This function uses the RTC_S function, provided in the ProConOS firmware library, which returns the real time clock as a string.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	Clock	RTCStruct	Structure containing year, month, day, hour, minute, second, and millisecond.	

Notes

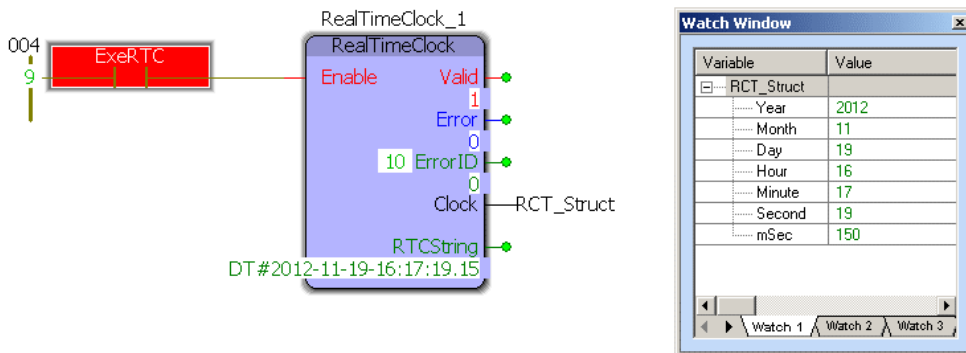
The controllers clock can be set from the web server, or by using the Y_SetRTC function block from the YMotion firmware library, which requires firmware version 2.0.0 or greater.

Error Description

No errors will be generated.

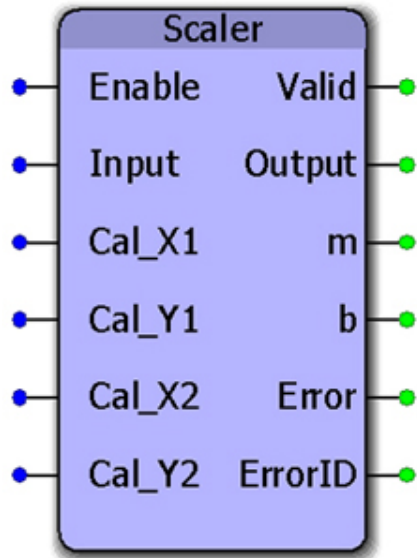
Example

The output of this block is continually updated as long as Enable is TRUE.





Scaler



This function block performs the calculation $y := mx + b$.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Input	LREAL	The x variable of $y := mx + b$.	LREAL#0.0
V	CalX1	LREAL	Datapoint specifying a line along which data is to be scaled.	LREAL#0.0
V	CalY1	LREAL	Datapoint specifying a line along which data is to be scaled.	LREAL#0.0
V	CalX2	LREAL	Datapoint specifying a line along which data is to be scaled.	LREAL#0.0
V	CalY2	LREAL	Datapoint specifying a line along which data is to be scaled.	LREAL#0.0
VAR_OUTPUT				

*	Parameter	Data Type	Description
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	Output	LREAL	The result of the calculation $y := mx + b$.
V	m	LREAL	The calculated slope of the line.
V	b	LREAL	The calculated intercept of the line.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

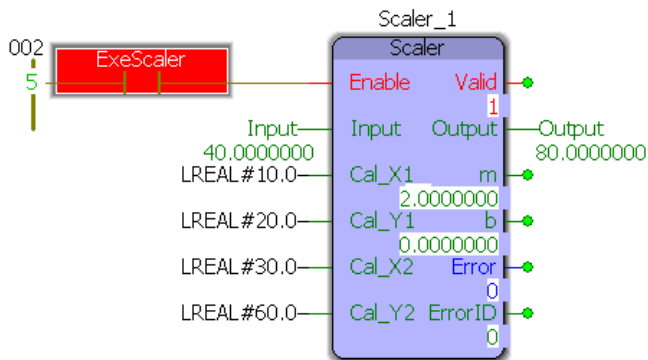
Notes

- This function can be used with temperature sensors or any analog value that must be adjusted before further processing takes place.
- m is determined by the slope of a line specified by Cal_X1, Cal_Y1, Cal_X2, Cal_Y2.
- x is the 'Input'
- b is determined by calculating the Y intercept of the line.

Error Description

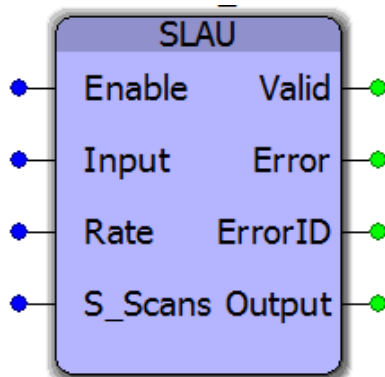
See the [Function Block ErrorID](#) list.

Example





SLAU



This function block generates an S-curve profile to the input value based on a moving average calculation. First, a slope is calculated based on the ramp input. Second, a moving average is applied to the ramp profile. The input value can be changed continuously on the fly.

Library

Yaskawa Toolbox

Parameters

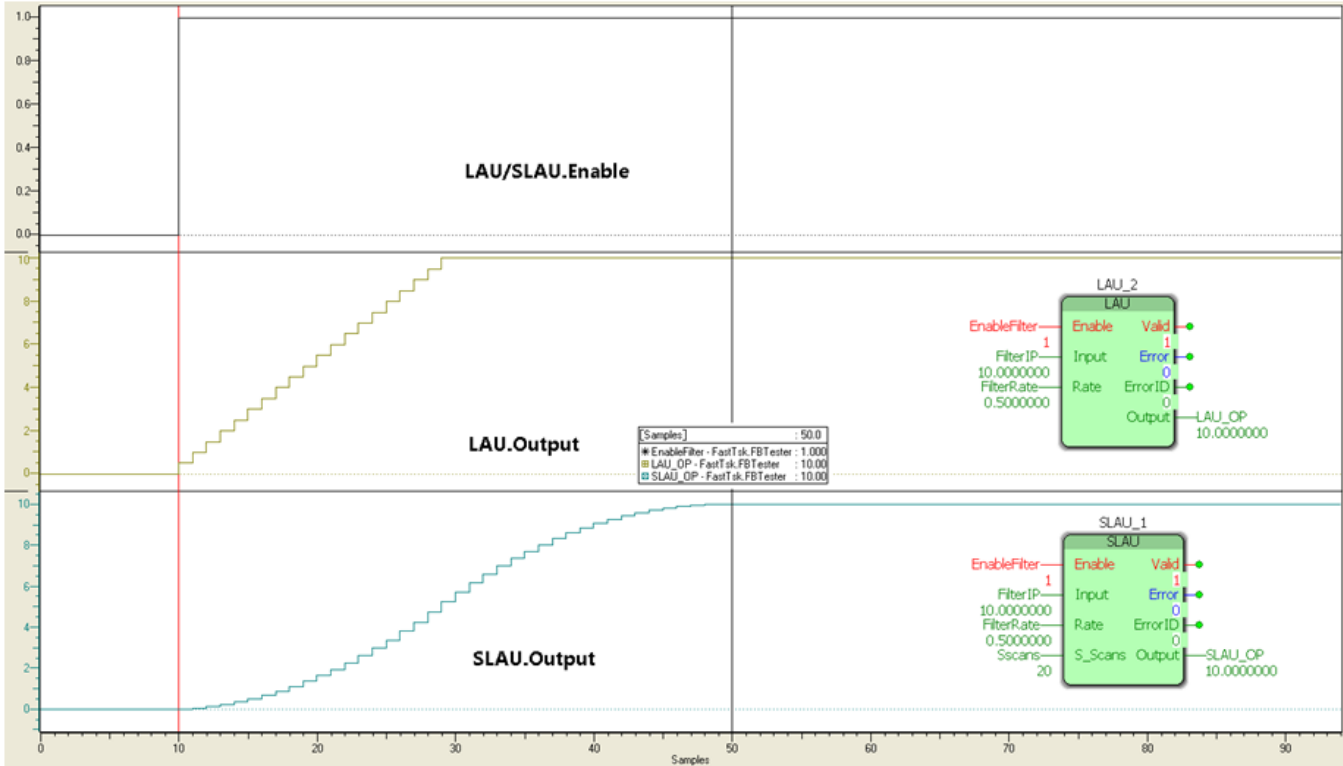
*	Parameter	Data Type	Description	Default
VAR_INPUT				
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Input	LREAL	Target Value.	LREAL#0.0
V	Rate	LREAL	Acceleration/Deceleration per scan. The time required for the Output to become the Input value profile depends on the Rate and S_Scans Inputs and the interval (Application task rate) at which the LAU function block is being run.	LREAL#0.0
V	S_Scans	UINT	Number of scans for the moving average calculation (S-Curve).	UINT#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
V	Output	LREAL	Target Value.	

Error Description

See the [Function Block ErrorID](#) list.

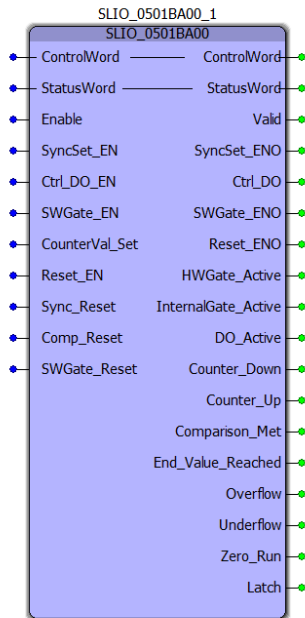
Example 1

An example of a step input converted to a smooth s-curve by the SLAU function block is shown below. The 0 to 10 unit step change is converted to a smooth s-curve profile with a 20 scan ramp and an additional 20 scan s-curve (moving average). Output of the LAU function block with a similar rate input is also shown.





SLIO_0501BA00



This function block breaks out the Status and Control words of the SLIO 050-1BA00 counter module into individual bits for simpler control of the counter functionality. For more information on functionality of the SLIO Counter modules, please see the User's Manual for the module.

At a minimum, the *Enable* and *SWGate_EN* inputs need to be TRUE to enable the counter function.

Library

Yaskawa Toolbox

Parameters

*	Parameter	DataType	Description
VAR_IN_OUT			
V	ControlWord	UINT	Control Word of the 050-1BA00 Counter Module (auto-generated by Hardware Configuration)
V	StatusWord	UINT	Status Word of the 050-1BA00 Counter Module (auto-generated by Hardware Configuration)
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.

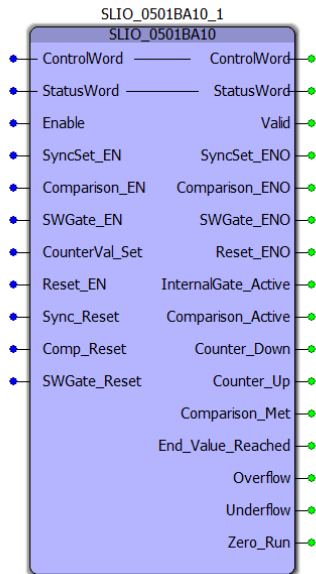
*	Parameter	DataType	Description
V	SyncSet_EN	BOOL	Activates the reset mode of the module
V	Ctrl_DO_EN	BOOL	Enables the comparison digital output of the module
V	SWGate_EN	BOOL	Sets the software gate of the module. It is required to set this value to TRUE to enable the counter function.
V	CounterVal_Set	BOOL	Sets counter temporarily to the value set in the module's IOx_Mxx_Set parameter
V	Reset_EN	BOOL	On rising edge, resets the following output bits: Comparison_Set; End_Value_Reached; Overflow; Underflow; Zero_Run
V	Sync_Reset	BOOL	Deactivates the reset mode
V	Comp_Reset	BOOL	Disables the comparison digital output.
V	SWGate_Reset	BOOL	Resets the software gate. This causes the module to stop counting input pulses.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	SyncSet_ENO	BOOL	Reset was active
V	Ctrl_DO	BOOL	Is TRUE when digital output is enabled.
V	SWGate_ENO	BOOL	Is TRUE when software gate is active (counter function enabled)
V	Reset_ENO	BOOL	Is TRUE when Reset_EN is active.
V	HWGate_Active	BOOL	Is TRUE when hardware gate is active
V	InternalGate_Active	BOOL	Is TRUE when internal gate is active
V	DO_Active	BOOL	Is TRUE when digital counter output is on.
V	Counter_Down	BOOL	Is TRUE when counter value is increasing.
V	Counter_Up	BOOL	Is TRUE when counter value is decreasing.
V	Comparison_Met	BOOL	Is TRUE when comparison condition is met.
V	End_Value_Reached	BOOL	Is TRUE when End Value (set in module parameters) is reached.
V	Overflow	BOOL	Is TRUE when counter is at Overflow value.
V	Underflow	BOOL	Is TRUE when counter is at Underflow value.
V	Zero_Run	BOOL	Is TRUE when module is in Zero Run state.
V	Latch	BOOL	Is TRUE when latch input is triggered.

Error Description

> See the [Function Block ErrorID](#) list.



SLIO_0501BA10



This function block breaks out the Status and Control words of the SLIO 050-1BA00 counter module into individual bits for simpler control of the counter functionality. For more information on functionality of the SLIO Counter modules, please see the User's Manual for the module.

At a minimum, the *Enable* and *SWGate_EN* inputs need to be TRUE to enable the counter function.

Library

Yaskawa Toolbox

Parameters

* Parameter	Data Type	Description
VAR_IN_OUT		
V ControlWord	UINT	Control Word of the 050-1BA00 Counter Module (auto-generated by Hardware Configuration)
V StatusWord	UINT	Status Word of the 050-1BA00 Counter Module (auto-generated by Hardware Configuration)
VAR_INPUT		

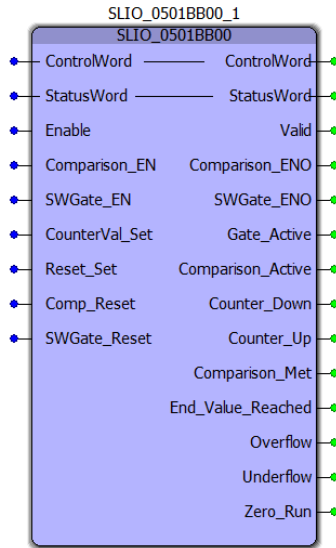
* Parameter	Data Type	Description
B Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V SyncSet_EN	BOOL	Activates the reset mode of the module
V Comparison_EN	BOOL	Enables the comparison function of the module
V SWGate_EN	BOOL	Sets the software gate of the module. It is required to set this value to TRUE to enable the counter function.
V CounterVal_Set	BOOL	Sets counter temporarily to the value set in the module's IOx_Mxx_Set parameter
V Reset_EN	BOOL	On rising edge, resets the following output bits: Comparison_Set; End_Value_Reached; Overflow; Underflow; Zero_Run
V Sync_Reset	BOOL	Deactivates the reset mode
V Comp_Reset	BOOL	Disables the comparison digital output.
V SWGate_Reset	BOOL	Resets the software gate. This causes the module to stop counting input pulses.
VAR_OUTPUT		
B Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V SyncSet_ENO	BOOL	Reset was active
V Comparison_ENO	BOOL	Is TRUE when Comparison_EN input is TRUE.
V SWGate_ENO	BOOL	Is TRUE when software gate is active (counter function enabled)
V Reset_ENO	BOOL	Is TRUE when Reset_EN is active.
V InternalGate_Active	BOOL	Is TRUE when internal gate is active
V Comparison_Active	BOOL	Is TRUE when the module's comparison function is active.
V Counter_Down	BOOL	Is TRUE when counter value is increasing.
V Counter_Up	BOOL	Is TRUE when counter value is decreasing.
V Comparison_Met	BOOL	Is TRUE when comparison condition is met.
V End_Value_Reached	BOOL	Is TRUE when End Value (set in module parameters) is reached.
V Overflow	BOOL	Is TRUE when counter is at Overflow value.
V Underflow	BOOL	Is TRUE when counter is at Underflow value.
V Zero_Run	BOOL	Is TRUE when module is in Zero Run state.

Error Description

See the [Function Block ErrorID](#) list.



SLIO_0501BB00



This function block breaks out the Status and Control words of the SLIO 05011BB00 counter module into individual bits for simpler control of the counter functionality. For more information on functionality of the SLIO Counter modules, please see the User's Manual for the module.

At a minimum, the *Enable* and *SWGate_EN* inputs need to be TRUE to enable the counter function.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	ControlWord	UINT	Control Word of the 050-1BA00 Counter Module (auto-generated by Hardware Configuration)
V	StatusWord	UINT	Status Word of the 050-1BA00 Counter Module (auto-generated by Hardware Configuration)
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V	Comparison_EN	BOOL	Activates the comparison mode of the module

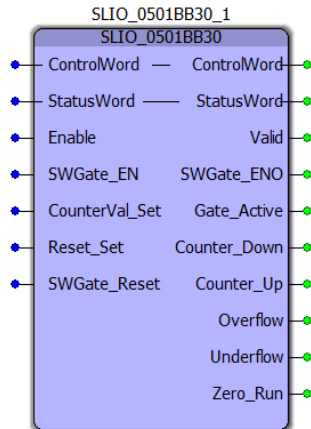
* Parameter	Data Type	Description
V SWGate_EN	BOOL	Sets the software gate of the module. It is required to set this value to TRUE to enable the counter function.
V CounterVal_Set	BOOL	Sets counter temporarily to the value set in the module's IOx_Mxx_Set parameter
V Reset_Set	BOOL	On rising edge, resets the following output bits: Comparison_Set; End_Value_Reached; Overflow; Underflow;Zero_Run
V Comp_Reset	BOOL	Disables the comparison digital output.
V SWGate_Reset	BOOL	Resets the software gate. This causes the module to stop counting input pulses.
VAR_OUTPUT		
B Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V Comparison_ENO	BOOL	Is TRUE when the Comparison_EN is TRUE.
V SWGate_ENO	BOOL	Is TRUE when software gate is active (counter function enabled)
V Gate_Active	BOOL	Is TRUE when internal gate is active
V Comparison_Active	BOOL	Is TRUE when the comparison function of the module is active.
V Counter_Down	BOOL	Is TRUE when counter value is increasing.
V Counter_Up	BOOL	Is TRUE when counter value is decreasing.
V Comparison_Met	BOOL	Is TRUE when comparison condition is met.
V End_Value_Reached	BOOL	Is TRUE when End Value (set in module parameters) is reached.
V Overflow	BOOL	Is TRUE when counter is at Overflow value.
V Underflow	BOOL	Is TRUE when counter is at Underflow value.
V Zero_Run	BOOL	Is TRUE when module is in Zero Run state.

Error Description

See the [Function Block ErrorID](#) list.



SLIO_0501BB30



This function block breaks out the Status and Control words of the SLIO 050-1BB30 counter module into individual bits for simpler control of the counter functionality. For more information on functionality of the SLIO Counter modules, please see the User's Manual for the module.

At a minimum, the *Enable* and *SWGate_EN* inputs need to be TRUE to enable the counter function.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
V	ControlWord	UINT	Control Word of the 050-1BB30 Counter Module (auto-generated by Hardware Configuration)
V	StatusWord	UINT	Status Word of the 050-1BB30 Counter Module (auto-generated by Hardware Configuration)
VAR_INPUT			
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.

*	Parameter	Data Type	Description
V	SWGate_EN	BOOL	Sets the software gate of the module. It is required to set this value to TRUE to enable the counter function.
V	CounterVal_Set	BOOL	Sets counter temporarily to the value set in the module's IOx_Mxx_Set parameter
V	Reset_EN	BOOL	On rising edge, resets the following output bits: Comparison_Set; End_Value_Reached; Overflow; Underflow; Zero_Run
V	SWGate_Reset	BOOL	Resets the software gate. This causes the module to stop counting input pulses.
VAR_OUTPUT			
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V	SWGate_ENO	BOOL	Is TRUE when software gate is active (counter function enabled)
V	Gate_Active	BOOL	Is TRUE when internal gate is active
V	Counter_Down	BOOL	Is TRUE when counter value is increasing.
V	Counter_Up	BOOL	Is TRUE when counter value is decreasing.
V	Overflow	BOOL	Is TRUE when counter is at Overflow value.
V	Underflow	BOOL	Is TRUE when counter is at Underflow value.
V	Zero_Run	BOOL	Is TRUE when module is in Zero Run state.

Error Description

See the [Function Block ErrorID](#) list.



SLIO_0501BB40



This function block breaks out the Status and Control words of the SLIO 050-1BB40 frequency measurement module into individual bits for simpler control of the counter functionality. For more information on functionality of the SLIO Frequency Measurement modules, please see the User's Manual for the module.

At a minimum, the *FM_Start* input need to be TRUE to enable the counter function.

Library

Yaskawa Toolbox

Parameters

* Parameter	Data Type	Description
VAR_IN_OUT		
V ControlWord	UINT	Control Word of the 050-1BA00 Counter Module (auto-generated by Hardware Configuration)
V StatusWord	UINT	Status Word of the 050-1BA00 Counter Module (auto-generated by Hardware Configuration)
VAR_INPUT		
B Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.
V FM_Start	BOOL	Starts the frequency measurement function of the module.
V FM_Stop	BOOL	Stops the frequency measurement function of the module.
VAR_OUTPUT		
B Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.
V FM_Active	BOOL	If TRUE, the frequency measurement function is active.

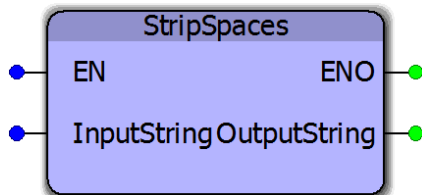
Error Description

See the [Function Block ErrorID](#) list.

StripSpaces



StripSpaces



This function block will copy an InputString to the OutputString while omitting all space characters (ASCII 32 decimal).

Library

Yaskawa Toolbox

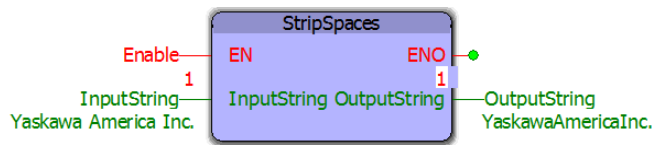
Parameters

	Parameter	Data Type	Description	
VAR_INPUT				Default
B	EN	BOOL	This function will continue to update while EN is held high.	FALSE
V	InputString	STRING	The input string that will be modified to not contain any spaces.	' '
VAR_OUTPUT				
B	ENO	BOOL	High if the function is executing normally.	
V	OutputString	STRING	Modified output of the string that was input to the function block.	

Notes

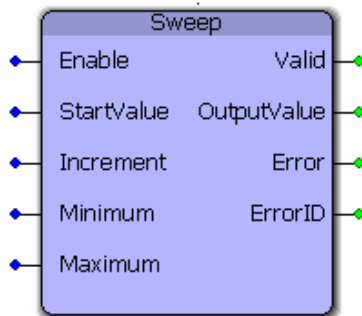
- The OutputString is only valid if ENO is high. If for some reason the conversion had an error, ENO will not be set.
- The PROCONOS firmware library must be added to the project, otherwise, the compiler error "A function block POU "STRING_TO_BUF" is referencing a sub function, but the library containing the function is not included."

Example





Sweep



This function block generates an output that rises and falls between the minimum and maximum outputs specified by the inputs. The OutputValue is the changed by the Increment input. This function block is useful for testing purposes by forcing other portions of application code to be tested with a full range of expected values.

Library

Yaskawa Toolbox

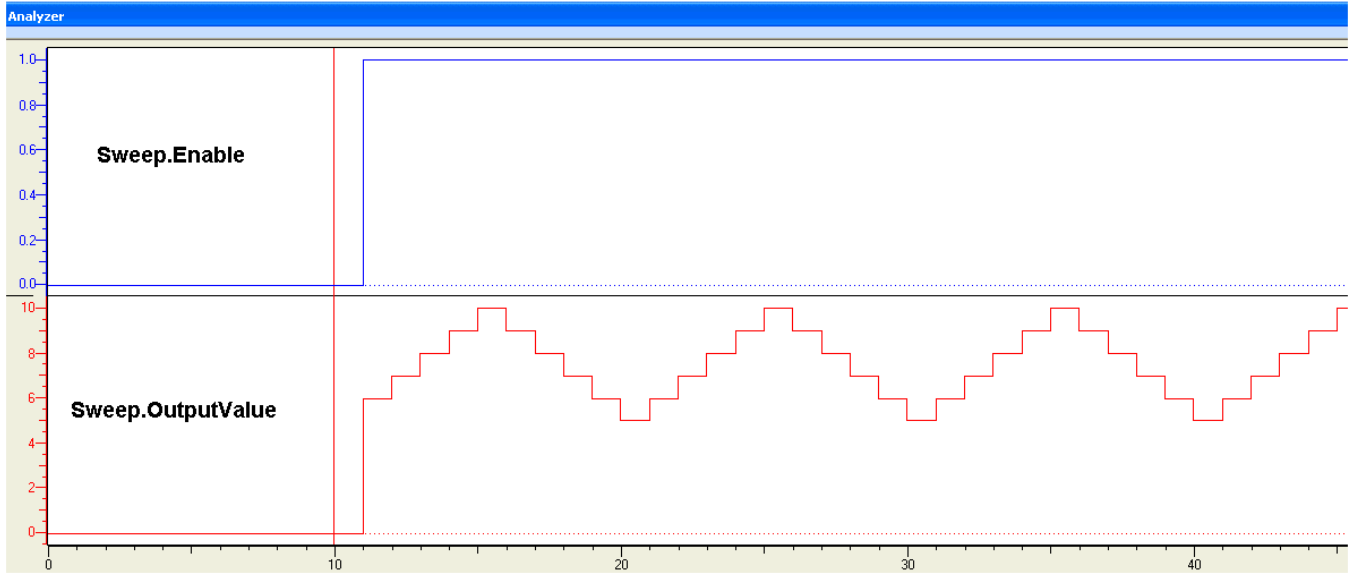
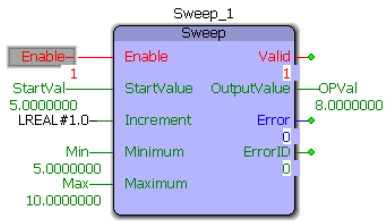
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	StartValue	LREAL	The OutputValue will start from this value.	LREAL#0.0
V	Increment	LREAL	The amount by which the Outputvalue is changed each scan.	LREAL#0.0
V	Minimum	LREAL	The minimum value output.	LREAL#0.0
V	Maximum	LREAL	The maximum value output.	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	OutputValue	LREAL	The output of the function.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Error Description

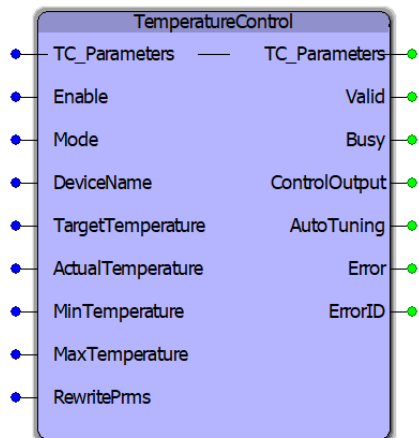
No errors will be generated.

Example:





TemperatureControl



This function block contains an autotuning and temperature control solution which can be incorporated with a range of devices via analog or PWM connection.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	TC_Parameters	AutoTuneStruct	Structure containing the parameters for operating the Temperature Control function.	
VAR_INPUT				
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	Default FALSE
V	Mode	INT	0: Normal operation. The function block looks for a previously stored DeviceName file and runs. If the file is not found, then an Autotuning process starts. Upon successful autotuning, the system runs normally. 1: Force autotuning before running, even if the DeviceName file already exists.	INT#0
V	DeviceName	STRING	The name of the device being controlled. This will be used when writing the the TC_Parameters to a file on the controller flash file system.	N/A
V	TargetTemperature	LREAL	The target temperature value.	LREAL#0.0

V	ActualTemperature	LREAL	The current temperature value.	LREAL#0.0
V	MinTemperature	LREAL	Minimum allowable temperature. An Error will be generated if the ActualTemperature goes below this value and the ControlOutput will be set to zero. If unconnected or set to zero, an operating value of -200.0 will be assumed.	LREAL#-200.0
V	MaxTemperature	LREAL	Maximum allowable temperature. An Error will be generated if the ActualTemperature exceeds this value and the ControlOutput will be set to zero. If unconnected or set to zero, an operating value of 400.0 will be assumed.	LREAL#400.0
V	RewritePrms	BOOL	Rewrites the TC_Parameters to the DeviceName file to update any changes made manually after the autotuning process.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid. This block does not set the Valid output until autotuning has been completed.	
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)	
V	ControlOutput	LREAL	Output value from the PID control block. The range of values will be governed by the PID parameters, especially the upper and lower limit.	
B	Autotuning	BOOL	Indicates if the function is performing autotuning.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

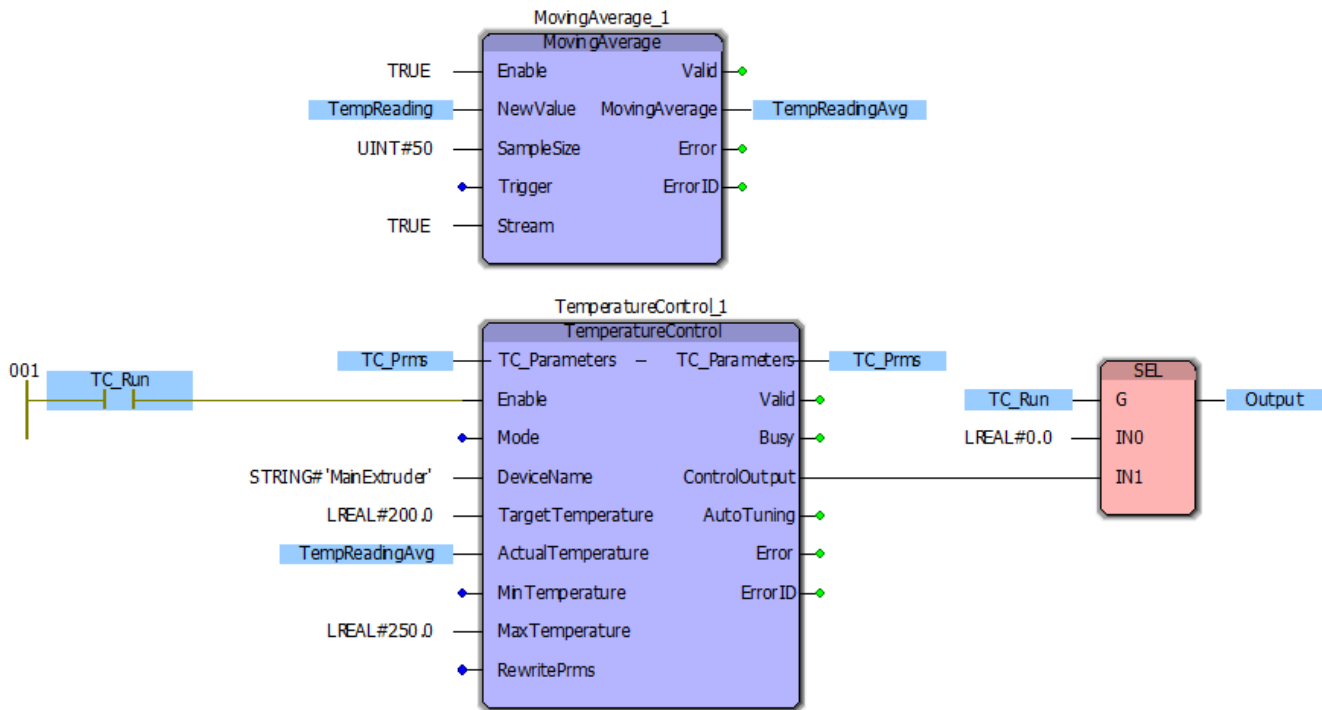
Notes

- It may be necessary to add external filtering to the ControlOutput to accommodate certain heaters such as relays.
- The autotuning feature can achieve 90-95% of the 'most ideal' performance.
 - The TC_Parameter values can be further adjusted manually if better performance is required.
- If loading autotuning parameters from a file, the DeviceName must be the same as the filename on the controller.
 - DeviceName cannot be left empty.
- RewritePrms must be toggled for every new write. When toggled, it will write the TC_Parameters the file as long as tuning has been completed.
- If TC_Parameters.ResponseType is left at 0, an operating value of 2 (medium response) is assumed.
- It may be necessary to add a moving average filter on the ActualTemperature input to reduce noise and result in more accurately calculated PID parameters.
- If the heater can only be controlled by turning it fully on or fully off, then it would be appropriate to place a PWM function block attached to the ControlOutput with a duty cycle of 100.

See the [TemperatureControl eLearning Module](#) on Yaskawa's YouTube Channel.

Fine Adjustments for Increased Performance

Function Block Setup



Initialization of Variables

Mode := INT#0; (* INT#0 = Run without calculating gains, INT#1 = Calculate gains and run *)

MinTemperature := LREAL#0.0; (* The MinTemperature will assume a value of -200.0 *)

Initialization of AutoTuneParameters

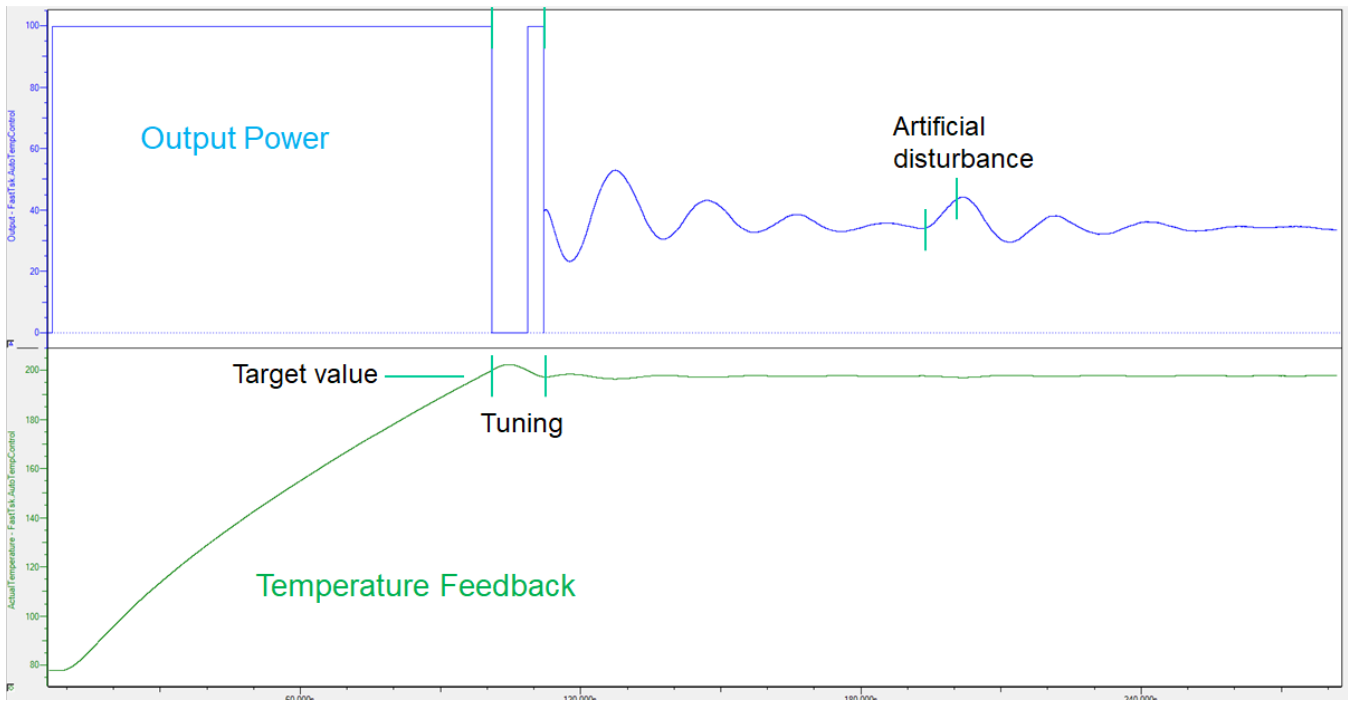
TC_Parameters.AutotuneCycles := INT#1; (* The number of oscillations to calculate gains for *)

TC_Parameters.IgnoreCycles := INT#0; (* The number of oscillations to ignore before calculating gains *)

TC_Parameters.ResponseType := INT#3; (* Response type: Slow (1), Medium (2), Fast (3) *)

Analysis of the Response

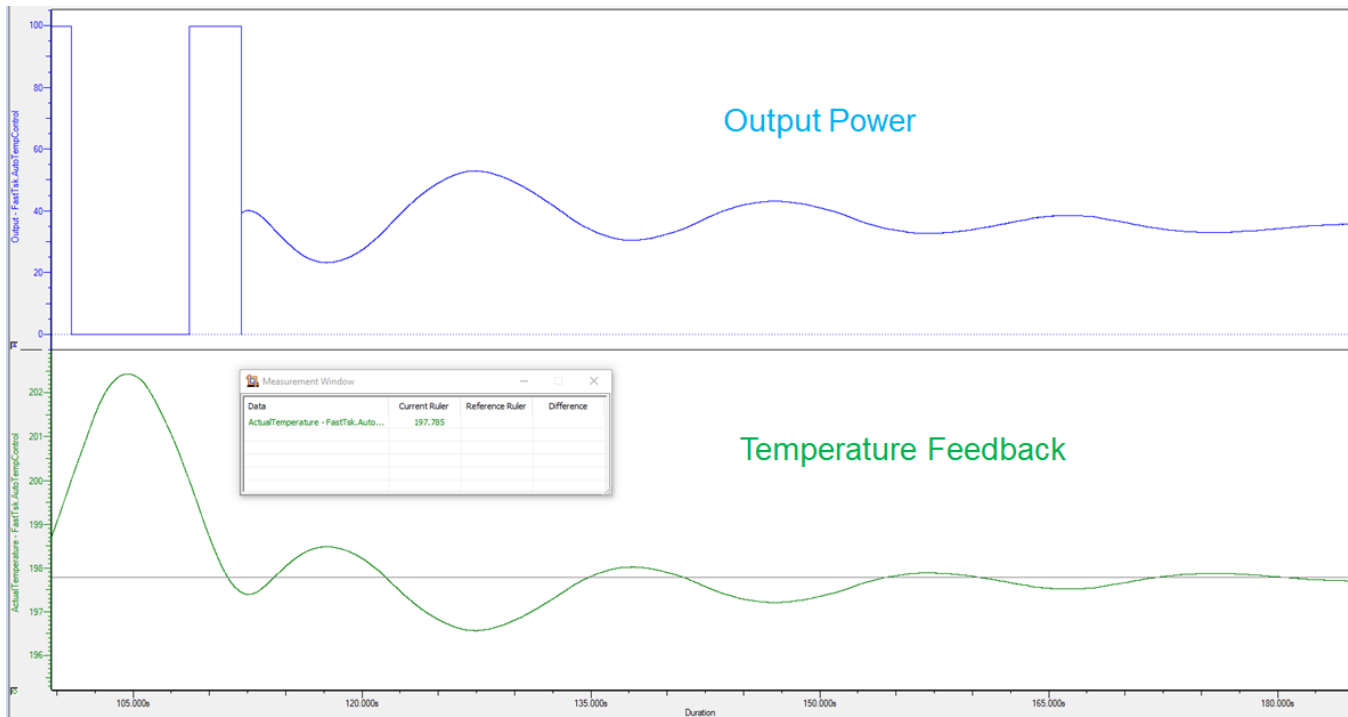
The following graph shows the output power vs temperature value for the rise time, tuning period, settling time, and then the ControlOutput compensation when the extruder was intentionally cooled with a fan (artificial disturbance).



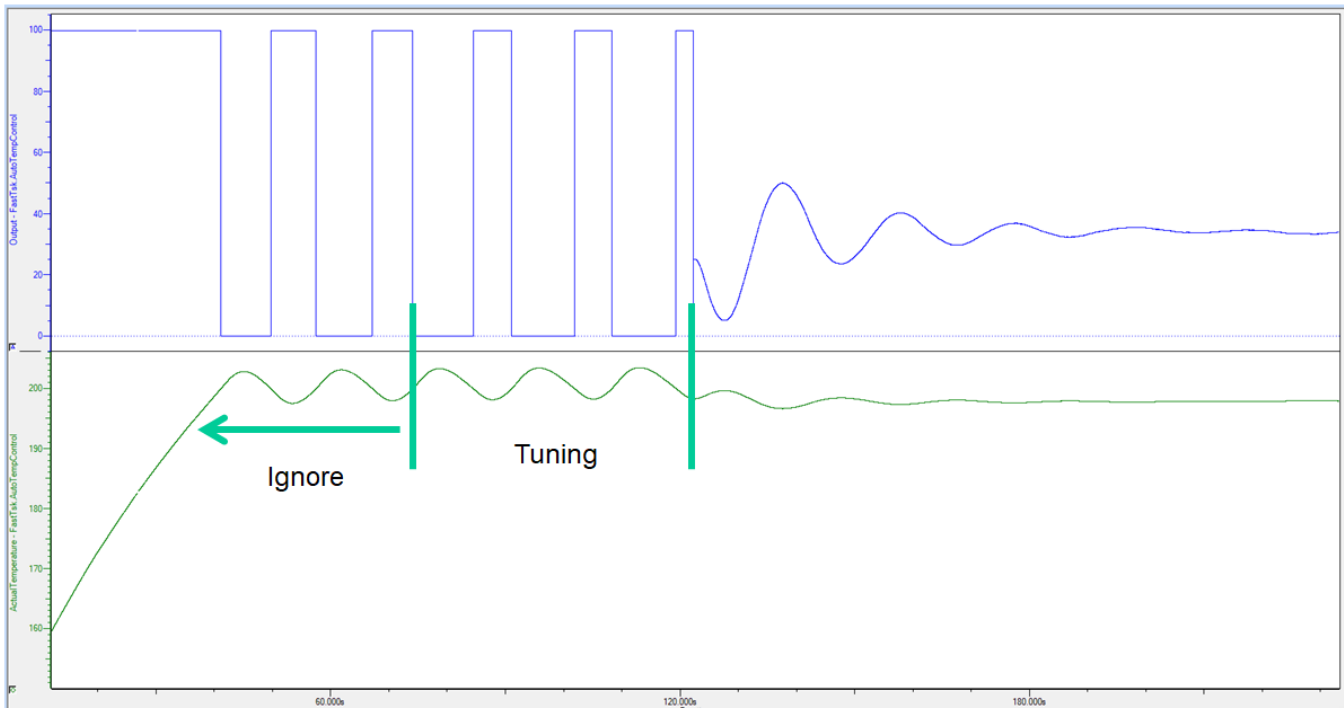
The calculated gains were as follows:

- $K_p = 15.5$
- $K_i = 0.1$
- $K_d = 0.5$

The following graph shows that it took 11 seconds to complete tuning and settled to 197.8 degrees in the following minute.

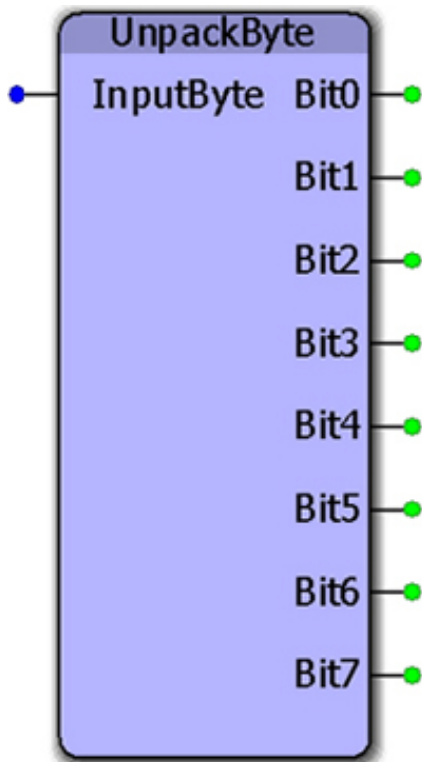


The following chart shows the tuning process when AutotuneCycles are set to 3 and IgnoreCycles are set to 2. The calculated gains of each oscillation during tuning are averaged together.





UnpackByte



This function block converts a byte into discrete bits.

Library

Yaskawa Toolbox

Parameters

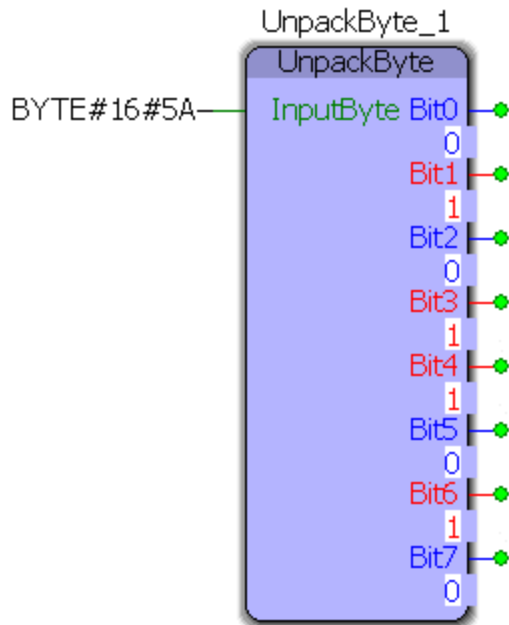
*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	InputByte	BYTE	The input data to be separated into bits.	BYTE#0
VAR_OUTPUT				
V	Bit0	BOOL	Bit 0 of the InputByte	
V	Bit1	BOOL	Bit 1 of the InputByte	
V	Bit2	BOOL	Bit 2 of the InputByte	
V	Bit3	BOOL	Bit 3 of the InputByte	

*	Parameter	Data Type	Description
V	Bit4	BOOL	Bit 4 of the InputByte
V	Bit5	BOOL	Bit 5 of the InputByte
V	Bit6	BOOL	Bit 6 of the InputByte
V	Bit7	BOOL	Bit 7 of the InputByte

Error Description

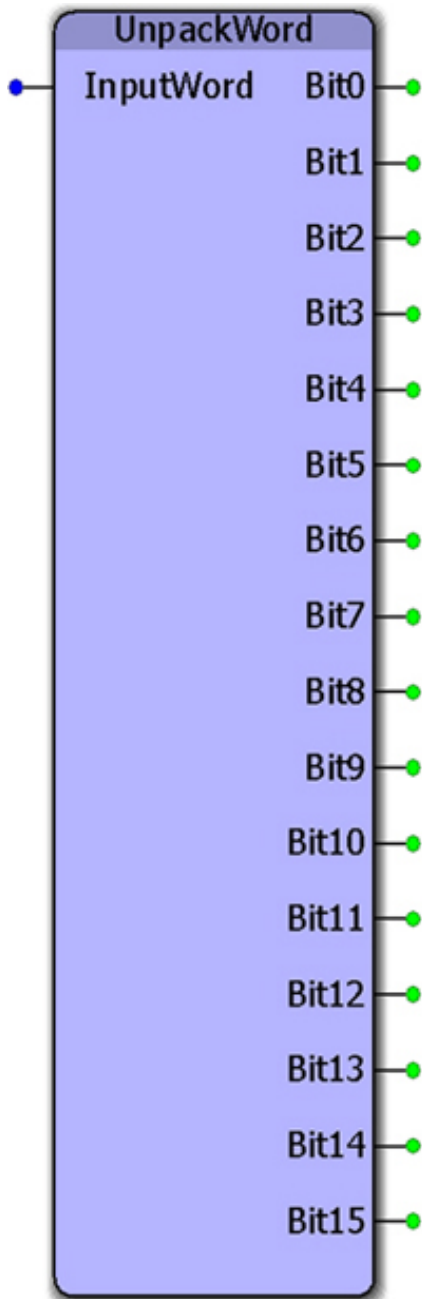
No errors will be generated.

Example





UnpackWord



This function block separates a word into individual bits.

Library

Yaskawa Toolbox

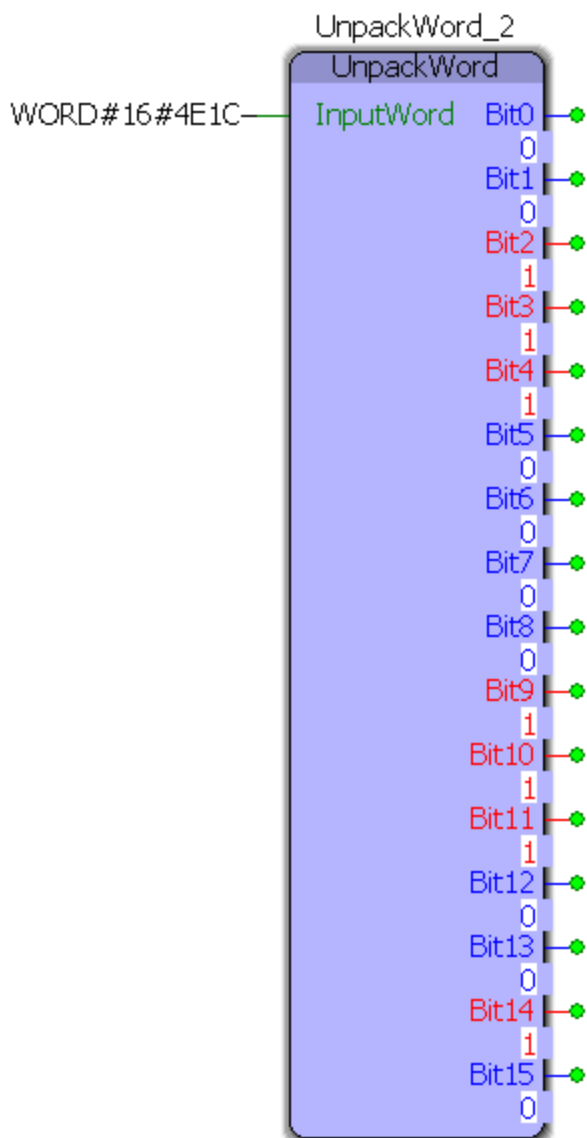
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	InputWord	WORD	The input data to be separated into bits.	WORD#0
VAR_OUTPUT				
V	Bit0	BOOL	Bit 0 of the InputWord	
V	Bit1	BOOL	Bit 1 of the InputWord	
V	Bit2	BOOL	Bit 2 of the InputWord	
V	Bit3	BOOL	Bit 3 of the InputWord	
V	Bit4	BOOL	Bit 4 of the InputWord	
V	Bit5	BOOL	Bit 5 of the InputWord	
V	Bit6	BOOL	Bit 6 of the InputWord	
V	Bit7	BOOL	Bit 7 of the InputWord	
V	Bit8	BOOL	Bit 8 of the InputWord	
V	Bit9	BOOL	Bit 9 of the InputWord	
V	Bit10	BOOL	Bit 10 of the InputWord	
V	Bit11	BOOL	Bit 11 of the InputWord	
V	Bit12	BOOL	Bit 12 of the InputWord	
V	Bit13	BOOL	Bit 13 of the InputWord	
V	Bit14	BOOL	Bit 14 of the InputWord	
V	Bit15	BOOL	Bit 15 of the InputWord	

Error Description

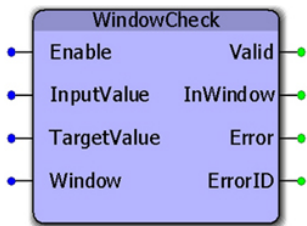
No errors will be generated.

Example





WindowCheck



This function block sets the InWindow output high if the InputValue is within +/- (Window/2) of the TargetValue. This function is useful when making a comparison that only relies on the InputValue to be close to the Target, but an exact match is not required.

Library

Yaskawa Toolbox

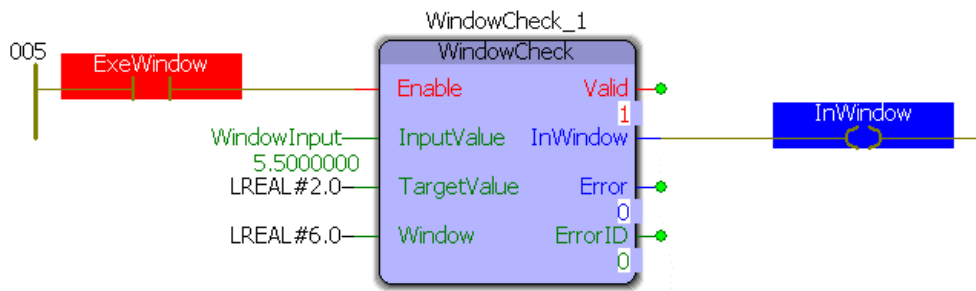
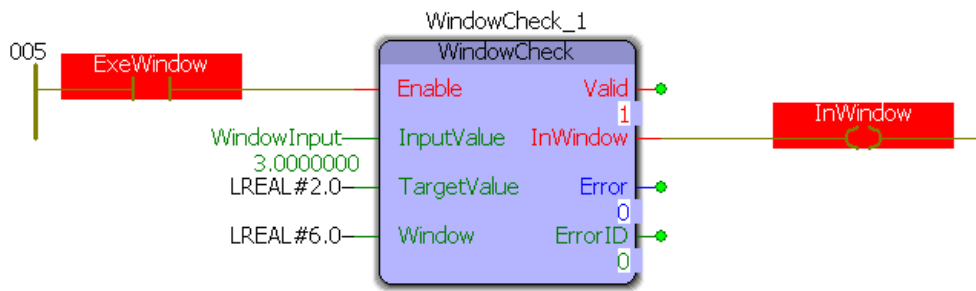
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	InputValue	LREAL	The data to be tested against the TargetValue	LREAL#0.0
V	TargetValue	LREAL	The desired data to be compared against.	LREAL#0.0
V	Window	LREAL	This amount will be divided in two. The InputValue must fall within half the window distance of the TargetValue for the InWindow output to go high. Window must be greater than zero.	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	InWindow	BOOL	Indicates that the InputValue is within the TargetValue +/- (Window/2) inclusive.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Error Description

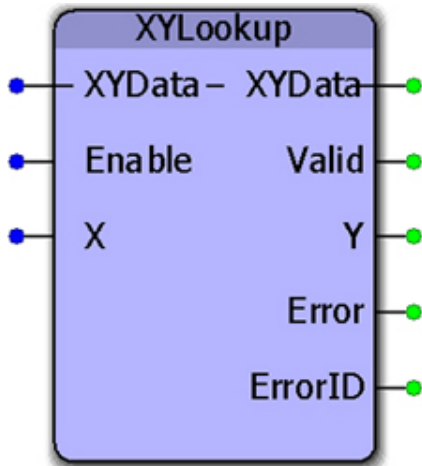
See the [Function Block ErrorID](#) list.

Example





XYLookup



This function block will do a binary search on the XYdata to find the X value, then output the corresponding Y value. This function will perform linear interpolation if the X value is between two data points in the XYData and calculate the appropriate Y value.

Library

Yaskawa Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	XYData	XYDataStruct	An array of X & Y data pairs	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	X	LREAL	The input reference	
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	Y	LREAL	The resulting output that relates the input.	
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

Works for sets where the X value is always increasing or always decreasing, but to use decreasing values of X requires v205 or higher.

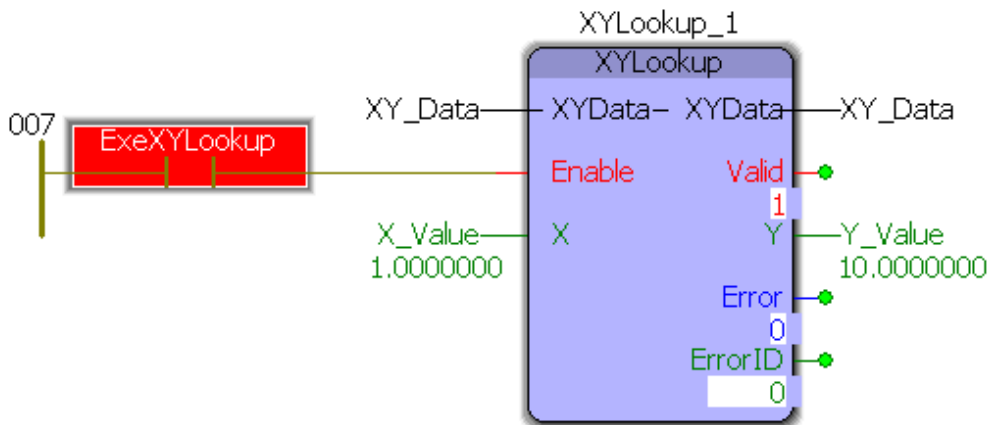
Error Description

See the [Function Block ErrorID](#) list.

Example

The XY_Data structure was initialized as:

```
1 1.0000000 XY_Data.Pair[0].X := LREAL#1.0;  
2 10.0000000 XY_Data.Pair[0].Y := LREAL#10.0;  
3 2.0000000 XY_Data.Pair[1].X := LREAL#2.0;  
4 20.0000000 XY_Data.Pair[1].Y := LREAL#20.0;  
5 3.0000000 XY_Data.Pair[2].X := LREAL#3.0;  
6 30.0000000 XY_Data.Pair[2].Y := LREAL#30.0;  
7 2 XY_Data.LastPair := INT#2;
```





Data Type: ExplicitData

For use with the [Explicit_Message](#) function block.

Data Type Declaration

TYPE

ExplicitData : ARRAY[0..503] OF BYTE;

END_TYPE



Data Type: MovingAverageArray

For use with the [MovingAverage](#) function block.

Data Type Declaration

TYPE

MovingAverageArray: ARRAY[0..30000] OF LREAL; (* Adjust the array size if more data elements are desired. *)

END_TYPE



Data Type: PIDStruct

Used with the [PIDControl](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyPIDStruct	PIDStruct		
U	Ts	LREAL	Sample time	MyPIDStruct.Ts
U	Kp	LREAL	Proportional Gain	MyPIDStruct.Kp
U	Ki	LREAL	Integral Gain	MyPIDStruct.Ki
U	Kd	LREAL	Derivative Gain	MyPIDStruct.Kd
U	Ti	LREAL	Integral Time (in Sec.)	MyPIDStruct.Ti
U	Td1	LREAL	Derivative Time for Divergent Inputs	MyPIDStruct.Td1
U	Td2	LREAL	Derivative Time for Convergent Inputs	MyPIDStruct.Td2
U	ILL	LREAL	Integral Lower Limit	MyPIDStruct.ILL
U	IUL	LREAL	Integral Upper Limit	MyPIDStruct.IUL
U	LowerLimit	LREAL	Lower Limit for ControlOutput	MyPIDStruct.LowerLimit
U	UpperLimit	LREAL	Upper Limit for ControlOutput	MyPIDStruct.UpperLimit
U	DeadBand	LREAL	Dead band limit	MyPIDStruct.DeadBand



Data Type: RTCStruct

Used with the [RealTimeClock](#), [DateCompare](#), and the Y_SetRTC function blocks.

Data Type Declaration

TYPE

RTC_Struct: STRUCT

Year:INT;

Month:INT;

Day:INT;

Hour:INT;

Minute:INT;

Second:INT;

mSec:INT;

END_STRUCT;

END_TYPE



Data Type: TempControlStruct

Used with the [TemperatureControl](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyAutoTuneStruct	TempControlStruct		
C	Version	UDINT	A unique value used to determine the structure format to allow for upgradability.	MyAutoTuneStruct.Version
U	AutotuneCycles	INT	The number of ActualTemperature oscillations about the TargetTemperature over which to calculate gains.	MyAutoTuneStruct.AutotuneCycles
U	IgnoreCycles	INT	The number of oscillations to ignore before calculating gains.	MyAutoTuneStruct.IgnoreCycles
U	AutotuneTimeout	DINT	Maximum amount of time in minutes to spend auto tuning (0 = infinite).	MyAutoTuneStruct.AutotuneTimeout
C	TuningComplete	BOOL	Denotes whether tuning has been completed or not.	MyAutoTuneStruct.TuningComplete
U	ResponseType	INT	Response type: Slow (1), Medium (2), Fast (3).	MyAutoTuneStruct.ResponseType
U	OutputBias	LREAL	Optional control output offset to be calculated based on setpoint.	MyAutoTuneStruct.OutputBias
U/C	PIDParameters	PIDStruct	Structure containing all parameters necessary for PID control.	MyAutoTuneStruct.PIDParameters



Data Type: XYArray

Supporting structure for [XYDataStruct](#). For use with the [XYLookup](#) function block.

Data Type Declaration

TYPE

XYArray: ARRAY[0..4000] OF [XYData](#);

END_TYPE



Data Type: XYData

Supporting structure for XYArray. For use with the [XYLookup](#) function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyXYData	XYData		
U	X	LREAL	Any data that will be used with the XY lookup function as input.	MyXYData.X
U	Y	LREAL	Any data that will be used with the XY lookup function as output.	MyXYData.Y



Data Type: XYDataStruct

For use with the [XYLookup](#) function block

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyXYDataStruct	XYDataStruct		
U	Pair	XYArray	Adjust the XYArray size if more data elements are desired.	MyXYDataStruct.Pair
U	LastPair	INT	Set this value to indicate the last array element that contains user data - be sure to check array bounds.	MyXYDataStruct.LastPair



ExplicitReceiveDataStruct

For use with the [Explicit_Message](#) function block.

Refer to 2-5.7.2 in Vol 2, Chapter 2 EtherNet/IP Adaptation of CIP.

Data Type Declaration

*	Element	Data Type	Description	Usage
---	---------	-----------	-------------	-------

	MyExplicitReceiveDataStruct	ExplicitReceiveDataStruct		
U	ED_Command1	BYTE		MyExplicitReceiveDataStruct.ED_Command1
U	ED_Command2	BYTE		MyExplicitReceiveDataStruct.ED_Command2
U	ED_Length1	BYTE		MyExplicitReceiveDataStruct.ED_Length1
U	ED_Length2	BYTE		MyExplicitReceiveDataStruct.ED_Length2
U	ED_SessionHandle1	BYTE		MyExplicitReceiveDataStruct.ED_SessionHandle1
U	ED_SessionHandle2	BYTE		MyExplicitReceiveDataStruct.ED_SessionHandle2
U	ED_SessionHandle3	BYTE		MyExplicitReceiveDataStruct.ED_SessionHandle3
U	ED_SessionHandle4	BYTE		MyExplicitReceiveDataStruct.ED_SessionHandle4
U	ED_Status1	BYTE		MyExplicitReceiveDataStruct.ED_Status1
U	ED_Status2	BYTE		MyExplicitReceiveDataStruct.ED_Status2
U	ED_Status3	BYTE		MyExplicitReceiveDataStruct.ED_Status3
U	ED_Status4	BYTE		MyExplicitReceiveDataStruct.ED_Status4
U	ED_SenderContext	SenderContext		MyExplicitReceiveDataStruct.ED_SenderContext[0]
U	ED_Options1	BYTE		MyExplicitReceiveDataStruct.ED_Options1
U	ED_Options2	BYTE		MyExplicitReceiveDataStruct.ED_Options2
U	ED_Options3	BYTE		MyExplicitReceiveDataStruct.ED_Options3
U	ED_Options4	BYTE		MyExplicitReceiveDataStruct.ED_Options4
U	ED_InterfaceHandle1	BYTE		MyExplicitReceiveDataStruct.ED_InterfaceHandle1
U	ED_InterfaceHandle2	BYTE		MyExplicitReceiveDataStruct.ED_InterfaceHandle2
U	ED_InterfaceHandle3	BYTE		MyExplicitReceiveDataStruct.ED_InterfaceHandle3
U	ED_InterfaceHandle4	BYTE		MyExplicitReceiveDataStruct.ED_InterfaceHandle4
U	ED_TimeOut1	BYTE		MyExplicitReceiveDataStruct.ED_TimeOut1
U	ED_TimeOut2	BYTE		MyExplicitReceiveDataStruct.ED_TimeOut2
U	ED_ItemCount1	BYTE		MyExplicitReceiveDataStruct.ED_ItemCount1
U	ED_ItemCount2	BYTE		MyExplicitReceiveDataStruct.ED_ItemCount2
U	ED_AddressItemID1	BYTE		MyExplicitReceiveDataStruct.ED_AddressItemID1
U	ED_AddressItemID2	BYTE		MyExplicitReceiveDataStruct.ED_AddressItemID2
U	ED_AddressItemLength1	BYTE		MyExplicitReceiveDataStruct.ED_AddressItemLength1
U	ED_AddressItemLength2	BYTE		MyExplicitReceiveDataStruct.ED_AddressItemLength2
U	ED_DataItemID1	BYTE		MyExplicitReceiveDataStruct.ED_DataItemID1
U	ED_DataItemID2	BYTE		MyExplicitReceiveDataStruct.ED_DataItemID2
U	ED_DataItemLength1	BYTE		MyExplicitReceiveDataStruct.ED_DataItemLength1
U	ED_DataItemLength2	BYTE		MyExplicitReceiveDataStruct.ED_DataItemLength2
U	ED_Response1	BYTE		MyExplicitReceiveDataStruct.ED_Response1
U	ED_Response2	BYTE		MyExplicitReceiveDataStruct.ED_Response2
U	ED_ResponseStatus1	BYTE		MyExplicitReceiveDataStruct.ED_ResponseStatus1
U	ED_ResponseStatus2	BYTE		MyExplicitReceiveDataStruct.ED_ResponseStatus2
U	ED_Data	ExplicitData		MyExplicitReceiveDataStruct.ED_Data[0]



ExplicitSendDataStruct

For use with the [Explicit_Message](#) function block.

Refer to 2-5.7.2 in Vol 2, Chapter 2 EtherNet/IP Adaptation of CIP.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyExplicitSendDataStruct	ExplicitSendDataStruct		
U	ED_Command1	BYTE		MyExplicitSendDataStruct.ED_Command1
U	ED_Command2	BYTE		MyExplicitSendDataStruct.ED_Command2
U	ED_Length1	BYTE		MyExplicitSendDataStruct.ED_Length1
U	ED_Length2	BYTE		MyExplicitSendDataStruct.ED_Length2
U	ED_SessionHandle1	BYTE		MyExplicitSendDataStruct.ED_SessionHandle1
U	ED_SessionHandle2	BYTE		MyExplicitSendDataStruct.ED_SessionHandle2
U	ED_SessionHandle3	BYTE		MyExplicitSendDataStruct.ED_SessionHandle3
U	ED_SessionHandle4	BYTE		MyExplicitSendDataStruct.ED_SessionHandle4
U	ED_Status1	BYTE		MyExplicitSendDataStruct.ED_Status1
U	ED_Status2	BYTE		MyExplicitSendDataStruct.ED_Status2
U	ED_Status3	BYTE		MyExplicitSendDataStruct.ED_Status3
U	ED_Status4	BYTE		MyExplicitSendDataStruct.ED_Status4
U	ED_SenderContext	BYTE		MyExplicitSendDataStruct.ED_SenderContext[0]
U	ED_Options1	BYTE		MyExplicitSendDataStruct.ED_Options1
U	ED_Options2	BYTE		MyExplicitSendDataStruct.ED_Options2
U	ED_Options3	BYTE		MyExplicitSendDataStruct.ED_Options3
U	ED_Options4	BYTE		MyExplicitSendDataStruct.ED_Options4
U	ED_InterfaceHandle1	BYTE		MyExplicitSendDataStruct.ED_InterfaceHandle1
U	ED_InterfaceHandle2	BYTE		MyExplicitSendDataStruct.ED_InterfaceHandle2
U	ED_InterfaceHandle3	BYTE		MyExplicitSendDataStruct.ED_InterfaceHandle3
U	ED_InterfaceHandle4	BYTE		MyExplicitSendDataStruct.ED_InterfaceHandle4
U	ED_TimeOut1	BYTE		MyExplicitSendDataStruct.ED_TimeOut1
U	ED_TimeOut2	BYTE		MyExplicitSendDataStruct.ED_TimeOut2
U	ED_ItemCount1	BYTE		MyExplicitSendDataStruct.ED_ItemCount1
U	ED_ItemCount2	BYTE		MyExplicitSendDataStruct.ED_ItemCount2
U	ED_AddressItemID1	BYTE		MyExplicitSendDataStruct.ED_AddressItemID1
U	ED_AddressItemID2	BYTE		MyExplicitSendDataStruct.ED_AddressItemID2
U	ED_AddressItemLength1	BYTE		MyExplicitSendDataStruct.ED_AddressItemLength1
U	ED_AddressItemLength2	BYTE		MyExplicitSendDataStruct.ED_AddressItemLength2
U	ED_DataItemID1	BYTE		MyExplicitSendDataStruct.ED_DataItemID1
U	ED_DataItemID2	BYTE		MyExplicitSendDataStruct.ED_DataItemID2
U	ED_DataItemLength1	BYTE		MyExplicitSendDataStruct.ED_DataItemLength1
U	ED_DataItemLength2	BYTE		MyExplicitSendDataStruct.ED_DataItemLength2
U	ED_DataService	Service		MyExplicitSendDataStruct.ED_DataService[0]
U	ED_Data	ExplicitData		MyExplicitSendDataStruct.ED_Data[0]



RegSessionRequestStruct

For use with the [Explicit_Message](#) function block.

Refer to 2-5.4.2 in Vol 2, Chapter 2 EtherNet/IP Adaptation of CIP

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyRegSessionRequestStruct	RegSessionRequestStruct		
U	RSR_Command1	BYTE		MyRegSessionRequestStruct.RSR_Command1
U	RSR_Command2	BYTE		MyRegSessionRequestStruct.RSR_Command2
U	RSR_Length1	BYTE		MyRegSessionRequestStruct.RSR_Length1
U	RSR_Length2	BYTE		MyRegSessionRequestStruct.RSR_Length2
U	RSR_SessionHandle1	BYTE		MyRegSessionRequestStruct.RSR_SessionHandle1
U	RSR_SessionHandle2	BYTE		MyRegSessionRequestStruct.RSR_SessionHandle2
U	RSR_SessionHandle3	BYTE		MyRegSessionRequestStruct.RSR_SessionHandle3
U	RSR_SessionHandle4	BYTE		MyRegSessionRequestStruct.RSR_SessionHandle4
U	RSR_Status1	BYTE		MyRegSessionRequestStruct.RSR_Status1
U	RSR_Status2	BYTE		MyRegSessionRequestStruct.RSR_Status2
U	RSR_Status3	BYTE		MyRegSessionRequestStruct.RSR_Status3
U	RSR_Status4	BYTE		MyRegSessionRequestStruct.RSR_Status4
U	RSR_SenderContext	SenderContext		MyRegSessionRequestStruct.RSR_SenderContext[0]
U	RSR_Options1	BYTE		MyRegSessionRequestStruct.RSR_Options1
U	RSR_Options2	BYTE		MyRegSessionRequestStruct.RSR_Options2
U	RSR_Options3	BYTE		MyRegSessionRequestStruct.RSR_Options3
U	RSR_Options4	BYTE		MyRegSessionRequestStruct.RSR_Options4
U	RSR_ProtocolVersion1	BYTE		MyRegSessionRequestStruct.RSR_ProtocolVersion1
U	RSR_ProtocolVersion2	BYTE		MyRegSessionRequestStruct.RSR_ProtocolVersion2
U	RSR_OptionFlags1	BYTE		MyRegSessionRequestStruct.RSR_OptionFlags1
U	RSR_OptionFlags2	BYTE		MyRegSessionRequestStruct.RSR_OptionFlags2



SenderContext

For use with the [Explicit_Message](#) function block.

Data Type Declaration

TYPE

SenderContext : ARRAY[0..7] OF BYTE;

END_TYPE



Service

For use with the [Explicit_Message](#) function block.

Data Type Declaration

TYPE

Service : ARRAY[0..7] OF BYTE;

END_TYPE



UnRegSessionRequestStruct

For use with the [Explicit_Message](#) function block.

Refer to 2-5.4.3 in Vol 2, Chapter 2 EtherNet/IP Adaptation of CIP.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyUnRegSessionRequestStruct	UnRegSessionRequestStruct		
U	USR_Command1	BYTE		MyUnRegSessionRequestStruct.USR_Commands1
U	USR_Command2	BYTE		MyUnRegSessionRequestStruct.USR_Commands2
U	USR_Length1	BYTE		MyUnRegSessionRequestStruct.USR_Length1
U	USR_Length2	BYTE		MyUnRegSessionRequestStruct.USR_Length2
U	USR_SessionHandle1	BYTE		MyUnRegSessionRequestStruct.USR_SessionHandle1
U	USR_SessionHandle2	BYTE		MyUnRegSessionRequestStruct.USR_SessionHandle2
U	USR_SessionHandle3	BYTE		MyUnRegSessionRequestStruct.USR_SessionHandle3
U	USR_SessionHandle4	BYTE		MyUnRegSessionRequestStruct.USR_SessionHandle4
U	USR_Status1	BYTE		MyUnRegSessionRequestStruct.USR_Status1
U	USR_Status2	BYTE		MyUnRegSessionRequestStruct.USR_Status2
U	USR_Status3	BYTE		MyUnRegSessionRequestStruct.USR_Status3
U	USR_Status4	BYTE		MyUnRegSessionRequestStruct.USR_Status4
U	USR_SenderContext	SenderContext		MyUnRegSessionRequestStruct.USR_SenderContext[0]
U	USR_Options1	BYTE		MyUnRegSessionRequestStruct.USR_Options1
U	USR_Options2	BYTE		MyUnRegSessionRequestStruct.USR_Options2
U	USR_Options3	BYTE		MyUnRegSessionRequestStruct.USR_Options3
U	USR_Options4	BYTE		MyUnRegSessionRequestStruct.USR_Options4



Axis Parameter List

The following tables contain controller side axis parameters which can be read or written using the function blocks [MC_ReadParameter](#), [MC_ReadBoolParameter](#), [MC_WriteParameter](#), [MC_WriteBoolParameter](#), and [Y_ReadStringParameter](#). This is a comprehensive list that contains parameters that may not be applicable for all axis types. Some parameters displayed in the Hardware Configuration cannot be read or written by the application program using this function block. Only the parameters listed here are accessible via the Application program.

Name	Parameter	Data Type	R/W	Default	Comments
AbsoluteEncoderOffset	1838	LREAL	R/W	LREAL#0.0	Offset stored when executing MC_SetPosition for a servo axis configured to use an absolute encoder. (ServoPack Pn002.2.) This value can be read and stored externally, and later written into the controller if the MPiec battery fails, or if the MPiec controller is replaced.
ActualPosition	1000	LREAL	R	N/A	Feedback position in user units.
ActualPositionCyclic	1005	LREAL	R	N/A	Requires firmware version 1.0.6 or greater.
ActualPositionNonCyclic	1006	LREAL	R	N/A	Requires firmware version 1.0.6 or greater.
ActualTorque	1004	LREAL	R	N/A	Feedback torque in percentage of rated torque.
ActualVelocity	1001	LREAL	R	N/A	Feedback velocity in user units per second.
AmplifierModel	1819	STRING	R	N/A	Amplifier model number.
AxisType	1810	LREAL	R	N/A	Returns a code corresponding to the type of axis configured. 0=Servo, 1=VFD, 2=External Encoder, 3=Virtual, 4=Stepper.
BufferedMotionBlocks	1600	LREAL	R	N/A	The number of motion blocks buffered in the motion queue. This value will increase when a motion block is executed with any of the non aborting types and decrement as each buffered block has control of the motion.

Name	Parameter	Data Type	R/W	Default	Comments
CamMasterCycle	1512	LREAL	R	1.0	If the axis is currently linked to a master axis for camming, this parameter indicates the cam master cycle as determined by the Cam Table currently in use. This parameter is not valid until Y_CamIn.Execute occurs. If Y_CamIn is executed while camming is already in operation, this parameter will not update until Y_CamIn.InSync is TRUE.
CamMasterPosition	1500	LREAL	R	N/A	Cam master position plus scan compensation added. See the Camming Block Diagram .
CamMasterScale	1510	LREAL	R	N/A	See the Camming Block Diagram .
CamMasterShift	1511	LREAL	R	N/A	This value holds the cumulative value of all previous Y_CamShifts, the absolute cam shift. To reset this parameter to zero, execute Y_CamShift with Y_CamShift.PhaseShift equal and opposite of the value of this parameter. See the Camming Block Diagram .
CamMasterShiftedCyclic	1502	LREAL	R	N/A	This is the master position that is fed into the cam look-up function. It includes all adjustments made from the initial master position, such as scan compensation, it is modularized for the master data in the cam table, and any shift applied via Y_CamShift. See the Camming Block Diagram .
CamMasterShiftedPosition	1501	LREAL	R	N/A	See the Camming Block Diagram .
CamOffset	1531	LREAL	R	N/A	See the Camming Block Diagram .
CamOffsetRemaining	1533	LREAL	R	N/A	If a Y_SlaveOffset is in progress, this is the remaining amount of offset to be added to the absolute CamOffset (Parameter 1531), otherwise this value is zero.
CamScale	1530	LREAL	R	100.0	This a multiplication factor applied to the slave data. See the Camming Block Diagram .

Name	Parameter	Data Type	R/W	Default	Comments
CamScaleRemaining	1534	LREAL	R	N/A	If a Y_CamScale is in progress, this is the remaining amount of Scaling to be added to the absolute CamScale (Parameter 1530), otherwise this value is zero.
CamShiftRemaining	1513	LREAL	R	N/A	If a Y_CamShift is in progress, this is the remaining amount of phase shift to be added to the CamMasterShift (Parameter 1511), otherwise this value is zero.
CamState	1540	LREAL	R	N/A	See CamState in the Camming Overview section of this manual. 0 = Not Engaged, 1 = Waiting to Engage, 2 = Engaging, 3 = Engaged, 4 = Waiting to Disengage, 5 = Disengaging.
CamTableCumulativeOutput	1521	LREAL	R	N/A	Initialized to 0 when the cam first engages and represents the total commanded slave distance traveled.
CamTableIDEngaged	1541	LREAL	R	0	Indicates the cam table currently in use by the motion engine. This number becomes valid when the CamState changes from 0 to 1. If a cam is already engaged (CamState = 3), this number becomes valid when the new table becomes engaged.
CamTableOutput	1520	LREAL	R	N/A	This is the table value selected as the synchronized position based on the master position. See the Camming Block Diagram .
CommandedAcceleration	1012	LREAL	R	N/A	Commanded acceleration in user units /second ² .
CommandedAccelerationFiltered	1022	LREAL	R	N/A	Commanded acceleration in user units /second ² (Post S-curve filter.)
CommandedPosition	1010	LREAL	R	N/A	Commanded position in user units.
CommandedPositionCyclic	1015	LREAL	R	N/A	If axis is set to rotary type, this value reports the position from 0 to MachineCycle.

Name	Parameter	Data Type	R/W	Default	Comments
CommandedPositionNonCyclic	1016	LREAL	R	N/A	Reports the unmodularized commanded position regardless of whether the axis is configured as rotary or linear.
CommandedPositionNonCyclicFiltered	1020	LREAL	R	N/A	Commanded position sent to the ServoPack (Post S-curve filter.) Refer to the Command Filtering (MP2300Siec/MP2310iec) and Command Filtering (MP2600iec) block diagrams for details
CommandedPositionSubFilter	1311	LREAL	R	N/A	Configures the servo amplifier to interpolate intermediate points in the motion profile between MECHATROLINK updates from the controller. This provides for a smoother motion profile. Settings are as follows: 0 = No interpolation; 1 = Exponential interpolation; 2 = Moving average filter.
CommandedTorque	1014	LREAL	R	N/A	Commanded torque in percentage of rated torque. Valid only when commanding a torque using MC_TorqueControl .
CommandedTorqueFiltered	1024	LREAL	R	N/A	Commanded torque in percentage of rated torque. Valid only when commanding a torque using MC_TorqueControl .
CommandedVelocity	1011	LREAL	R	N/A	Commanded velocity in user units / second.
CommandedVelocityFiltered	1021	LREAL	R	N/A	Commanded velocity in user units / second. (Post S-curve filter.)
ControllerFeedForwardEnable	1310	BOOL	R/W	TRUE	Alternative of servo amplifier's Pn109 parameter. User can enable this gain in either the controller of servo amplifier. Both settings are not recommended simultaneously.
ExternalRawPositionCyclic	1007	LREAL	R	N/A	Applicable for External Encoder set in rotary mode only. Refer to the external encoder block diagram for details (Requires FW version 1.2.3 or higher.)

Name	Parameter	Data Type	R/W	Default	Comments
ExternalRawPositionNonCyclic	1008	LREAL	R	N/A	Applicable for External Encoder only. Refer to the external encoder block diagram for details. (Requires FW version 1.2.3 or higher)
ExternalVelocityUnfiltered	1009	LREAL	R	N/A	Instantaneous external encoder velocity. Refer to the external encoder block diagram for details. (Requires FW version 1.2.3 or higher.)
FeedConstant	1101	LREAL	R/W		Feed constant at final output shaft of the machine
FilterMovingAverage	1301	LREAL	R/W		This value is the S-Curve time constant. The units are seconds, and the range is 0.0 to 5.0 (zero exclusive). This parameter can only be written if the axis is in the "Stand-still" state. Use MC_ReadStatus to verify.
FilterMovingAverageEnable	1300	BOOL	R/W		Apply S-Curve filter. This parameter must first be enabled in the Hardware configuration to be settable via the MC_WriteParameter function block via the application program.
HighSpeedOutputEnable	1050	BOOL	R/W	FALSE	Set TRUE to arm or toggle to re-arm the external encoder high speed output.
HighSpeedOutputPosition	1052	LREAL	R/W	0.0	Set this value before the high speed output function is enabled.
HighSpeedOutputPositionNonCyclic	1053	LREAL	R/W	0.0	This is the user unit equivalent of the raw 32 bit encoder value set in the LIO hardware for high speed output compare.
HighSpeedOutputStatus	1051	BOOL	R	N/A	Status bit indicates when the hardware sets the high speed output and remains set until the function is disabled.
InputRatio	1110	DINT	R/W		GearBox Input ratio.
LatchPositionCyclic	1030	LREAL	R	N/A	Reports the modularized latch position. This value is only valid if the axis is configured as rotary.
LatchPositionNonCyclic	1031	LREAL	R/W	N/A	Reports the unmodularized latch position regardless of whether the axis is configured as rotary or linear.

Name	Parameter	Data Type	R/W	Default	Comments
LatchPositionNonCyclic	1033	LREAL	R/W	N/A	When using Mechatrolink III ServoPacks, and two MC_TouchProbe function blocks simultaneously, this parameter reports the second (TRIGGER_REF.ID=1) unmodularized latch position regardless of whether the axis is configured as rotary or linear.
LimitAccelEnable	1222	BOOL	R/W	TRUE	Enable the acceleration limit function.
LimitAccelNegative	1220	LREAL	R/W	-1.797693E+308	Negative acceleration limit.
LimitAccelPositive	1221	LREAL	R/W	1.797693E+308	Positive acceleration limit
LimitDecelEnable	1232	BOOL	R/W	TRUE	Enable the deceleration limit function.
LimitDecelNegative	1230	LREAL	R/W	-1.797693E+308	Negative deceleration limit
LimitDecelPositive	1231	LREAL	R/W	1.797693E+308	Positive deceleration limit.
LimitPositionEnable	1202	BOOL	R/W	TRUE	Enable the position limit function.
LimitPositionNegative	1200	LREAL	R/W	-1.797693E+308	Negative position limit in user units.
LimitPositionPositive	1201	LREAL	R/W	1.797693E+308	Positive position limit in user units.
LimitTorqueDefault	1400	LREAL	R/W	100.0	Default torque limit for blocks with a torque limit input.
LimitVelocityEnable	1212	BOOL	R/W	TRUE	Enable the velocity limit function.
LimitVelocityNegative	1210	LREAL	R/W	-1.797693E+308	Negative velocity limit in user units / second.
LimitVelocityPositive	1211	LREAL	R/W	1.797693E+308	Positive velocity limit in user units / second.
LoadType	1807	BOOL	R	N/A	0=Linear, 1=Rotary; as set in the Hardware Configuration.
MachineCycle	1833	LREAL	R	N/A	If the LoadType is set for Rotary operation, this is the rollover position. If the load type is set for Linear, this value has no meaning. Firmware Version 1.2.2 is required.

Name	Parameter	Data Type	R/W	Default	Comments
MechatrolinkCompensation	1307	BOOL	R	TRUE	Only applicable for camming and gearing modes and for MECHATROLINK axes. This value determines if scan compensation is calculated to account for the network delay when sending commanded positions to the amplifier. Its purpose is to eliminate master / slave phase lag due to the time required to send the position data to the amplifier. Firmware Version 1.2.3 is required. Refer to the Camming Block Diagram .
MotorModel	1823	STRING	R	N/A	Motor model number. Requires FW 2.0. Use Y_ReadStringParameter in YMotion firmware library.
NodeStatus	1330	BOOL	R	N/A	Indicates if the node (drive) is powered up and the MECHATROLINK cable is properly connected. TRUE = Node is communicating.
OptionMonitor	1312	LREAL	R	N/A	Returns the value of the servo amplifier's Un Monitor as selected by Pn825 according to the MECHATROLINK-II Communication Manual SIEPS8000054 , section 5.2.7 (3). For example, to read the RMS torque output, set Pn825 to UINT#16#19. On MP2300iec controllers, firmware version 1.2.2 is required. On the MP2600iec, firmware version 2.1.0 is required.
OutputRatio	1111	DINT	R/W		GearBox Output ratio.
PositionCompensationEnable	1308	BOOL	R/W	FALSE	Enables/disables position compensation mode. See Using Position Compensation Tables
PositionError	1130	LREAL	R	N/A	Position Error, following error, or difference between commanded and actual position in user units.
PositionScalePerRev	1829	LREAL	R	N/A	User units per motor revolution.
PositionScalePerTick	1830	LREAL	R	N/A	User units per encoder count.

Name	Parameter	Data Type	R/W	Default	Comments
PositionWindow	1120	LREAL	R/W		Window for positioning commands. Actual feedback must be within this window for a move to be considered complete.
ScanCompensation	1305	LREAL	W	2 scans	For external encoders only. This value provides scan compensation to ensure the master and slave remain synchronized even at high speeds. Units are in seconds. The default of 2 scans was predetermined by Yaskawa and should not need adjustment in most cases. The maximum compensation is 10 MECHATROLINK scans. For example, if the MECHATROLINK update is 2 ms, then parameter 1305 can range from 0 to 0.020000 seconds). Firmware Version 1.2.2 is required. Refer to the Camming Block Diagram .
VelocityFilter	1306	LREAL	R/W	0.0	Provides a moving average filter for the feedback velocity over a specified time period. Units are in seconds. Note that the time value will be rounded to the nearest number of MECHATROLINK or DPR scans. For example, if the MECHATROLINK is set for 2.0 ms, and the VelocityFilter is set to 0.010, then the velocity will be averaged over 5 samples. The maximum filter time is 0.100 seconds.



Function Block ErrorID List

ErrorID	Description
0	No error.
1	Time limit exceeded.
2	Distance limit exceeded.
3	Torque limit exceeded.
4	The file is already opened.
5	The file is write protected or access is denied. Tip #1: Make sure a leading / has not been omitted. Example STRING#'/flash/user/data/myfile.txt' Tip #2: If the FileName is a STRING variable, do not include single quotes in the string. Only include single quotes if when specifying the FileName as a literal such as STRING#'MyFile.txt'
6	File name not defined. Tip: If the FileName is a STRING variable, do not include single quotes in the string. Only include single quotes if when specifying the FileName as a literal such as STRING#'MyFile.txt'
17	Soft Limit Violation for group (position max).
Motion State Error	
4369	The move could not be buffered because the motion queue is full. The default queue size for single axis moves is 16. For Mechatrolink groups, the queue size can be set in the Hardware Configuration, Yaskawa recommends setting the IO Task Assignment to the cyclic task with the fastest update interval. For MLX hosted robots, the queue size is 25.
4370	The move could not be started because motion is prohibited. Possible causes: 1) The axis may not be enabled. Check MC_Power.Status output. 2) An MC_Stop.Execute might be held high - no other block can override the Stopping state. 3) Verify that the limit switches are not active - check the Global Variables for the Axis. 4) A motion block may be attempting to abort an MC_TorqueControl move. 5) If this axis is included in a set of Grouped axes, this Error may occur when executing a single axis function block while the axis is part of an Enabled Group. Use MC_GroupDisable first. 6) If the axis is a cam slave AND configured as a rotary axis, verify that the slave positions in the Y_MS_CAM_STRUCT are within 0.0 to MachineCycle.
4371	The servo drive failed to enable or disable. Check the amplifier wiring for L1 / L2 / L3. The amplifier could be e-stopped or has an alarm.
4375	CamOut called while not camming.
4376	The master slave relationship cannot be modified because the master axis has not been set yet.
4377	File reading already in progress.
4378	The function block is not applicable for the external axis specified.
4379	A homing sequence is already in progress.
4380	MC_SetPosition cannot be executed while the axis is already moving.
4381	Motion was aborted due to an axis alarm.
4382	When an axis is configured for rotary mode, and the MC_SetPosition tries to set a position that is equal to or greater than the MachineCycle, this error is generated, and the position is not set.
4383	Axis must be commanded at standstill when homing is attempted. Refer to the Motion State Diagram and MC_ReadStatus. Maybe the axis is not enabled using MC_Power?
4384	Clear alarms is already in progress.

ErrorID	Description
4385	Axis reset is already in progress.
4386	MECHATROLINK reset is already in progress.
4387	Already copying cam data (This error occurs if Execute transitions to TRUE while Busy is already TRUE).
4388	CamTableRead cannot read a second cam structure while the first cam structure is being read.
4389	CamTableWrite cannot write a second cam structure while the first cam structure is being written.
4390	Position cannot be defined while the axis is in a master / slave relationship. To redefine the position, use the MC_Stop function block for slave axis, then execute MC_SetPosition. If attempting the redefine a master position, execute MC_Stop for all slaves first.
4391	The function block cannot be used with a virtual axis.
4392	The function block cannot be used with an inverter axis.
4393	Y_VerifyParameters and Y_WriteParameters cannot be called a second time while the first instance is still in progress.
4394	More than 10 Y_CamIn, Y_CamOut, or MC_GearInPos function blocks for a given axis are active at the same time. Most likely the application program is not coded correctly, and the Execute input is being fired too frequently.
4395	Window parameters are outside of the master axis' machine cycle. (0 to Prm 1502, the last master position in the active cam table.)
4396	Axis latch function already in use.
57005	Function block or function block library is not implemented in this product (i.e. FMK component not installed)
4397	Over travel is limit still ON after attempting to move away from it.
4398	The cam shift is not possible with EndPosition and current master position. This error occurs if the shift is greater than the distance to the end of the window. For example: shift = 90, window [180,360], and the master position = 300 when Y_CamShift.Execute=TRUE. (There is only 60 degrees of distance remaining.) To remedy this situation, execute Y_CamShift sooner. The function itself will monitor for the StartPosition and wait if necessary to actually start the correction. It is not necessary to monitor for the right position in the IEC application task before executing this function. This error has also been experienced when using very low resolution external encoders, or during testing when using a hand wheel to manually move the master. Both of these situations can lead to noisy encoder movement, and if the master moves backwards during a CamShift, it may cause this error. (A low resolution encoder is defined as a system with a Master / Slave equivalent movement pulse ratio greater than 50 or 100 or higher.)
4399	The L1 / L2 / L3 power inputs on the drive may not be supplied with power, possibly due to an E-Stop condition.
4400	The safety input (HBB on the CN8 connector) is preventing the drive from enabling.
4401	The controller cannot communicate with the drive. It may be disconnected from the MECHATROLINK network.
4402	The scan compensation delay parameter 1305 is only valid for external encoders.
4403	The High Speed Output functionality is only available on external encoders.
4404	Cannot execute MC_GearOut because the axis is not in gear.
4405	Y_CamOut was aborted.
4406	Continuous Latch Mode is not supported on Sigma II, Sigma III, or external encoders.
4407	Internal buffer overflow.
4408	PatternSize is out of range (1-8) or PatternCount is out of range (0-255).
4409	Parameter write is already in progress.
4410	Parameter is read-only.
4411	The function block cannot be re-executed while it already is in progress.
4412	Parameter not supported for the specified axis or group.
4413	The stepper axis does not support the mode of motion commanded.
4414	MECHATROLINK communications to the drive was disrupted. Execute MC_Reset to restore the connection.

ErrorID	Description
4415	Reboot is already in progress.
4416	Add IP Address already in progress
4417	Remove IP Address already in progress
4418	Debug Print already in progress
4419	Motion queue resize failed. Motion queue is not empty.
4420	Brake release function failed to execute. Brake release is prohibited while servo on, or the axis may not support brake release
4421	Servo ON is prohibited while the brake is manually released with Y_BrakeRelease
4422	Position offset update failed. This can happen if MC_SetPosition is called too often with absolute encoder axis on Sigma-7Siec (can't write the offset to flash fast enough).
4423	The function block cannot be used with an active slave axis.
4424	Velocity override value is outside the allowable range of 0.0 to 1.0 inclusive.
4425	Slave axis must be configured as a rotary axis type with a valid Machine Cycle.
4426	The function block is only supported for VFDs.
Invalid Structure Value	
4624	RESERVED - General structure value error.
4625	AXIS_REF.AxisNum does not correspond to an axis configured on the system. Verify the value of AxisNum matches a logical axis number in the configuration. Tip: Make sure a variable of type AXIS_REF is properly declared as a VAR or VAR_GLOBAL in all relevant POU's.
4626	The master / slave relationship is already defined. If a slave must follow a different master, use the MC_Stop block on the slave before executing the next Y_CamIn. If cascading master slaves, a maximum of two levels of cascaded master / slave relationships can be configured.
4630	Trigger reference is not valid.
4633	Table size results in misaligned data. Refer to the help section 'Internally Created Cam Data.' A cam table will have a multiple of 16 bytes if created correctly.
4634	Buffer size results in misaligned data.
4635	Table type is not supported.
4636	Invalid start index.
4637	Invalid end index.
4638	Buffer Overrun. User Buffer is full.
Invalid Enumeration Type	
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4643	Start mode does not correspond to a valid enumeration value.
4644	Invalid shift mode.
4645	Offset mode does not correspond to a valid enumeration value.
4646	Mode does not correspond to a valid enumeration value or the enumeration is not supported.
4647	The synch mode does not correspond to a valid enumeration value.
4648	The parameter number does not exist for the specified axis - OR The parameter number requires the other function block (e.g. MC_WriteParameter vs. MC_WriteBoolParameter)
4649	Invalid adjust mode.
4650	RampIn does not correspond to a valid enumeration value.
4651	ControlMode does not correspond to a valid enumeration value.
4652	EndMode does not correspond to a valid enumeration value. Y_CamOut only supports 'AtPosition' mode.
4653	ExecutionMode does not correspond to a valid enumeration value or the selected behavior is unsupported.
4654	Invalid Speed Unit setting in Y_MoveOptions.ProfileUnit. Select 0 for Absolute units, or 1 for % of maximum.
Range Error	
4657	Distance parameter is less than or equal to zero.
4658	Velocity parameter is less than or equal to zero.

ErrorID	Description
4659	Acceleration is less than or equal to zero. For remote hosted robots, acceleration cannot be less than 20.0.
4660	Deceleration is less than or equal to zero. For remote hosted robots, deceleration cannot be less than 20.0.
4661	Torque is less than or equal to zero.
4662	Time is less than or equal to zero.
4663	Specified time was less than zero.
4664	Specified scale was less than or equal to zero.
4665	Velocity parameter is negative. (Conveyor moving in wrong direction.)
4666	Denominator is zero.
4667	Jerk is less than or equal to zero. For remote hosted groups such as MLX200, the Jerk units are in percentage, and the range is 20 to 100%.
4668	Torque Ramp is less than or equal to zero.
4669	Engage position is outside the cam table domain.
4670	Engage window is less than zero.
4671	Disengage position is outside the cam table domain.
4672	Negative Disengage Window.
4673	StartPosition is outside of master's range.
4674	EndPosition is outside of master's range.
4675	Axis filter time constant out of range, or an attempt to change the value was made while the axis was enabled. (The axis must be disabled to change the moving average time constant.)
4676	The time value must be within 0 to 10 MECHATROLINK cycles.
4677	Array size too large.
4678	Buffer array index out of range.
4679	Invalid date or time values entered.
4680	Invalid acceleration filter type entered.
4681	Position value exceeded configured limits.
4682	Velocity value exceeded configured limits.
4683	Acceleration value exceeded configured limits.
4684	IdentInGroup not found. Verify that the string name of the joint exactly matches the definition in the Hardware Configuration. OR - Invalid JointIndex. The value provided does not map to a valid AXIS_REF on the system.
Invalid Input Data	
4847	MotomanSync controller did not receive a watchdog value from MPiec in time. There may be network communication issues causing delays. Check S3C1380 (communication timeout, 50ms default) on MotomanSync controller.
4875	MotomanSync controller did not receive a watchdog value from MPiec in time. There may be network communication issues causing delays. Check S3C1380 (communication timeout, 50ms default) on MotomanSync controller.
4880	RESERVED
4881	The specified Pn does not exist.
4882	The mask does not correspond to valid tracks.
4883	The profile must start with relative time equal to zero, and the time must be increasing.
4884	The specified cam file does not exist.
4885	Invalid header for the cam file (missing # of rows, #of columns, or feed-forward velocity flag). You must first populate the TableType and DataSize in the Y_MS_CAM_STRUCT before executing the function.
4886	The first (master) column must be either increasing or decreasing. If the master data is incremental, even the very first point cannot be zero.
4887	CamTableID does not refer to a valid cam table.
4888	The engage phase exceeded the time limit. Slave axis could not attain the target position and velocity within the user specified time limit.

ErrorID	Description
4889	The engage phase exceeded the distance limit. Slave axis could not attain the target position and velocity within the user specified master distance.
4890	Invalid width input. Width is an enumeration type with the following allowable values 'WIDTH_8'=0, 'WIDTH_16'=1, and 'WIDTH_32'=2.
4891	The slave axis cannot be the same as the master axis.
4892	Default drive parameter info is not available for this parameter. Use the DataType Override input to specify the parameter size.
4893	The specified external axis may not be used. A physical axis is required.
4894	The specified virtual axis may not be used with this function block.
4895	Missing or unrecognized file extension.
4896	Could not find the axis parameter file.
4897	The drive's model number or type does not match the parameter file.
4898	The S Curve filter parameter must first be enabled in the Hardware Configuration before it is possible to enable/disable it using MC_WriteParameter.
4899	Axis position compensation file not found.
4900	Invalid axis position compensation file format.
4901	Cannot enable/disable axis position compensation while servo on.
4902	Invalid compensation table wrap range.
4903	Data Type value does not match the data type of the Value input.
4904	Data Type is out of range.
4905	Invalid IO Identifier.
4912	Invalid Mechatrolink-III Node ID
4913	Aux Cmd for Mechatrolink-III IO module not supported
4914	Invalid Mode argument for Y_MLinkIOMode
Y_DeviceComm ErrorIDs	
8705	The maximum number of concurrently open user sockets/IO device handles has been reached or exceeded.
8706	The socket/IO device handle was invalid. Invalid IP address.
8707	The IP address string was not in a valid format.
8708	The socket/IO device handle could not be created.
8709	The specified address or port is already in use on the local network.
8710	The specified address or port is not available for use. (Maybe the IP address specified is not assigned to one of the networks available on this MPiec?)
8711	Unable to accept new socket/IO device handle connection.
8712	Unable to bind to the specified address.
8713	The socket/IO device handle type argument was invalid.
8714	The local address or port was not valid.
8715	Connecting to the socket/IO device handle failed.
8716	The remote IP address is unreachable. Check the default gateway.
8717	The socket/IO device handle is already connected to another endpoint.
8718	The socket/IO device handle connection attempt was actively refused by the remote device. Possibly the remote device is not listening on the port.
8719	The socket/IO device handle was not connected to a remote endpoint. Call Y_ConnectSocket prior to Y_ReadDevice or Y_WriteDevice.
8720	An error occurred trying to get or set the device option.
8721	The communication device could not be read.
8722	The communication device could not be written.
8723	A valid buffer argument to WriteDevice and ReadDevice is required.
8724	Invalid Device Option ID.
8725	The device option value was not the right size or the data was out of range.
8726	The serial port ID was not a valid serial port.
8727	The serial port specified could not be opened.

ErrorID	Description
Group ErrorIDs. Please note that Group Errors up to 200 listed here are reported as 32 bit values on the Web UI such as "3202 0018" and also reported via MC_GroupReadError as an ErrorClass (3202) and a GroupErrorID (0018), but in some cases only the GroupErrorID is reported if the function block reporting an error only outputs the traditional Error and ErrorID.	
0001	The command position was outside the allowable range for the axis in the positive direction (positive overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
0002	The command position was outside the allowable range for the axis in the negative direction (negative overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
0003	The command speed was greater than the allowable range for the axis in the positive direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
0004	The command speed was greater than the allowable range for the axis in the negative direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
0005	The command acceleration was greater than the allowable range for the axis in the positive direction. The axis may not be moved again until the alarm condition is cleared.
0006	The command acceleration was greater than the allowable range for the axis in the negative direction. The axis may not be moved again until the alarm condition is cleared.
0007	The command torque was greater than the allowable range for the axis in the positive direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
0008	The command torque was greater than the allowable range for the axis in the negative direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
0011	The command position was outside the allowable range for the axis in the positive direction (positive overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
0012	The command position was outside the allowable range for the axis in the negative direction (negative overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
0013	The command speed was greater than the allowable range for the axis in the positive direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
0014	The command speed was greater than the allowable range for the axis in the negative direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
0015	The command acceleration was greater than the allowable range for the axis in the positive direction. The axis may not be moved again until the alarm condition is cleared.
0016	The command acceleration was greater than the allowable range for the axis in the negative direction. The axis may not be moved again until the alarm condition is cleared.
0017	The command torque was greater than the allowable range for the axis in the positive direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
0018	The command torque was greater than the allowable range for the axis in the negative direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
0021	The command position was outside the allowable range for the axis in the positive direction (positive overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.

ErrorID	Description
0022	The command position was outside the allowable range for the axis in the negative direction (negative overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
0023	The command speed was greater than the allowable range for the axis in the positive direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
0024	The command speed was greater than the allowable range for the axis in the negative direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
0025	The command acceleration was greater than the allowable range for the axis in the positive direction. The axis may not be moved again until the alarm condition is cleared.
0026	The command acceleration was greater than the allowable range for the axis in the negative direction. The axis may not be moved again until the alarm condition is cleared.
0027	The command torque was greater than the allowable range for the axis in the positive direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
0028	The command torque was greater than the allowable range for the axis in the negative direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
0031	The move specified would exceed the software position limits in the positive direction and was rejected before being started. The group may be moved again immediately if desired.
0032	The move specified would exceed the software position limits in the negative direction and was rejected before being started. The group may be moved again immediately if desired.
0033	The move specified would exceed the software speed limits in the positive direction and was rejected before being started. The group may be moved again immediately if desired.
0034	The move specified would exceed the software speed limits in the negative direction and was rejected before being started. The group may be moved again immediately if desired.
0035	The move specified would exceed the software acceleration limits in the positive direction and was rejected before being started. The group may be moved again immediately if desired.
0036	The move specified would exceed the software acceleration limits in the negative direction and was rejected before being started. The group may be moved again immediately if desired.
0037	The move specified would exceed the software torque limits in the positive direction and was rejected before being started. The group may be moved again immediately if desired.
0038	The move specified would exceed the software torque limits in the negative direction and was rejected before being started. The group may be moved again immediately if desired.
0039	The predictive soft limit encountered a segment that doesn't support the predicted stopping point.
0041	Cam and Contour tables must have a header indicating the number of rows and columns and a feed forward velocity flag. Comma separated data values following the header.
0042	In CamTables, the first (master) column must be either increasing or decreasing.
0043	In ContourTables, the first (time) column must start at zero and be increasing.
0044	The master position was outside the range of the CamTable, which automatically stopped the cam motion.
0045	One or more slave axes could not attain the target position and velocity within the user specified time limit for the Cam or Gear motion.
0046	One or more slave axes could not attain the target position and velocity within the user specified distance limit for the Cam or Gear motion.
0051	Axis enable failed. This problem is usually a result of communication problems with the servo drive.
0052	Runtime computation detected an invalid motion parameter.
0053	Runtime computation detected jerk limit too low to allow correction for off-axis motion.
0054	Runtime computation detected acceleration limit too low to allow correction for off-axis motion.
0056	Off-axis initial conditions correction was not able to complete before the main motion profile ended, leading to a discontinuity in velocity. This can happen if the motion planning computations are not adequate for short moves in cross-frame transitions.

ErrorID	Description
0100	The inverse kinematics computation detected a world position that can not be reached.
0101	The inverse kinematics computation detected that the elbow 'handedness' (orientation) does not match the configuration. The 'handedness' must be fixed by commanding the individual axes or manually moving the robot.
0102	The robot XY position intruded into the configured dead zone area near the origin.
0103	A valid command velocity is required for inverse kinematics computation of Scara robots when near the singularity. Choose a different motion type which includes valid command velocity generation.
0104	The solution for Scara inverse kinematics requires that any motion very near the singularity be continuous. Coming to a stop very near the singularity can cause a sudden position jump in subsequent motions. This region is controlled with the configuration parameter 'correctionRadius'.
0105	The axis group has been misconfigured with too few axes or too small a world coordinate space to satisfy the dimensionality of the kinematics.
0106	The commanded motion generated elbow values that lie within the configured 'outerSingularInterpRange', and the orthogonality check between the direction of motion and the singular direction was not within the configured 'singularityOrthogonalityTolerance'
0107	The kinematics algorithm encountered an unexpected error
0108	The kinematics solution produced angles outside the allowable range
0109	Inverse kinematics is not supported when at the singularity configuration.
0200	The axis group split-axis command data did not agree. The component axes must be moved to identical locations separately before issuing multi-axis commands.
8960	Invalid axes group. Confirm that the AxesGroup variable has the correct %M address as automatically assigned by the Hardware Configuration. Be sure the AxesGroup variable is global, and that if the group has been recently defined, the controller was rebooted.
8961	An axis is already owned by another group.
8962	Group activation is blocked. Ownership cannot be changed while Mechatrolink reset is in progress.
8963	Specified Coordinate System is not supported. Possible causes: If a custom group is specified, only the ACS Coordinate System is applicable.
8964	Move prohibited because group has an alarm.
8965	Group activation prohibited, invalid axis/joint config.
8966	Group activation prohibited, mismatched axis command position for split axis. Example: X and X Prime sharing the same load. The group must be disabled to clear this alarm.
8967	The group reports one or more of its axes has an error.
8968	Axis group reset is already in progress.
8969	Invalid circular path method.
8970	Invalid PathChoice. MC_MoveCircularAbsolute.PathChoice cannot be set to Clockwise or Counter Clockwise if the Group can move in three dimensions.
8971	Invalid circle geometry. Check the PathChoice input, only Longest and Shortest are allowed if the group has more than two dimensions. Be sure that the starting point (which is the position of the group before executing MC_MoveCircularAbsolute,) the AuxPoints, and the EndPoint define the intended arc. If using MC_CircleMode#Center, the calculated radius of the start position to the center and the end position to the center must be within 0.1%. It may be necessary to increase the resolution of the values provided.
8972	A grouped axis is disabled. If using MC_GroupEnable, all axes must be powered up using Y_GroupPower first.
8973	Invalid transition mode.
8974	Invalid transition parameter.
8975	Invalid transition geometry. The values for the acceleration, deceleration, and/or velocity of the transition yield an invalid geometry. Given the limits of accel/decel, velocity, and length of the segment, can't create the corner geometry to meet the specification.
8976	Invalid axes group state transition. Axes groups cannot transition directly between certain states, such as direct transition to disabled state from moving or error states. Use the appropriate function block to transition to the correct intermediate state.

ErrorID	Description
8977	Invalid axes group motion coordinate type. The optional limit coordinate type specifier (a.k.a. VelocityUnit) parameter for the motion was outside the allowed range.
8978	Infinite velocity constraint. The resolved velocity limit for the move was infinite. If there is no Cartesian motion, and a rotational change only, use the MoveOptions input to specify the VelocityUnits as 'UseRotationalScalers.' Or - There is a non zero value in the position VECTOR for a degree of freedom that the AxesGroup does not support.
8979	Infinite acceleration constraint. The resolved acceleration limit for the move was infinite.
8980	Infinite deceleration constraint. The resolved acceleration limit for the move was infinite.
8981	Insufficient Coordinate Frame size.
8982	Invalid Tangent Plane.
8983	Invalid Colinearity Angle. The range is 0.0 to 10.0.
8984	The points specified to describe the circle are invalid due to a mismatch in dimensions, or the rotations do not match.
8985	Acceleration constraints violation. Computed motion violates acceleration constraints.
8992	Computed motion violates velocity constraints.
8993	Group must not be enabled for this action.
8994	Group filter not supported.
8995	Axis group filter time constant too large.
8996	Group filter time constant must be set to non-zero value prior to enabling
8997	The specified action was not supported for the specified coordinate frame combination.
8998	A zero-length vector was supplied for the direction
8999	The specified coordinate frame was actively involved in commanded motion and could not be modified. PCS frame already in motion with ExecutionMode#immediate (use queued mode instead.)
9000	The specified blending transition is not possible based on the two moves in the buffer to be blended. For example, Move 1 is a line and Move 2 is an arc, and they are not in the same plane.
9001	The specified blending transition required exact corner distance or deviation, but insufficient distance remained in the segment to satisfy the transition geometry.
9002	The position was unreachable due to inverse kinematics limitations. Try a direct move instead of a linear move.
9003	Specified blending transition for the function block could not be realized due to blending parameter restrictions. Typically due to accel limit too low or segment length too short, relative to transition velocity.
9004	Invalid or unsupported ExecutionMode.
9005	Invalid or unsupported TrackProfile
9006	Invalid TrackProfile parameter, possibly the SyncIn and SyncOut master distance sum is greater than (EndDistance - StartDistance)
9007	The execution of the function with ExecutionMode = Queued or Delayed failed. The conditions for applying the function were validated when it was executed, but when the deferred execution occurred the conditions were no longer valid. Check the sequencing of the program.
9008	Invalid ServoPack memory address. The specified memory address is out of range.
9009	Unsupported ServoPack memory operation. Memory operations are not supported on this ServoPack model.
9010	Invalid ServoPack memory access mode. Attempted non volatile write to volatile only address.
9011	The tracking position was already past the allowable sync range before the conditions for sync-in were met.
9012	Conveyor tracking cannot be re executed while TrackState is non zero. (Tracking is currently in operation.)
9013	The target coordinate frame is invalid or not defined. Tip: Use Y_GroupSetFrameOffset or MC_TrackConveyorBelt before executing a move in PCS.
9014	The Conveyor reversed direction and re entered a previously completed TrackState region.
9015	The tracking position traveled past TrackOptions.StartDistance + DelayLimit with ExecutionMode#Queued or #Delayed without the ExecutionMode being satisfied.

ErrorID	Description
9016	Group not paused / interrupted.
9017	A synchronized group cannot enter the velocity override state.
9018	Attempted non volatile write while axis enabled.
9022	The alarm code used to query an alarm description was not recognized. Possibly a user alarm is active, but User Alarm Descriptions are not reported by Y_GetAlarmDesc.
9023	The Alarm Describer is fetching the description of the alarm code entered from an xml file.
9024	The Alarm History operation is already in progress.
9025	Specified points are collinear when not allowed.
9026	Specified points are coincident when not allowed.
9027	The radius, as measured from the center to start, end, and via points, is not consistent within the required tolerance of 1.0%.
9028	Via point (AusPoint) is missing when required, incorrect size, or otherwise invalid.
9029	Normal vector is missing when required, incorrect size, or otherwise invalid.
9030	Circle center could not be calculated due to numeric imprecision error.
9031	The read multiple alarms operation is already in progress.
MLX200 ErrorIDs	
9216	Invalid Host_ID. Supported Host_IDs are (0 = MECHATROLINK group, 1 = MLX robot group, 2 = MotomanSync robot group.)
9217	Invalid Interface_ID. Supported Interface_IDs are (0..7)
9218	Invalid Device_ID. Device_ID must be 0.
9219	The groups motion engine generated an error. Use the MC_GroupReadError function block to obtain the GroupErrorID. Possibly a motion input such as velocity or acceleration are out of range.
9220	Group is not enabled. Enable the group using MC_GroupEnable.
9221	EtherNet/IP communication between the MPiec and the MLX robot interface was lost.
9222	State Transition Error. Refer to MC_GroupReadError for further details.
9223	Trajectory Shape Error. The value passed to MoveOptions.TrajectoryShape is invalid. Valid trajectory types are 0 = Trapezoid and 1 = S-Curve.
9224	Profile Unit Error. The value passed to MoveOptions.ProfileUnit is invalid. Valid values are 0 (% of maximum) or 1 (Absolute units).
9225	Invalid Control Mode. The group is set for Jogging or Manual mode, and an MC_MoveLinear or similar function block was executed, or Y_GroupJog or similar function block was called while the group was set for Automatic mode.
9226	IdentInGroup not found in AxesGroup.Axis.Label[]
9227	The Conveyor Position is already greater than RecordedPosition +Track-Options.SyncIn.EndDistance before MC_TrackConveyorBelt became Active.
9228	An unsupported value of Y_VelocityUnit was used as the VelocityUnit input for the function block. Valid values are 0, 1 and 2.
9229	The AxisRef passed into the function block does not match any AxisNums in AxesGroup.AxisRef[].AxisNum
9230	MLX_Driver startup error. Possible causes: 1) Verify that the correct IP address for the remote hosted controller is configured. 2) Possibly the EIP_Output structure is not mapped to the correct hardware address. 3) Try slowing the cyclic task interval of the MLX_Driver FB to 8 mSec.
9231	MLX_Driver startup error. Possibly the EIP_Input structure is not mapped to the correct hardware address. The Ethernet/IP status word has a value of zero, which should never occur. Open the Hardware Configuration and save again. The Hardware Configuration configures the necessary variables and maps them to the appropriate hardware address.
9232	Trajectory Type Error. An invalid trajectory type input was used for the function block. Valid trajectory types are 0 = Axis and 1 = Tool Center Point (TCP)
9233	An invalid value was passed to the TCPCoordinate input. Valid value are (0, 1, 2, 3, 4, 5) which correspond to (X, Y, Z, Rx, Ry, Rz)
9234	A watchdog error for data passing between the MPiec and an MLX Ethernet/IP interface has occurred

ErrorID	Description
9235	Module Info Read Error. There was an internal error in the function which reads the MLX module information.
9236	Ethernet/IP communication Error. The status variable for the MLX interface does not indicate healthy communication. (It is not 1000 Hex.)
9237	When there is more than one robot configured on an MLX interface, there must be one MLX_Driver for each robot. The first function block to run is designated as the primary. A secondary MLX_Driver will indicate this error if the primary MLX_Driver function block is no longer 'Valid'.
9238	Invalid [Origin, XX, XY] points. Origin coincides with XX, or Origin coincides with XY, or XX coincides with XY or Origin to XX is parallel to XX-XY.
9239	Invalid input or output frame. This function supports WCS, MCS, and PCS.
9240	Calculation leads to a singularity. Calculated output coordinates is in a singular or gimbal lock configuration.
9241	The AxesGroup BaseOffset, ToolOffset, or PartFrameOffset was changed while this function block was enabled.
9242	The MLX interface has a firmware version that is not compatible with the MLX200_**** user library in the project. Check the MLX hardware to verify the firmware version. Either change to the appropriate user library or send the MLX200 unit to Yaskawa for a firmware upgrade.
9243	The MLX reports Error 17. Possible causes include: HardwareMode was selected, but the MLX could not connect to the ServoPacks via EtherCat.
9244	The HardwareMode selected does not match the MLX operation mode. Possible causes are: 1) The HardwareMode cannot be changed after the Enable input goes high. 2) If the MLX is configured to support more than one robot, all robots must be set for the same HardwareMode. 3) The MLX may require a reboot to enter Hardware mode.
9245	Setting the Tool Transformation offset failed. The expected TCP position was out of range of 0.1 units on one or more of the axes.
9246	One of the GroupInputs is in a state which prevents this function block from executing.
9247	Use the MLX_Conveyor_Config Function Block to configure MLX200 Conveyors.
9248	ConveyorTracking, PalletSolver, or MotoPick is not installed on the MLX200
9249	The Group's E-Stop input is preventing motion.
9250	The Group's guard circuit input is preventing motion.
9251	One of the Group's interference zones is violated.
9252	The Group's live man switch is preventing manual mode operation.
9253	The Group's Safety circuit is preventing motion.
9258	MC_GroupSetPosition.SetMode can only have values of 0, 1, or 2.
9259	MC_GroupSetPosition cannot be used on remote hosted groups with simulated or virtual axes.
9261	Interference Zone can't be activated during motion.
9262	General SetFrameOffset error. One of the lower level functions returned an error. Please call Yaskawa America motion applications for assistance.
9263	There is more than one scanner (EIP master) device trying to communicate with the same remote hosted robot controller.
9264	Although communication between the MPiec controller and the remote hosting controller has been established, valid data read is not possible. Try re-enabling the function block.
9265	The required Option Monitor value is not retaining the required UserFeedbackDataType. Maybe more than one function block is trying to use the same service.
9266	For remote hosted groups such as MLX200, the Y_GroupInputs function block cannot be used when MLX_Driver.HardwareMode = TRUE.
9267	Wrong MPiec controller hardware for a configured group mechanism. Part number ending in "-RBT" or "-ER" is required and a minimum of firmware version 3.4.
9268	Invalid Tool Number. Valid Tool numbers range from 0 - 63.
9269	This function block must be executed in a task with an update interval of 4 or 8 mSec.
9270	Robot Controller Key switch or E-Stop error. Robot Controller programming pendant Key switch might not be set to 'Remote' or the E-Stop is pressed. Set the Key switch to 'Remote' and verify that the E-Stop is released, then re-enable MS_Driver.

ErrorID	Description
9271	Robot Controller is configured for the wrong endianness. Either downgrade the MSync user library to MSync_3f220, or configure the Robot Controller for MotomanSync as BIG endian.
9272	Motion or command buffer must be empty before executing.
9273	It took longer than normal for the axes to be switched to the requested power state. Make sure that there are not multiple instances of Y_GroupPower or other MC_Power or AxisControl FBs for the same group or axis as this can cause conflicting commands and therefore a timeout. This error may also occur if Y_GroupPower is executed in a task with an interval less than 4 mSec.
9274	Wrong MPiec controller hardware for 5 or 6 axis mechanisms. Part number ending in "-ER" is required and a minimum of MPiec firmware version 3.4.
9275	Remote controller keyswitch on robot controller must be in remote
Toolbox ErrorIDs	
10020	ProductSize cannot be less than or equal to zero.
10021	Maximum allowed consecutive missed registration marks reached.
10022	Product or circular buffer overrun / full.
10023	Buffer size too small / cannot be zero.
10024	DataSize must be greater than zero.
10025	SensorMinimum must be less than SensorMaximum.
10026	Positive Position Limit must be greater than Negative Position Limit.
10027	Negative Position Limit must be less than Positive Position Limit.
10028	Positive Velocity Limit must be LREAL#0.0 or greater.
10029	Negative Velocity Limit must be LREAL#0.0 or lower.
10030	Positive Acceleration Limit must be greater than 0.
10031	Negative Acceleration Limit must be less than 0.
10032	Positive Deceleration Limit must be greater than 0.
10033	Negative Deceleration Limit must be less than 0.
10034	Interpolation calculation error.
10035	Gripper Close Error (Timeout).
10036	Latch Error. LatchReference was negative. This situation should never occur. Verify that the normal axis movement is in a positive direction. Use PLCopen Toolbox v340 which contains improved code for applications with registration marks near the end of the default move. DCR 1183
10037	Offset cannot be in the same direction as the original motion into the limit switch.
10038	CamData.LastSegment must be greater than 0 and less than 400, or whatever value has been declared as the ARRAY size in the CTB_Types file.
10039	Cam Segment 'Resolution' cannot be zero unless the CurveType is TB_CurveType#StraightLine. Another cause of this error is if the Resolution is greater than the master delta for a given segment.
10040	Curve Type selected in a segment is not valid.
10041	Total pairs required would exceed DataType definition for MS_Array_Type based on number of segments and resolution settings in CamData.
10042	Master must be always increasing from segment to segment.
10043	Tangent Match formula error, cannot have only one segment.
10044	Tangent Blend error, must have two segments, a straight line and a Tangent Blend, in either order.
10045	SlavePosition not found in Y_MS_CAM_STRUCT.
10046	Both cam tables must have the same number of point to be added together.
10047	Both tables must have the same master cycle to be added together.
10048	The IndexSpeed is less than 20.
10049	Frequency cannot be less than 1 Hz.
10050	The dwell cannot be greater than the IndexTime.
10051	There must be a whole number of oscillations in an index at a given speed.

ErrorID	Description
10052	There is a discrepancy between the master values in Profile1 and Profile 2. At the same pair somewhere in the table, the masters have values differing by more than 1 user unit.
10053	DataPoint Error.
10054	One of the Segments in the PathData has an invalid Segment Type. Valid Segments types are defined in the Group Toolbox GroupTypes file as enumeration GTB_SegmentType. This error may also be the result of a mismatch in support for certain G-Code / Segment Types based on the user's MachineStruct.MachineType setting.
10055	The absolute sum of the motion for all axes relative travel from the previous segment cannot be zero. One axis must always be in motion from segment to segment, otherwise the virtual master distance cannot be calculated.
10056	Arc Error.
10057	Point Error.
10058	The start angle must be a value from 0.0 to 360.0 degrees.
10059	The axes got out of sync during the path motion. All Cam Slaves InSync output must be on or off at the same time, or this ErrorID is generated.
10060	The axis must be configured as a rotary type for this function block to be applicable.
10061	MasterType is something other than 0 or 1.
10062	MachineCycle must be a positive value if MasterType = 0
10063	LastSwitch is set outside the 0-255 range. Sigma7_FT62 feature supports switch with the range of 0-31
10064	Track Number outside the 0-31 range. Sigma7_FT62 feature supports track(signal) with the range of 0-
10065	FirstOnPosition is not equal to 0.
10066	LastOnPosition is not equal to 0.
10067	AxisDirection is not equal to 0. Direction enumeration is ranging from 0 to 2.
10068	CamSwitchMode is not equal to 0. Sigma7_FT62 supports both position mode or time mode, which is 0 OR 1.
10069	Duration is set to 0 or a negative value.
10070	OnCompensationScaler is set to an invalid value.
10072	ImproperOnPos_SetError.
10073	OnOffPosition_Error. In Sigma7_FT62 feature, when it is position mode, FirstOnPosition and LastOnPosition can't be the same value.
10074	Direction must be 0 for positive, or 2 for negative.
10075	Calibration Error: Cal_X2 must be greater than Cal_X1.
10076	WindowSize must be greater than zero.
10077	Cubic Spline maximum number of consecutive segments exceeded. DataType definition for the Matrix could be increased if necessary.
10078	Formula 27 Error is reserved for errors with circle calculations.
10079	When using UserNoDwellModifiedConstant Velocity, there must be three contiguous segments with the same formula code applied, and the master percentages must be increasing.
10080	Formula 29 error.
10081	ControlData.DecisionPosition is 0. The position to determine when to disengage the cam cannot be less than or equal to zero.
10082	Mode Error. ControlData.Mode can only be 1 (one way cam) or 2 (two way cam).
10083	Unsupported Cubic Spline Sequence.
10084	One of the Cam Tables has an invalid TableID.
10085	Arc Error. The direction must be 1 or -1 and the radius must be greater than zero.
10086	MaxPosCorrection must be zero or positive, MaxNegCorrection must be or zero or negative.
10087	Arc Error. The arc radius cannot be less than half the distance between the two points that define the arc.
10088	Arc Error. The arc profile makes the x axis (master) go backwards.
10089	Bezier Error. There should be a straight line segment before and after the Bezier segment.
10090	LREAL value too large to convert to the specified ASCII representation.

ErrorID	Description
10091	Checksum Error.
10092	Invalid Method.
10093	Rate is less than or equal to 0.
10094	S_Scans is less than 2 or greater than 30000.
10095	TiError - PIParameters.Ti is set to ≤ 0.0 . This variable must be a positive number'.
10096	Time Period value for RateCalculator FB (e.g. ppm) is less than or equal to 10 ms.
10097	Bezier Slope Error. The slopes of the two straight lines before and after the Bezier segment should have slopes with same signs. If the slopes are positive, the slave end point should be GE slave start point. If the slopes are negative, slave end point should be LE slave start point.
10098	BufferPatternSize cannot be equal to 1 or greater than 10.
10100	Both axes must be configured for the same axis type (Rotary / Linear) and if Rotary, they must have the same Machine Cycle.
10101	Invalid Mode Error: The specified mode is invalid. The acceptable modes are 0:Calculate Gains, 1:Run
10102	Temperature Out of Range Error. The actual temperature is above or below the acceptable range.
10103	Response Error. The specified tuning response is invalid. The acceptable responses are 1:Slow, 2:Medium, 3:Fast
10104	Output Limit Error. The lower limit must be less than the upper limit.
10105	Sample Time Error. The sample time cannot be 0. Responsible for divide by 0 error.
10106	Clock Error. The clock was left unconnected.
10107	Cycle Error. The system must be able to tune for at least 1 cycle. Check CycleCount and IgnoreCycle variables.
10108	Timeout Error. The system has taken too long to complete auto tuning.
10110	Too many tabs specified.
10111	Pitch between labels would be negative, need more spacing between tabs.
10112	Tab mode must be specified as 1 (Tabbing) or 2 (Stamp).
10113	Incorrect cam table size (check the CamTable.Header.datasize)
10114	Incorrect cam table size (check the CamTable.Header.Datasize).
10115	XML Tag not found. Tip: If this error is generated from an application with a group configured, and the AxesGroup has additional non group axes added to it such as for tangent operation, confirm that the AXIS_REF specified in the AxesGroup.Axis_Ref[] matches the intended AXIS_REF of the additional axis. See the Toolbox manual, Group Toolbox section on "Tangent Mode." Otherwise, possibly the file is corrupt or the schema is not compatible with this function block.
10116	Problem converting string data to the output buffer.
10117	The controller already has a String Conversion Error at the rising edge of this function. Clear the alarm using Y_ClearAlarms and try again.
10118	STRING_TO_BUF Conversion Error.
10119	In the Data Structure, rows must be set greater than zero and columns must be set greater than zero.
10120	File could not be opened. Check for accurate directory path and use of "/"
10121	The CSV file was written in a format unsupported by this function block.
10122	Row Error. The data is out of sync with the expected row / column arrangement expected.
10123	Column Start Error. The data is corrupted.
10124	Unsupported Case condition.
10125	Conversion Error. Check the ErrorRow and ErrorCol / ErrorString outputs for details.
10126	NoDataError - The End Of File was reached, but the record count is zero. Verify the file is not corrupted.
10127	TooManyRecords - DataType is not large enough.
10128	MaxNotDefined - The user must set the maximum number of records that can be added to the structure.

ErrorID	Description
10129	No Carriage return found in CSV buffer. The function searched the file for twice the length of the specified buffer and was unable to find a carriage return indicating the end of a row. Either the buffer size is too small, or the data is invalid.
10130	The center to co-ordinate distance for the two input co-ordinates are not the same
10131	Zero radius is invalid.
10132	Only modes 0 (center + 2 co-ordinates) and 1 (radius + 2 coordinates) are supported.
10133	The coordinates of the two data points are the same.
10138	The positions of the main and prime axes are outside the specified Allowance to permit motion on the prime axis.
10139	VelocityScaler cannot be less than or equal to zero.
10140	Must be greater than zero and less than 20.
10141	Same Points Error - The 2 points given are equal.
10142	Negative Radius Error - The Radius given is less than or equal to 0.
10143	Incorrect Circle Info - incorrect inputs were given for the Direction or PathChoice.
10144	Points Too Close Error - The smallest possible diameter for a circle given two points can be calculated as the distance between both the points.
10150	Theta 1 Below Minimum.
10151	Theta 1 Above Maximum.
10152	Theta 2 Below Minimum.
10153	Theta 2 Above Maximum.
10154	Imaginary ChordHeight (impossible for mechanism).
10155	Maximum Compression Reached (Mechanism squats too deeply).
10156	Locked Leg at Knee Joint B (Link2-Link3).
10157	Locked Leg at Knee Joint D (Link1-Link4).
10160	CommandString length is invalid.
10161	Invalid CommandCode.
10162	Parameter being searched for is out of range.
10163	Mode input not valid.
10164	Invalid character position input.
10165	CommandString length is too long or command delimiter not found.
10166	File Not Found.
10168	Buffer Size Error.
10170	Maximum deviation between motor and encoder was exceeded.
10171	Home approach distance limit encountered. Latch input not detected within the approach distance limit.
10172	Use Pointer Error when using MultiUsePointers.
10173	Duplicate latch values detected. Consider upgrading to firmware 3.4 or higher to avoid this issue. SCR 10736
10174	Out of range Error. When GridLookup started executing, the Coordinate position was already out of the range of the measured region.
10175	Too many custom G or M Code segments were deferred for future processing to be synchronized with buffered motion preceding the custom codes.
10176	Unrecognized motor model number.
10177	There was an internal problem calculating the segment length. Contact Yaskawa for support.
10178	There was an internal problem calculating the segment time. Contact Yaskawa for support.
10179	There was an internal problem managing segments during Tool Compensation. Contact Yaskawa for support.
10182	Speed Match Range Error: SpeedMatch start angle <= SpeedMatch end angle.
10183	Sensor Distance Error: Sensor distance <= Part length. Cam blend cannot be accomplished with such a set up.
10184	Engage Distance Error: Safe Engage Distance <= Part length. Cam Blend cannot be accomplished with such a set up.
10190	Part Length Error: PartLength input must be greater than zero.

ErrorID	Description
10191	Speed Match Distance Negative: The SpeedMatchDistance input must be greater than zero.
10192	Master Speed Match Position Negative: The MasterSpeedMatchStartPosition must be greater than zero.
10193	Speed Match Length Error: The PartLength must be > SpeedMatchDistance.
10194	Stroke Length Exceeded: The calculated stroke length is greater than the allowable ShearStrokeLength. Increase the allowable shear stroke length or reduce the speed match distance.
10141	Same Points Error - The 2 points given are equal.
10200	Region Error: The min and max values for the region size are swapped.
10201	No Outputs Error: The function did not find an output on the tooling which resides in the area for the part to be picked.
10202	Duplicate Output Error: The function tried to allocate an output for a part which was already assigned to another part.
10203	Container Error: CP_Struct.LastContainer must be greater than zero.
10204	Part Error: CP_Struct.LastPart must be greater than zero.
10205	Layer Error: CP_Struct.LastLastLayer must be greater than zero.
10206	Pick Error: Each CP_Struct.Layer[n].LastRegion must be greater than zero.
10207	Pick Dimension Error occurred when calculating the size of the region to be picked.
10208	Region Location Error.
10209	Location Error: CP_Struct.LastLayer must be greater than zero.
10210	Region Error: All parts in a region must be from the same location / conveyor system.
10211	Tool Error: CP_Struct.Tool.XTips and CP_Struct.Tool.YTips must be greater than zero.
10220	Encoder ConfigurationError. Encoder must be configured as an absolute encoder for Sigma7_FT62 feature. Set Pn002.2 to 0.
10221	Invalid Enumeration. Sigma7_FT62 high speed output signal enumeration does not match acceptable values.
10222	Reboot Required Error. Reboot the controller to complete writing of non-volatile memory.
10223	Out Of SyncError. The current encoder position is out of sync with the current output and checkpoint positions.
10224	Output Delta Error. The output position cannot be changed by more than +/- 45 degrees in a single machine cycle.
10225	Excessive Pos Error: Configured output distance exceeds one machine cycle. Decrease position setting to within one machine cycle.
10226	Excessive Distance Error: Configured output distance exceeds one machine cycle. Decrease the distance setting to within one machine cycle.
10227	Flash Memory Write Error: A failed attempt was made to write to the drive's non-volatile flash memory. The drives non-volatile memory cannot be written while the axis is enabled. Disable the axis and try again.
10228	Invalid Cam Switch Mode Error: The configured CamSwitchMode does not correspond to a valid enumeration value.
10230	Winder1RevPulse cannot be less than or equal to zero.
10231	LineRoll1RevPulse cannot be less than or equal to zero.
10232	LineRollDiameter cannot be less than or equal to zero.
10233	WinderInitDiameter cannot be less than or equal to zero.
10234	WinderMinDiameter cannot be less than or equal to zero.
10235	WinderMaxDiameter cannot be less than or equal to zero.
10236	Normal_Rate cannot be less than or equal to zero.
10237	E_Stop_Rate cannot be less than or equal to zero.
10238	Sshape_Time cannot be less than or equal to zero.
10239	Rate cannot be less than or equal to zero.
10240	PI_Parameters.RefRate cannot be less than or equal to zero.
10250	Data Size Mismatch. The size of the Temperature Control Parameter File is different than the size of the structure it must be copied to. Possible data corruption.

ErrorID	Description
10251	Version Error. The file version of the Temperature Control Parameters is not recognized by this version of the function block.
10600	Unsupported Letter Code. A G-Code started with a character that was not recognized.
10601	Unsupported G-Code
10602	Unsupported M Code
10603	PathData is currently in use by MC_MovePath, it is not possible to START reading data into the structure until MC_MovePath is Done.
10604	Circle Error. When specifying an arc (G02 or G03), both the I and J registers cannot be zero.
10605	Offset Error. G10 'P' parameter must be 1 through 9.
10606	User Unit Error. An invalid combination of user units between the Hardware Configuration and the G-Code data was found. Example: HC is configured for revolutions, and the G-Code file specifies mm. The G-Code Processor can only convert between linear units.
10607	Segment Error. A function inside MC_MovePath could not find a SegmentID for the current motion that matches one assigned when the motion function block was executed.
10608	Tool Compensation Error - There was no valid combination of motion segments (Line-Line, Line-Arc, Arc-Line, Arc-Arc)
10609	Tool Compensation Error - There was no valid solution found for an Arc - Arc combination
10610	Tool Compensation Error. No Solution Found (Logic Error)
10611	Division by zero.
10612	Tool Compensation Error. A segment transition from line to line, line to arc, arc to line, or arc to arc was not detected.
10613	Tool Compensation Error. No solution found for an arc to arc transition.
10614	Tool index as specified in the 'P' register must be between 1 and MaxTools, which is the size of the ToolDataStruct in the Group Toolbox G-Code datatypes file.
10615	G10 Error. The 'L' register must be 1 or 2.
10616	OperationMode Error. The VAR_INPUT is requesting "Infinite Repeat" but the path is too large to fit within the PathData.Segment struct at once, and the beginning of the path was overwritten. Infinite repeat mode is only possible if the entire path can be contained in the PathData structure.
10617	Group Name Error. Check AxesGroup.Name for validity.
10618	ControllerInfo Error. Connect a Global variable of datatype CONTROLLER_INFO and locate it at address %MD3.66560
10621	Final Segment Error. The PathData.FillMethod is set as n/a and the PathData.FinalSegment is zero. Please set PathData.FinalSegment properly. Tip for G-Code users: Make sure that either the Read_Code_File or Read_GCode_Stream functions are 'Busy' before executing MC_MovePath.
10622	Tangent configuration error. Maybe the Tangent axis is configured as a theta axis of the group, and not as an external axis. (Must not be a group's theta axis.)
10623	Max Segments Error. AxesGroup.Status.FreeMotionSegments <= (MachineData.Prms.MaxSegmentsPerScan * 2). Either increase the Motion Queue Size in the Hardware Configuration for this group, or reduce MyMachine.MaxSegmentsPerScan.
10624	Emulation Error. MachineData.Emulation is not set to a valid value.
10625	Program List Error. There are more sub program calls in the data than are supported by the G-Code functions. (O register)
10626	Stack Overflow. There are more Jump Segment Types in MC_PATH_DATA_REF than the defined size for PathData.Logic.Events
10627	Variable Error. A line that starts a variable assignment is missing an equal sign, or the variable index is out of range [1 to 1000]
10628	The canned cycle G-Code is not supported.
10629	Cycle Array Error. The G-Code data contains more information for a canned cycle than an internal structure which holds all the XZ information contained within the P & Q line numbers.
10630	Canned Cycle Finishing Error. Possible causes are: #1) (End - Start) block must be at least one block. 2) The range of commands to be repeated must come after G71.

ErrorID	Description
10631	A command was encountered that is not supported for the configured MachineType. For example, M270 is found (for 3D printing,) but MachineData.MachineType <> GTB_MachineType#Printer
10632	Joint Not Configured. Could not find a Degree of Freedom in the HomeData. For Example, in HomeData.Sequence[1].DOF[1] :='Z', 'Z' was not found in the AxesGroup configuration.
10633	Queue Size Error occurs if the Motion Queue Size set in the Hardware Configuration is larger than the DataType MC_PATH_DATA_REF.Segment[] in the Group Toolbox.
10634	The Motion Queue Size set in the Hardware Configuration is larger than the DataType MC_PATH_DATA_REF.Segment[] in the Group Toolbox. Adjust the Motion Queue Size so it is smaller than the size declared for MC_PATH_DATA_REF.Segment[].
10635	Program List Error. The G-Code data refers to more program labels (typically from an IF or M98 command) than the declared size of ProgramList.Name[], which is typically 16.
10636	Logic Type Error. The command is not formatted correctly. Examples: N25 IF [#500 LT 25] N35 or N10 #3=0
10637	Conversion Error. A logical expression could not be converted from a byte array to a string. Contact technical support for assistance.
10638	JumpError. The IF instruction must include a line number reference as Nxx where the first character is "N" and the remaining characters are numeric. Example: N25 IF [#500 LT 25] N35
10639	Operand Error. The following operands are supported in an IF instruction: EQ, GE, GT, LE, LT, NE, =, >=, >, <=, <, <>
10640	Tool Length Error - G-Code Register._H is not 0, 1, or 2.
10641	Evaluation Error. There was something wrong with a logical command such as a missing # sign, unsupported or improperly formatted logical comparison, or divide by zero.
10642	The MC_MoveAbsolute to touch the plane set its Done output before the specified torque limit was reached. Maybe the position specified for the contact measurement is not enough. It must be a position which cannot be achieved due to contact with the surface.
10643	Input product has an invalid ProductType. Must be 0-49.
10644	Specified ConveyorBelt not defined in DistributionConfig.
10645	DistributionConfig has an invalid SequenceLength for the specified ConveyorBelt. The valid range is 0-16. Check DistributionConfig[x].SequenceDistribution.SequenceLength, where x is the element that specifies sequence parameters for the selected ConveyorBelt.
10646	DistributionConfig has invalid sequence element values for specified ConveyorBelt. Must be 0-7. Check DistributionConfig[x].SequenceDistribution.SequencePattern[] where x is the element that specifies sequence parameters for selected ConveyorBelt.
10647	GroupNumber is invalid. Must be 0-7.
10648	Specified ProductNumber is invalid. Must be 0-49.
10649	Number of products is greater than maximum allowed.
10651	The sample size must be between 0 and 32.
10652	Y_ProbeContinuous is configured incorrectly to catch both the rising and falling edge of the incoming part. SlaveBuffer.PatternSize must equal 2, .PatternArray[0] must be 1, .PatternArray[1] must be 2, and Pn511.1 must be 4 and Pn511.2 must be E.
10653	Incoming products have a part length delta that exceeds the defined tolerance.
10654	The StartPoint must always be less than the EndPoint.
10655	The specified Degree is invalid. The acceptable range is 7 to 31.
10656	The specified Mode is invalid. The acceptable modes are 1:SyncInMode, 2:GapMatchMode, 3:SyncOutMode.
10657	The sensor distance must be greater than 0.0
10658	The tolerance must be greater than 0.0
10660	Selected Extruder Error. Up to three extruders are supported. The G-Code data specified a T value which is out of range.
10661	Unconfigured Extruder Error. The G-Code data specified a valid extruder number as Tn but the corresponding extruder is not part of the group configuration. Check the MachineStruct.Variant setting, possibly a feature being accessed is not supported given the variant.

ErrorID	Description
10662	Grid Setup Error. X or Y range is less than or equal to zero, or the resolution is less than 3 points.
10663	Group name not found in Group.XML configuration file. Contact Yaskawa Motion Support.
10664	Structure version or size mismatch. Data received from DLL does not fit the expected data format. Maybe using incompatible DLL and Toolbox.
10665	Set Position Error. Only 3D Printer Extruder 'E' axes are supported.
10666	ExtruderOffsetInvalid. Extruder offset has not been revalidated since the extruder has been changed. Make sure that the extruder position is set with a G92 command before attempting to move it.
10667	Feed Mode Error. When using Inverse Time Mode (G93) each motion instruction must contain an 'F' register feedrate value.
10668	Parameter Error. A G-Code register contained an invalid (non numeric parameter)
11050	Cam correction (shift/offset) has been aborted by another function block.
11051	Segment Error, could not find the previous motion segment to determine change in XYZ coordinates. Maximum search is 20 segments. Contact Yaskawa America for support.
11052	There are no degrees of freedom defined for the Group. There could be a group configuration error. At least one of the first 6 AxesGroup.Machine.Label[] must be populated with a string name, or the AxesGroup.HostID must indicate a remote hosted robot.
11053	Specified Degree of Freedom not found. (X,Y,Z,Rx,Ry,Rz) These are case sensitive and must match a Group Label from the Hardware Configuration.
11054	Tolerance Too Wide Error. The tolerance specified is larger than the distance between the Approach and Measurement Positions.
11055	Premature Touch Error. The measurement torque was achieved before the position was within GridSetup.MeasurementPosition + /- Gridsetup.Tolerance
11056	Transfer, Approach, and Measurement Positions provided in the setup data are not in a progressive order
11057	The External Plane specified contains axes which are not servos, which is required for this function to operate when the group is configured with a virtual axis as the joint used for measurements.
11058	No axes specified for the Plane Joint which is required for this function to operate when the group is configured with a virtual axis as the joint used for measurements. Populate Setup.ExternalPlane with the AXIS_REF of the servo.
11059	There are more secondary axes in the AXES_GROUP_REF than there is room allocated in the Plane Ref
11060	The groups is configured with a virtual axis operating the plane to be measured and Setup.ExternalPlane is not configured to reference the real servo axis to be operated.
11061	The mechanism is not supported by this function.
11062	Stream Watchdog. The KeepAlive packets were not received by Read_GCode_Stream within the specified InactivityTimeout.
11063	There were too many expression commands in the file. See the Group Toolbox DataType definition for LogicArray: ARRAY[1..256] OF LogicData; to see or adjust the size
11064	Rotary Error. The R register was specified on a G0 or G1 move, but the C axis is not configured as LoadType = Rotary in the Hardware Configuration.
11065	Rotary Error. Rotary moves within one MachineCycle are only supported on the C axis.
11066	Unhandled Override Error. A code was declared to be overridden, but no supporting IEC 61131 code was added in the Custom_Code_* function blocks.
11067	Path Capacity Error. This was thought to be an internal error that should never occur. Please report this problem to Yaskawa America Motion Application Engineering.
11068	G53 Error. The Motion code in effect at the time of the G53 command must be G0 or G1, and no tool offsets can be enabled.
11069	Block Sequence Error. There are more than one command on a line and the parser could not determine how to arrange the proper command sequence.
11070	Sensor Config Error. MachineData.Printer.Extruder[SelectedExtruder].TempSensors is greater than the maximum size of the array for Temp Settings OR MachineData.Printer.BedTempSensors is greater the max size in the related structure.

ErrorID	Description
11072	No Previous Motion. An arc segment cannot be the first motion because a previous datapoint is required to calculate the arc parameters.
11073	No Axes Specified. PathData.Segment[].AxesFlags cannot be zero, at least one of the axes positions must be specified.
11074	Lookahead Error - Buffer capacity reached the limit while trying to look ahead the required stopping distance for a segment. This problem occurs when the average motion point density is high and the deceleration rate is low, requiring more lookahead capacity than the system can support.
11075	There was an internal problem sub dividing segments. Contact Yaskawa for support.
11076	Home Preset Error. There are no axes specified for the Home Preset selected.
11077	Block Sequence Error. All of instructions on the line did not resolve, not all of the commands were processed.
11078	There was an internal problem with the Segment SubType. Contact Yaskawa for support.
11079	G92 is configured to inhibit changing the rotation of the Work Coordinate System via registers A, B, C. See MyMachine.G92RotationInhibit.
11080	Drill Cycle Error. Check to make sure all the register data provided with the G73 or G83 command are valid. In particular, make sure the Q value is positive.
11081	Position Delta Calculation failure. The GCode_Processor was not able to determine the previous position. Call Yaskawa for support.
11082	Feature Combination Error. For example, variable support (MyMachine.Feature.X3) cannot be used in conjunction with other features including: Lookahead (Transition mode), Feature.X0 (Constant Accel Feedrate Override), and G41 / 42 Tool Compensation. See the Compass or Toolbox manual for the feature compatibility chart.
11083	L10 Looping Error. The L10 command does not include the required parameter information
11084	GroupHomeData.Sequence[].LastDOF is larger than the size of the GroupHomeData.Sequence [].DOF array
11085	GroupHomeData.LastSequence is larger than the size of the GroupHomeData.Sequence array
11086	Relative extrusion mode (M83) cannot be used in combination with Constant Acceleration Feedrate Override (MachineData.Feature.X0).
11100	BufferUseError. The buffer bank where data is to be recorded is currently writing data to the disk. This problem is due to a timing issue / race condition. (Recording is faster than writing. Consider changing the task interval or buffer size.)
11101	Precision must be greater than 0 and less than 16.
11102	reserved
11103	TimeLimitError. The max recording time must be in the range of 0 to 60 seconds.
11104	ChannelError. A maximum of 7 channels is supported.
11105	DuplicateFilenameError. The file name's cannot be identical.
11106	The reference file contains no data.
11107	The comparison file contains no data.
11108	The number of channel data in each file is different and cannot be compared.
11109	Reference data in the file was not located. The file may be corrupt or it was not created by the DataLogWrite function block.
11110	The Noise Window must be within the range of 0 to 20 samples.
11111	Percent deviation must be within the range of 0 to 100%.
11112	Cannot find matching values in the comparison file.
11113	A samples timestamp is less than the previous timestamp, the file may be corrupt.
11114	SampleRateError. The sample rate must be greater than zero.
11115	FileLocationError. The file can only be stored in Flash or Ram.
12000	Read response timeout, no response was received within the supplied TimeOut.
12010	Not a response (QR should be 1 but it was 0).
12011	Response was truncated because it extended beyond the 512byte UDP packet size.
12012	Recursive is not available but was requested by the Query packet
12021	Format error, the name server was unable to interpret the query.
12022	Server failure, the name server was unable to process the query due to an internal problem.

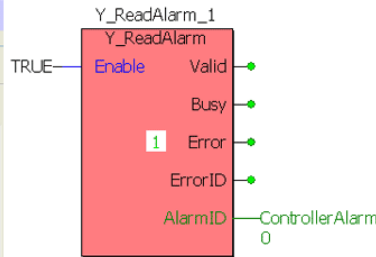
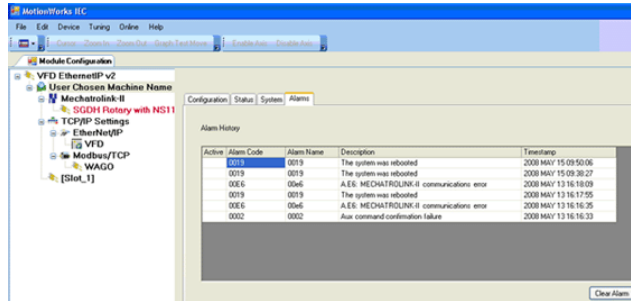
ErrorID	Description
12023	Name error, not valid for this block (only valid for Authoritative servers).
12030	Address length was less than 3 characters which is not possible.
12031	Address format was incorrect as it does not contain a '.'.
12032	Buffer Size Error. CommandType is set to 'Fixed'.
12033	Packet Size Error. CommandType is set to 'Fixed'.
12034	Comm Type Error - CommType must be set to either 'serial' or 'Ethernet'. Default of 0 is n/a.
12100	Connect to SMTP server timeout, no connection was established within the supplied TimeOut.
12101	DATA portion of e-mail was not successful and therefore the e-mail may not send/be malformed.
12102	QUIT error, there was an error sending the 'QUIT' command to the server.
12103	NumRcpt cannot equal 0.
12200	Connect to FTP server timeout, no connection was established within the supplied TimeOut.
12201	Connect to FTP data socket timeout, no connection was established within the supplied TimeOut.
12202	QUIT error, there was an error sending the 'QUIT' command to the server.
12203	The credentials for the FTP server were incorrect (either one or both username and password).
12300	File Error, no error information available.
12301	Invalid file handle.
12302	Maximum number of files are already opened.
12304	File is already opened.
12305	File is write protected or access denied.
12306	File name not defined.
12310	End of data reached.
12312	The number of characters to be read from file is greater than the data buffer.
12322	No data could be read from file.
12421	Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.
12425	Can't open data connection.
12426	Connection closed; transfer aborted.
12430	Invalid username or password.
12434	Requested host unavailable.
12450	Requested file action not taken / Requested mail action not take (mailbox unavailable).
12451	Requested action aborted. Local error in processing.
12452	Requested action not taken, insufficient storage space in system (FTP: File unavailable)
12500	Syntax error, command unrecognized.
12501	Syntax error in parameters or arguments.
12502	Command not implemented.
12503	Bad sequence of commands.
12504	Command not implemented for that parameter.
12521	[domain] does not accept mail.
12530	Not logged in / Access denied.
12532	Need account for storing files.
12550	Requested action not taken. File unavailable (e.g., file not found, no access) / Mailbox unavailable.
12551	Requested action aborted. Page type unknown / User not local.
12552	Requested file action aborted, exceeded storage allocation / Requested mail action aborted, exceeded storage allocation.
12553	Requested action not taken, file name not allowed / mailbox name not allowed.
12554	Transaction failed.
12560	Invalid Equipment Module number.
12561	Equipment Module not enable in the system.

ErrorID	Description
12562	Invalid number of enabled Control Modules in selected Equipment Module.
12563	Time rollover warning.
Axis Error	
40960	RESERVED
45332	Sending clear alarms command to ServoPack failed.
45333	Failed to reset Mechatrolink network.
45334	Mechatrolink reset cannot be performed when there is a axis enabled or in motion.
45335	Failed to initialize the absolute encoder.
45336	Function block could not be executed because a program download was in progress.
45337	Rebooting the controller is prohibited while an axis is enabled.
Operating System Error	
57620	The DataType connected to a function block parameter specified as ANY type does not match the required data size. Right click on the function block and select 'Object Properties' to determine which parameters are ANY type. The size of the variable connected to these parameters is not checked during the compilation but validated at run time. Typically errors occur when a variable of type AXIS_REF is not connected to an Axis VAR_IN_OUT, or a variable of type Y_Engage_Data is not connected to the Y_CamIn function block.
57873	InvalidStructureSize. The structure size does not match. Check all the variables connected to the function block. A common mistake is to connect a structure element, not the entire structure. Example: EngageData.StartMode is connected instead of just EngageData
57874	Argument data is NULL. The EngageData input must be connected.
Kernel Error	
60909	This function block is not supported for the AxesGroup specified - OR -.... NOTE: Starting with MotionWorks IEC release 3.5.0, a new strategy for supporting remote hosted groups was implemented. When operating remote hosted robots groups via MotomanSync, the MSync_3**** user library MUST appear above the PLCopen Part4 library in the libraries folder of the Project Tree Window, or this error may be erroneously thrown.
Unsupported functions based on Group Type	
61713	This function block caused an internal error. Possible causes: MC_Power - Check if multiple instances of this block are executed for the same axis. Y_CamIn - Check in the cam table if the master values are the same for two datapoints or decreasing. Y_CamStructSelect - Y_MS_CAM_TABLE.Header.DataSize must not be zero.



Controller AlarmID List

The following is a list of alarm codes that are reported in the Hardware Configuration's Controller Alarms tab, [Y_ReadAlarm](#) and [MC_GroupReadError](#) function blocks, and the webUI. These are non axis specific system alarms.



	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
	ErrorClass+AxisErrorID output from MC_ReadAxisError		
motionKernel	1201	0103	An alarm task queue was full when a new alarm was posted. This indicates that the task is being starved of execution time or that the system is generating many alarms simultaneously.
app	1401	0005	The script environment ran out of memory. This is a serious condition because it may prevent further errors from being handled correctly.
app	1401	0006	An error occurred while running the standard error handler for a general script error. This is a serious condition because it indicates the standard error handler is malfunctioning.
app	1401	0007	This error should never occur and is included only for completeness. It indicates that an unknown and potentially fatal problem has occurred within the script engine.
app	1401	000A	The script task failed to stop cleanly, which may result in unreleased system resources. Error recovery requires the controller be reset.
app	1401	000B	The command line task failed to stop cleanly, which may result in unreleased system resources. Error recovery requires the controller be reset.
app	1403	0002	The task responsible for publishing events to a remote client failed to stop cleanly, which may result in unreleased system resources. Error recovery requires the controller be reset.
app	1403	0003	The task responsible for replying to remote clients failed to stop cleanly, which may result in unreleased system resources. Error recovery requires the controller be reset.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
app	1403	0004	The task responsible starting and stopping connections to remote clients failed to stop cleanly, which may result in unreleased system resources. Error recovery requires the controller be reset.
app	1407	0001	The file system on which the configuration file directory resides could not be read and may be unmounted or corrupted. The system has booted in a minimal configuration mode, and most functionality is limited. If possible, the file system should be recovered or reformatted and new config files uploaded if applicable.
app	1407	0103	The watchdog timer expired.
app	1407	0108	A CPU exception occurred.
app	1407	0109	The firmware files on the controller do not match the expected checksums.
app	1407	010A	The manufacturing procedure failed. The controller probably could not fetch the current time from the network.
app	140A	0009	Network reset detected multiple Axes connected to the same servo network node.
app	140A	000A	Network reset detected multiple I/O connected to the same network node.
app	140A	0015	Controller memory was corrupted during network reset resulting in a lost logical Axis data structure.
app	140A	0016	Controller memory was corrupted during network reset resulting in a lost logical I/O data structure.
app	140A	0018	An Abort input specified in the configuration could not be found. The abort condition is considered permanently asserted. No motion is possible until the I/O configuration can be matched to the abort inputs (restart required).
app	140A	0021	Too many events were posted from the system ISR. The motion scan and servo net loop have been shut down.
app	140B	0002	The controller ran out of free memory, possibly resulting in an unrecoverable failure. Please reboot the controller. A common cause for this alarm is when an application uses camming, and continues to create new cam tables without releasing the memory (Y_ReleaseCamTable) for old cam tables that are no longer needed. See the Cam Toolbox for a function to help called "CamTableManager." If using the Gantry toolbox and the MovePath function block, a similar function called PathIDManager is available.
app	140B	0004	The largest free memory block is too small, possibly resulting in an unrecoverable failure. Please reboot the controller.
app	140C	1035	The manufacturing data on the controller is invalid. The controller needs to be returned to Yaskawa for reprogramming.
Mechatrolink	2301	0001	The drive returned an invalid watch dog code indicating a possible dropped communication packet.
Mechatrolink	2301	0002	The drive failed to return confirmation of last aux command within the default timeout period.
Mechatrolink	2301	0003	An unrecoverable error occurred during auto configuration. As a result, one or more drives are excluded from the servo network.
Mechatrolink	2301	0004	Overriding the auto configured axes parameters failed. As a result, one or more drives are excluded from the servo network.
Mechatrolink	2301	0005	Two or more nodes have the same ID. As a result, all servo network communication has been suspended.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
Mechatrolink	2301	0006	The controller must be the root node on the servo network. All servo network communication has been suspended
Mechatrolink	2301	0007	The servo network communication device failed to initialize. Servo network communication is not possible.
Mechatrolink	2301	0008	An error occurred sending command to a node during initialization. The node may not support the configured communications rate. Communication with this node has been prohibited, but communication with other nodes may be possible.
Mechatrolink	2301	000E	The drive does not return response packet.
Mechatrolink	2301	000F	Bus reset generation that controller is not demanding.
Mechatrolink	2301	0010	It receives response with the same channel at the same Iso cycle.
Mechatrolink	2301	0011	The ID in the response packet is not same to ID of AxisNode.
Mechatrolink	2301	0012	The data length in the response packet is not same to value of CSR register(SEND_DSP_DATA_LENGTH) of drive.
Mechatrolink	2301	0013	The packet type in the response packet is not same S-DSP.
Mechatrolink	2301	0014	Invalid cycle time has passed with configuration file 'servonet.xml'. As a result, all servo network communication has been suspended.
Mechatrolink	2301	0015	Node is not found on 1394 network.
Mechatrolink	2301	0016	Invalid node.
Mechatrolink	2301	0017	Error matching node IDs.
DPRAM	2309	0001	Invalid watch dog code from drive
DPRAM	2309	0002	Aux command confirmation failure
DPRAM	2309	0003	Auto configuration failed
DPRAM	2309	0004	Overriding auto configuration failed
DPRAM	2309	0005	Invalid cyclic check sum from drive
DPRAM	2309	0006	Invalid watch dog from drive
DPRAM	2309	0007	Control mode is not supported
DPRAM	2309	0008	Communication with a node failed during servo network startup
motionKernel	3103	0100	Controller SRAM battery is low
motionKernel	3103	0101	The file system failed the integral consistency check. Remedy: Power up the controller in supervisory mode using the SUP switch. (On 3000 series controllers, this switch is labeled 'MNT' as in Maintenance.) Clear the alarm. Turn off the SUP switch. Power cycle the controller.
motionKernel	3201	0001	The motion kernel didn't request to enable axis. But, the axis is enabled.
motionKernel	3201	0002	The motion kernel didn't request to disable axis. But, the axis is disabled.
motionKernel	3201	0004	The encoder position stored in SRAM could not be validated. The value has been reset.
motionKernel	3201	0005	Main bus power was disconnected while the axis was enabled. Main power must be restored and this alarm cleared before motion can continue.
motionKernel	3201	0101	Configuration error: multiple alarm tasks with duplicate priority.
motionKernel	3201	0102	Configuration error: Alarm task not configured. Using default priority and name.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
motionKernel	3202	0001	Axis Coordinate System: The command position was outside the allowable range for the axis in the positive direction (positive overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	3202	0002	Axis Coordinate System: The command position was outside the allowable range for the axis in the negative direction (negative overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	3202	0003	Axis Coordinate System: The command speed was greater than the allowable range for the axis in the positive direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0004	Axis Coordinate System: The command speed was greater than the allowable range for the axis in the negative direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0005	Axis Coordinate System: The command acceleration was greater than the allowable range for the axis in the positive direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0006	Axis Coordinate System: The command acceleration was greater than the allowable range for the axis in the negative direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0007	Axis Coordinate System: The command torque was greater than the allowable range for the axis in the positive direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0008	Axis Coordinate System: The command torque was greater than the allowable range for the axis in the negative direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0011	Joint Coordinate System: The command position was outside the allowable range for the axis in the positive direction (positive overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	3202	0012	Joint Coordinate System: The command position was outside the allowable range for the axis in the negative direction (negative overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	3202	0013	Joint Coordinate System: The command speed was greater than the allowable range for the axis in the positive direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0014	Joint Coordinate System: The command speed was greater than the allowable range for the axis in the negative direction (overspeed). The axis may not be moved again until the alarm condition is cleared.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
motionKernel	3202	0015	Joint Coordinate System: The command acceleration was greater than the allowable range for the axis in the positive direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0016	Joint Coordinate System: The command acceleration was greater than the allowable range for the axis in the negative direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0017	Joint Coordinate System: The command torque was greater than the allowable range for the axis in the positive direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0018	Joint Coordinate System: The command torque was greater than the allowable range for the axis in the negative direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0021	World Coordinate System: The command position was outside the allowable range for the axis in the positive direction (positive overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	3202	0022	World Coordinate System: The command position was outside the allowable range for the axis in the negative direction (negative overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	3202	0023	World Coordinate System: The command speed was greater than the allowable range for the axis in the positive direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0024	World Coordinate System: The command speed was greater than the allowable range for the axis in the negative direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0025	World Coordinate System: The command acceleration was greater than the allowable range for the axis in the positive direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0026	World Coordinate System: The command acceleration was greater than the allowable range for the axis in the negative direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0027	World Coordinate System: The command torque was greater than the allowable range for the axis in the positive direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0028	World Coordinate System: The command torque was greater than the allowable range for the axis in the negative direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0031	The move specified would exceed the software position limits in the positive direction and was rejected before being started. The group may be moved again immediately if desired.
motionKernel	3202	0032	The move specified would exceed the software position limits in the negative direction and was rejected before being started. The group may be moved again immediately if desired.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
motionKernel	3202	0033	The move specified would exceed the software speed limits in the positive direction and was rejected before being started. The group may be moved again immediately if desired.
motionKernel	3202	0034	The move specified would exceed the software speed limits in the negative direction and was rejected before being started. The group may be moved again immediately if desired.
motionKernel	3202	0035	The move specified would exceed the software acceleration limits in the positive direction and was rejected before being started. The group may be moved again immediately if desired.
motionKernel	3202	0036	The move specified would exceed the software acceleration limits in the negative direction and was rejected before being started. The group may be moved again immediately if desired.
motionKernel	3202	0037	The move specified would exceed the software torque limits in the positive direction and was rejected before being started. The group may be moved again immediately if desired.
motionKernel	3202	0038	The move specified would exceed the software torque limits in the negative direction and was rejected before being started. The group may be moved again immediately if desired.
motionKernel	3202	0039	The predictive soft limit encountered a segment that doesn't support the predicted stopping point.
motionKernel	3202	0041	Cam and Contour tables must have a header indicating the number of rows and columns and a feed forward velocity flag. Comma separated data values following the header.
motionKernel	3202	0042	In CamTables, the first (master) column must be either increasing or decreasing.
motionKernel	3202	0043	In ContourTables, the first (time) column must start at zero and be increasing.
motionKernel	3202	0044	The master position was outside the range of the CamTable, which automatically stopped the cam motion.
motionKernel	3202	0045	One or more slave axes could not attain the target position and velocity within the user specified time limit for the Cam or Gear motion.
motionKernel	3202	0046	One or more slave axes could not attain the target position and velocity within the user specified distance limit for the Cam or Gear motion.
motionKernel	3202	0051	Axis enable failed. This problem is usually a result of communication problems with the servo drive.
motionKernel	3202	0052	Runtime computation detected an invalid motion parameter. This alarm ID can occur if a discrete move has to be completed but the commanded deceleration for that move is not sufficient. For example if a MC_MoveAbsolute aborts another move and the axis has to stop at a position that will come up in a couple of scans, but the deceleration input on the MC_MoveAbsolute is not high enough to make the desired profile, this alarm will occur.
motionKernel	3202	0061	The axis Positive Overtravel (P-OT) limit has been exceeded. Motion is prevented in the positive direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0062	The axis Negative Overtravel (N-OT) limit has been exceeded. Motion is prevented in the negative direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	3202	0100	The inverse kinematics computation detected a world position that can not be reached.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
motionKernel	3202	0101	The inverse kinematics computation detected that the elbow 'handedness' (orientation) does not match the configuration. The 'handedness' must be fixed by commanding the individual axes or manually moving the robot.
motionKernel	3202	0102	The robot XY position intruded into the configured dead zone area near the origin.
Mechatrolink	3301	0009	Some motor properties, such as encoder resolution, maximum speed, and maximum torque, could not be determined for the attached motor. The serial encoder may be malfunctioning, incorrectly programmed, or unplugged.
Mechatrolink	3301	000B	Setting of Pn002, digits 3 and 4, disables torque limit and/or velocity limit in velocity and/or torque control modes. Set Pn002 = xx11 to initialize. Saving in the Hardware Configuration will automatically set Pn002.
Mechatrolink	3301	000D	The servo network does not support this motion control mode.
Mechatrolink	3301	0018	The command position specified an instantaneous jump too large relative to the current position. Sigma-5 amplifiers give an A.94b warning and ignore subsequent position commands for any absolute position reference greater than 2,097,152 encoder pulses (2 revolutions of a 20-bit encoder). The controller watches for deviation between command position and actual motor position greater than 1,966,080 encoder pulses and issues an alarm. This is at 1.875 revolutions of a 20-bit motor little bit of margin. Sigma-II/III drives have a lower maximum following error limit of 1,048,576 encoder pulses. The position error limit on the Servopack (Pn520) should not be set greater than 1.875 rev = 1,966,080.
Mechatrolink	3301	0019	Setting of Pn002 digit 4 specifies torque feed-forward, but the SERVOPACK model does not support torque FF in position mode.
Mechatrolink	3302	00E4	The setting of the MECHATROLINK-II transmission cycle is out of the allowable range.
Mechatrolink	3304	0000	The base code for io alarms. The io's alarm value is bitwise OR'd in with this base value.
DPRAM	3309	0009	An error occurred sending command to a servo
DPRAM	3309	000A	The drive has an alarm
DPRAM	3309	000B	The data buffer for reading drive parameters via the messaging interface was too small
DPRAM	3309	1000	Error code prefix for data link errors
DPRAM	3309	100F	Servo check sum error for data link
DPRAM	3309	1010	Invalid function code for data link
DPRAM	3309	1040	Option card computed an invalid check sum
DPRAM	3309	1041	Invalid data size from the option card
DPRAM	3309	2000	Error code prefix for message errors
DPRAM	3309	2001	Unsupported message function code
DPRAM	3309	20A0	Controller option card detected bad CRC
DPRAM	3309	3000	Error code prefix for data link errors
Mechatrolink	3312	0000	The base code for inverter alarms. The inverter's alarm value is bitwise OR'd in with this base value.
Mechatrolink	3312	00EC	Power reset required.
Mechatrolink	3312	00ED	(Access not possible 10 consecutive times). Power reset required.
Mechatrolink	3312	00EE	(1s elapsed). Power reset required.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
app	3401	0001	The user script encountered an alarm, suspending its operation.
app	3401	0002	Script syntax errors are detected before the script is actually executed, during the pre-compile phase. The syntax must be corrected before the script can be run successfully.
app	3401	0003	Script runtime errors can be caused by a variety of incorrect script routines. The most common error is an attempt to use a 'nil' object where it should not be used.
app	3401	0004	The system could not find the file specified.
app	3401	0011	A data value argument provided to the API function was out of the expected range.
app	3401	0012	An argument provided to the API function was not the expected type.
app	3401	0013	An object argument provided to the API function was not the expected object type.
app	3401	0014	A scalar value was provided where a vector was expected, or a vector value was provided where a scalar was expected.
app	3401	0015	The script attempted to write to a read-only variable.
app	3401	0016	Use of that API function is not permitted with the current conditions and/or arguments.
app	3401	0017	The number of data values provided did not match the expected number of axes.
app	3401	0018	CamTable must have a header indicating the number of rows and columns and a feed forward velocity flag. Comma separated data values follows the header. The first (master) column must be either increasing or decreasing.
app	3401	0019	ContourTables must have a header indicating the number of rows and columns and a feed forward velocity flag. Comma separated data values follow the header. In ContourTables, the first (time) column must start at zero and be increasing.
app	3401	001A	It is prohibited to start a torque (or velocity) move when any moves other than torque moves (or velocity moves) are currently in progress or queued.
app	3401	00ED	'LastMove' events should be detected when a move completes normally or is aborted. However, the controller detected a situation in which the move finished but the event did not occur. Please submit an SCR.
communication	3403	0200	Invalid EtherNet/IP I/O configuration. Common causes of invalid configuration include duplicate t2o/o2t assembly instances or invalid client connection parameters.
communication	3403	0202	EtherNet/IP remote server unreachable. There is no route to the EtherNet/IP server. Common causes include: invalid remote server address, invalid gateway, invalid subnet mask, or the Ethernet network is not correctly configured.
communication	3403	0203	EtherNet/IP remote server unreachable. There is no route to the EtherNet/IP server. Common causes include: invalid remote server address, invalid gateway, invalid subnet mask, or the Ethernet network is not correctly configured.
communication	3403	0204	EtherNet/IP network unreachable. Unable to reach the network for the EtherNet/IP server. Common causes include: invalid remote server address, invalid gateway, invalid subnet mask, or the Ethernet network is not correctly configured.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
communication	3403	0205	EtherNet/IP connection refused. Remote server rejected connection attempt. The remote server may not be listening for connections or there may be a firewall preventing the connection.
communication	3403	0206	Too many EtherNet/IP connections. The Ethernet/IP client ran out of connection slot resources. Reduce the number of concurrent client connections.
communication	3403	0302	Error connecting to the Modbus TCP slave. Unable to connect to the Modbus TCP slave. Common causes include: invalid Modbus TCP slave address, invalid gateway, invalid subnet mask, or the Ethernet network is not correctly configured.
communication	3403	0303	Modbus TCP slave unreachable. There is no route to the Modbus TCP slave. Common causes include: invalid Modbus TCP slave address, invalid gateway, invalid subnet mask, or the Ethernet network is not correctly configured.
communication	3403	0304	Modbus TCP network unreachable. Unable to reach the network for the Modbus TCP slave. Common causes include: invalid Modbus TCP slave address, invalid gateway, invalid subnet mask, or the Ethernet network is not correctly configured.
communication	3403	0305	Modbus TCP slave connection refused. Modbus TCP slave rejected connection attempt. The Modbus TCP slave may not be listening for connections or there may be a firewall preventing the connection.
communication	3403	0306	
app	3406	0001	A web server login user was assigned to a group which did not exist. The system is unaffected, but that user will have limited (default) access.
app	3406	0002	The default login group for the web server was assigned to a group which did not exist. Access control has been disabled, because a minimal amount of access is required in order to log in. The configuration file should be fixed before continuing.
app	3406	0003	The web server configuration specified access control should be enabled, but did not specify at least one path to control access to. Access control has been disabled. The configuration file should be fixed before continuing.
app	3407	0002	The base directory for configuration files was missing and has been created automatically. The system has booted in a minimal configuration mode, and most functionality is limited. Please upload a new complete configuration file set.
app	3407	0003	A required default configuration file was missing. A minimal configuration for the corresponding component has been loaded, and some functionality may be limited.
app	3407	0004	A required default configuration file was incorrectly formatted. A minimal configuration for the corresponding component has been loaded, and some functionality may be disabled.
app	3407	0005	A configuration file specified by the user configuration file set was incorrectly formatted. The corresponding default configuration file is being used instead.
app	3407	0006	The file describing which configuration set to use was corrupted. The default configuration set is being used.
app	3407	0007	An error occurred while writing a config file. The file system may be full or damaged.
app	3407	0101	The configured RAM disk on the controller was unable to be created.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
AlarmID (UDINT)	AlarmID output from Y_ReadAlarm		
app	3407	0102	Detected an unsupported option card inserted in the controller.
app	3407	0104	Data in the controller SRAM did not match the expected value. It should be treated as corrupted until it is re-initialized.
app	3407	0106	The SRAM battery backup power failed. SRAM data should be treated as corrupted until it is re-initialized.
app	3407	0107	The controller's time-of-day clock detected a voltage decrease in the backup battery. The current time and date is likely to be incorrect. This alarm can be cleared, but will recur when the controller is powered ON until the time and day is reset and the battery is replaced.
app	3407	0204	Unable to set configured network default gateway
app	3409	0001	The servo network axis node for the axis specified in the configuration file was not found.
app	3409	0002	Axis enable failed. This problem is usually a result of communication problems with the servo drive. It may occur after a drive was disconnected from the network. In this case, use Y_ResetMechatrolink to establish communication with the drives once again.
app	3409	0003	Axis group motion activation failed. Some axes in the group are currently under control of another group, or motion has been blocked by the user.
app	3409	0004	The motion segment could not be added to the motion queue because it is already queued.
app	3409	0005	Moves are prohibited when any of the group's axes are disabled, have an alarm, or are in violation of their soft limits.
app	340A	0001	The source for the logical input was not found, the configured input will not be available.
app	340A	0002	The source for the logical output was not found, the configured output will not be available.
app	340A	0003	Two or more axis in the configuration file had the same axis ID.
app	340A	0004	The servo network axis node for the axis specified in the configuration file was not found.
app	340A	0005	The axis group specified in the configuration file could not be created because either one or more of its axes are invalid or the group name is already being used.
app	340A	0006	The type of AtTargetAgent specified in the configuration file is unknown. This is because AtTargetAgent could not be created.
app	340A	0007	The number of constraints for axis group soft limit must be the same as the number of axes in the axis group.
app	340A	0008	The axis group doesn't have the configured frame.
app	340A	000B	A continuous-wrap range for an axis causes its position to automatically wrap around between two user-specified numbers. Generally these numbers evaluate to full revolutions of the encoder but other ranges are permitted. However, all ranges specified in user units must map exactly to an integral number of encoder pulses. This alarm indicates that the mapping from user units to encoder ticks was inexact. Use more precise numbers to describe the range or choose a different range that evaluates to an integral number of encoder pulses. When this alarm occurs at startup or servo-net reset, it indicates that the axis has not been connected to an axis node and cannot be servoed on. Otherwise, this alarm indicates that the specified continuous-wrap range was not put into effect.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
app	340A	000D	Two or more logical outputs specified in the I/O configuration file use the same physical bit. This can cause writes to not correctly generate value-change events on logical outputs for the shared bits. The configuration file should be fixed.
app	340A	000E	One or more of the data parameters in the axis configuration file were out-of-range or otherwise incorrectly specified for the axis. The axis was not created and is not available.
app	340A	0010	After servo network reset, the Axis failed to reconnect to the servo network. The drive might have been removed from the network, the node ID of the drive might have changed or there might be a communication problem.
app	340A	0012	After servo network reset, the network I/O failed to reconnect to the servo network. The network I/O module might have been removed from the network, the node ID of the network I/O module might have changed or there might be a network communication problem.
app	340A	0013	After servo network reset, a new axis node was discovered. This axis node is not associated with any existing axes and will not be available. To make this node available, update the configuration and power cycle the controller.
app	340A	0014	After servo network reset, a new I/O node was discovered. This I/O node is not associated with any existing I/O and will not be available. To make this node available, update the configuration and power cycle the controller.
app	340A	0017	One or more of the axis data or configuration parameters were inconsistent or incompatible with the axis node specified. The axis was created but was not connected to the servo node.
app	340A	001B	Two or more LogicalInput have the same ID. The configuration file should be fixed.
app	340A	001C	Two or more LogicalOutput have the same ID. The configuration file should be fixed.
app	340A	001D	Two or more AnalogInput have the same ID. The configuration file should be fixed.
app	340A	001E	Two or more AnalogOutput have the same ID. The configuration file should be fixed.
app	340A	001F	Analog I/O configuration is missing the 'hardwareConfig' element, and configuration could not be resolved by the physical hardware. The configuration file should be fixed by adding this element to the analog I/O element.
app	340A	0020	One or more axes failed to respond to a servo-off command during a system I/O initiated abort. This is normally the result of communication problems with the drive, which also causes an automatic servo-off.
app	340A	0022	Reset of a servo node failed.
app	340A	0023	The axis position may not be valid because the persistent axis data was corrupted. SRAM should be reinitialized and the axis should be homed.
app	340C	0001	Time limit exceeded.
app	340C	0002	Distance limit exceeded.
app	340C	0003	Torque limit exceeded.
app	340C	0100	Modbus TCP I/O Driver Error on Server because of invalid address range
app	340C	0101	MBTCP Client I/O driver, MBTCP Connection config is missing input member

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
app	340C	0102	I/O memory area is not aligned to the correct byte to accommodate reading and writing.
app	340C	0103	Watchdog Error
app	340C	0104	Reserved
app	340C	0106	Reserved
app	340C	0107	Reserved
app	340C	0108	Reserved
app	340C	0109	Reserved
app	340C	010A	Not enough memory on PLC for POU during insertion. Project size must be reduced.
app	340C	010B	Internal PLC Error in memory management. This error can occur if an older project was loaded on the controller which was compiled to use less of the controllers total memory space. By using the "Resource" Dialog box, perform "Delete On target," for the bootproject, and then download the application code again.
app	340C	010C	Internal PLC Error: POU invalid
app	340C	010D	Internal PLC Error: Unknown POU type
app	340C	010E	Cannot insert a POU because there is no project.
app	340C	010F	Internal PLC Error: Cannot insert a POU because it does not belong to the project.
app	340C	0110	Internal PLC Error: Cannot insert a POU.
app	340C	0111	Internal PLC Error: Invalid POU type
app	340C	0112	Internal PLC Error: Memory reorganization not possible; PLC stopped.
app	340C	0113	Internal PLC Error: SPG defined more than once.
app	340C	0114	Internal PLC Error: Memory error for initialized data of POU.
app	340C	0115	Internal PLC Error: Retain CRC failed. Possible reasons: (1) actual project does not have any retain data, (2) actual project is 'old style' without retain CRC (3) PLC isn't in STOP mode
app	340C	0116	Internal PLC Error: FB defined more than once.
app	340C	0117	Internal PLC Error: Not all POU sent.
app	340C	0118	Internal PLC Error: No program memory defined.
app	340C	0119	Internal PLC Error: Invalid FB number.
app	340C	011A	Internal PLC Error: Invalid PG number.
app	340C	011B	Internal PLC Error: Invalid SPG number.
app	340C	011C	POU uses more than 80 percent of POU memory.
app	340C	011D	Project uses more than 80 percent of program memory.
app	340C	011E	Internal PLC Error: Invalid function or function block.
app	340C	011F	Internal PLC Error: Invalid firmware function or function block.
app	340C	0120	Internal PLC Error: Invalid program.
app	340C	0121	Internal PLC Error: Invalid change of mode.
app	340C	0122	Internal PLC Error: Unknown system mode! PLC stopped!
app	340C	0123	Stack overflow. Increase stack size.
app	340C	0124	System error in module. Check debugging output via controller's web interface.
app	340C	0125	System error in module. Check debugging output via controller's web interface.
app	340C	0126	Internal PLC Error: Error during indirect variable access.
app	340C	0127	PLC CPU overload.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
app	340C	0128	Internal PLC Error: Breakpoint unexpected.
app	340C	0129	Internal PLC Error: Error in data configuration.
app	340C	012A	Internal PLC Error: Error in retain data configuration.
app	340C	012B	Internal PLC Error: Floating point error.
app	340C	012C	Internal PLC Error: Fatal error.
app	340C	012D	Output string is too short.
app	340C	012E	Input string is too short.
app	340C	012F	Invalid input parameter 'p' or 'l' (position or length).
app	340C	0130	String is identical to the output string.
app	340C	0131	Invalid string comparison.
app	340C	0132	Invalid data type for string conversion.
app	340C	0133	Error in format string.
app	340C	0134	Error during string conversion.
app	340C	0135	Error in I/O configuration.
app	340C	0136	Initializing I/O driver failed.
app	340C	0137	Board not instantiated.
app	340C	0138	Board number not allowed.
app	340C	0139	Input Group doesn't fit.
app	340C	013A	Output Group doesn't fit.
app	340C	013B	Board not found.
app	340C	013C	Error reading inputs.
app	340C	013D	Error writing outputs.
app	340C	013E	Error creating I/O semaphore.
app	340C	013F	Invalid memory size.
app	340C	0140	Invalid I/O memory address.
app	340C	0141	Internal PLC Error: PG defined more than once.
app	340C	0142	POU exceeds 64K module size during insertion. POU size must be reduced.
app	340C	0143	Internal PLC Error: Error in task configuration.
app	340C	0143	Unknown I/O Driver.
app	340C	0200	Ethernet/IP I/O driver not initialized: change configuration to include Ethernet/IP driver.
app	340C	0201	Modbus/TCP I/O driver not initialized: change configuration to include Modbus/TCP driver.
app	340C	0201	PLC Controller initialization failed.
app	340C	0202	Modbus/TCP I/O driver ran out of resources. This can be caused by using too many poll blocks per server.
app	340C	0202	PLC Domain initialization failed.
app	340C	0203	PLC Communication server initialization failed.
app	340C	0204	PLC initialization failed.
app	340C	0205	PLC System Error.
app	340C	0206	Unspecified PLC Error. Could be divide by zero in application code?
app	340C	1028	The driver parameter specified in the axis configuration caused an exception
app	340C	1029	The driver parameter did not match the axis configuration
app	340C	1030	The configured axis count exceeded the allowable limit.
app	340C	1031	The axis count exceeded the allowable limit due to an auto-detected axis.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
app	340C	1033	Using an incompatible version of the PLCopenPlus firmware function block library may result in controller instability. Consequently, the PLC application will not be allowed to run. Please change either the controller's firmware or the firmware function block library.
app	340C	1110	All motion error codes are in the range from 0x1111 to 0x111f.
app	340C	1111	The move could not be buffered because the motion queue for that axis is full.
app	340C	1112	The move could not be started because motion is prohibited.
app	340C	1113	The servo drive failed to enable or disable.
app	340C	1114	Drive parameter read/write did not complete.
app	340C	1115	Drive parameter read/write failed
app	340C	1116	Torque move prohibited while non-torque moves queued or in progress.
app	340C	1117	Y_CamOut called while not camming.
app	340C	1118	The master slave relationship can not be modified because the master axis has not been set yet.
app	340C	1119	Y_CamFileSelect can not open a second cam table while the first cam table is still being opened.
app	340C	111A	The function block can not command an external axis.
app	340C	111B	The homing sequence is already in progress.
app	340C	111C	MC_SetPosition can not be called while the axis is moving.
app	340C	111D	Motion aborted due to axis alarm.
app	340C	111E	MC_SetPosition can not set the position to be outside the configured wrap range (Machine Cycle).
app	340C	111F	Can not transition to homing state; must be in StandStill state first.
app	340C	1120	Clear alarms is already in progress.
app	340C	1121	Axis reset is already in progress.
app	340C	1122	Mechatrolink reset is already in progress.
app	340C	1123	Y_CamStructSelect cannot transfer a second cam structure while the first cam structure is being transferred.
app	340C	1124	Y_ReadCamTable cannot be read a second cam structure while the first cam structure is being read.
app	340C	1125	Y_WriteCamTable cannot write a second cam structure while the first cam structure is being written.
app	340C	1126	MC_SetPosition cannot be called while either the master or slave axis is camming.
app	340C	1127	The function block can not be used with a virtual axis.
app	340C	1128	The function block can not be used with an inverter axis.
app	340C	1129	Y_VerifyParameters and Y_WriteParameters can not be called a second time while the first one is in progress.
app	340C	1210	All error codes for structures are in the range from 0x1211 to 0x121f.
app	340C	1211	Axis ID does not correspond to an axis.
app	340C	1212	The master slave relationship is not defined.
app	340C	1213	The input reference does not correspond to a real input
app	340C	1214	The output reference does not correspond to a real output.
app	340C	1215	The input/output number does not correspond to a real input or output bit.
app	340C	1216	Trigger reference is not valid.
app	340C	1217	The cam switch structure is not valid.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
app	340C	1218	The track structure is not valid.
app	340C	1219	Table size results in misaligned data.
app	340C	121A	Buffer size results in misaligned data.
app	340C	121B	Table type is not supported.
app	340C	121C	Invalid start index.
app	340C	121D	Invalid end index.
app	340C	1220	All error codes for invalid enumeration values are in the range from 0x1221 to 0x122f.
app	340C	1221	'BufferMode' does not correspond to a valid enumeration value.
app	340C	1222	'Direction' does not correspond to a valid enumeration value.
app	340C	1223	'StartMode' does not correspond to a valid enumeration value.
app	340C	1224	'ShiftMode' does not correspond to a valid enumeration value.
app	340C	1225	'OffsetMode' does not correspond to a valid enumeration value.
app	340C	1226	'Mode' does not correspond to a valid enumeration value.
app	340C	1227	'SynchMode' does not correspond to a valid enumeration value.
app	340C	1228	'Parameter' does not correspond to a valid enumeration value.
app	340C	1229	'AdjustMode' does not correspond to a valid enumeration value.
app	340C	122A	'RampIn' does not correspond to a valid enumeration value.
app	340C	122B	'ControlMode' does not correspond to a valid enumeration value.
app	340C	1230	All error codes for range errors are from 0x1221 to 0x122f.
app	340C	1231	Distance parameter is less than zero.
app	340C	1232	Velocity parameter is less than or equal to zero.
app	340C	1233	Acceleration is less than or equal to zero.
app	340C	1234	Deceleration is less than or equal to zero.
app	340C	1235	Torque is less than or equal to zero.
app	340C	1236	Time is less than or equal to zero
app	340C	1237	Specified time was less than zero.
app	340C	1238	Specified scale was less than or equal to zero.
app	340C	1239	Velocity is negative.
app	340C	123A	Denominator is zero.
app	340C	123B	Jerk is less than or equal to zero.
app	340C	123C	TorqueRamp is less than or equal to zero.
app	340C	123D	Engage position is outside the table domain.
app	340C	123E	Negative engage width.
app	340C	123F	Disengage position is outside the table domain.
app	340C	1240	Negative disengage width.
app	340C	1241	StartPosition is outside of master's range.
app	340C	1242	EndPosition is outside of master's range.
app	340C	1310	All error codes for invalid input data range from 0x1211 to 0x121f.
app	340C	1311	The specified Pn does not exist.
app	340C	1312	The mask does not correspond to valid tracks.
app	340C	1313	The profile must start with relative time equal to zero, and the time must be increasing.
app	340C	1314	The specified cam file does not exist.
app	340C	1315	Invalid header for the cam file. Cam tables must have a header indicating the number of rows, number of columns and a feed forward velocity flag

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
AlarmID (UDINT)	AlarmID output from Y_ReadAlarm		
app	340C	1316	The first (master) column must be either increasing or decreasing.
app	340C	1317	Cam table reference does not refer to a valid cam table.
app	340C	1318	The engage phase exceeded the time limit. Slave axis could not attain the target position and velocity within the user specified time limit.
app	340C	1319	The engage phase exceeded the distance limit. Slave axis could not attain the target position and velocity within the user specified master distance.
app	340C	131A	Invalid width input. Width is an enumeration type with the following allowable values 'WIDTH_8'=0, 'WIDTH_16'=1, and 'WIDTH_32'=2.
app	340C	131B	The slave axis can not be the same as the master axis.
app	340C	131C	Default drive parameter info is not available for this parameter.
app	340C	131D	Invalid external axis.
app	340C	131E	Invalid virtual axis.
app	340C	131F	File extension is not recognized or missing.
app	340C	1320	Could not find the axis parameter file.
app	340C	2110	All log error codes are in the range from 0x2111 to 0x211f.
app	340C	2111	Adding log items or setting up log is not possible because the data log is already set up.
app	340C	2112	Starting or stopping logging is not possible because the data log is not set up.
app	340C	2113	Invalid handle for user log item.
app	340C	2114	Data log can not be created because too many data logs are in use.
app	340C	2115	Invalid handle for data log.
app	340C	2116	A user log item can only support eight inputs for each type.
app	340C	2117	Saving the log failed.
app	340C	2300	Invalid group handle
app	340C	2301	An axis is already owned by another group
app	340C	2302	Group activation is blocked
app	340C	2303	Invalid coordinate system
app	340C	2304	Move prohibited because group has an alarm
app	340C	2305	Group activation prohibited, invalid axis/joint config
app	340C	2306	Group activation prohibited, mismatched axis command position
app	340C	2307	The group reports one or more of its axes has an error.
app	340C	2308	Axis group reset is already in progress.
app	340C	2309	Invalid circular path method
app	340C	230A	Invalid circular path direction
app	340C	230B	Invalid circle geometry
app	340C	230C	Grouped axis is disabled.
app	340C	230D	Invalid transition mode.
app	340C	230E	Invalid transition parameter.
app	340C	230F	Invalid transition geometry. The values for the acceleration, deceleration, and/or velocity of the transition yield an invalid geometry. Can't create the corner geometry to meet the specification.
app	340C	B114	Failed to send clear alarms command.
app	340C	B115	Failed to reset Mechatrolink.
app	340C	B116	Mechatrolink reset is prohibited while axes are moving.
app	340C	B117	Failed to initialize absolute encoder.
app	340C	E110	All error codes for ProConOS errors range from 0xE111 to 0xE11f.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
app	340C	E111	Instance object is NULL.
app	340C	E112	The instance data is NULL.
app	340C	E113	The structure pointer check sum is invalid.
app	340C	E114	The structure size does not match.
app	340C	EDED	This function block was implemented in a later firmware version. If you would like to use this function block, then the controller must be updated.
app	340C	F110	All error codes for kernel errors range from 0xF111 to 0xF11f.
app	340C	F111	An internal assertion in the motion kernel failed indicating the controller is not in a stable state. This error should be reported to Yaskawa Electric America.
app	340D	????	User generated alarm via the Y_PostUserAlarm FB in IEC6131 code.
user	3501	0000	A user script task posted an alarm directly.
motionKernel	4202	0001	The command position will soon reach the allowable range for the axis in the positive direction (positive overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	4202	0002	The command position will soon reach the allowable range for the axis in the negative direction (negative overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	4202	0003	The command speed will soon reach the allowable range for the axis in the positive direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0004	The command speed will soon reach the allowable range for the axis in the negative direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0005	The command acceleration will soon reach the allowable range for the axis in the positive direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0006	The command acceleration will soon reach the allowable range for the axis in the negative direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0007	The command torque will soon reach the allowable range for the axis in the positive direction (overtorque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0008	The command torque will soon reach the allowable range for the axis in the negative direction (overtorque). The axis may not be moved again until the alarm condition is cleared.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
motionKernel	4202	0011	The command position will soon reach the allowable range for the axis in the positive direction (positive overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	4202	0012	The command position will soon reach the allowable range for the axis in the negative direction (negative overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	4202	0013	The command speed will soon reach the allowable range for the axis in the positive direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0014	The command speed will soon reach the allowable range for the axis in the negative direction (overspeed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0015	The command acceleration will soon reach the allowable range for the axis in the positive direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0016	The command acceleration will soon reach the allowable range for the axis in the negative direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0017	The command torque will soon reach the allowable range for the axis in the positive direction (over torque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0018	The command torque will soon reach the allowable range for the axis in the negative direction (over torque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0021	The command position will soon reach the allowable range for the axis in the positive direction (positive overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	4202	0022	The command position will soon reach the allowable range for the axis in the negative direction (negative overtravel). The axis may not be moved again until the alarm condition is cleared. After the alarm is cleared, it is permissible to execute a move which brings the axis back toward the allowed region, even though the axis is probably still outside the allowed region. Any move which pulls the axis further away from the allowed region will re-trigger the alarm.
motionKernel	4202	0023	The command speed will soon reach the allowable range for the axis in the positive direction (over speed). The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0024	The command speed will soon reach the allowable range for the axis in the negative direction (over speed). The axis may not be moved again until the alarm condition is cleared.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
motionKernel	4202	0025	The command acceleration will soon reach the allowable range for the axis in the positive direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0026	The command acceleration will soon reach the allowable range for the axis in the negative direction. The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0027	The command torque will soon reach the allowable range for the axis in the positive direction (over torque). The axis may not be moved again until the alarm condition is cleared.
motionKernel	4202	0028	The command torque will soon reach the allowable range for the axis in the negative direction (over torque). The axis may not be moved again until the alarm condition is cleared.
Mechatrolink	4301	000A	The SERVOPACK model type was unable to be determined. This can indicate that some parameters may be incorrect.
Mechatrolink	4301	000C	The controller was unable to send the drive command because servo network resources were allocated to motion. Brake on, brake off, absolute encoder initialization and alarm clear can only be sent when not moving.
Mechatrolink	4301	001C	The Mechatrolink.xml file specified duplicate configuration structures for a node. The first match was used, subsequent matches were ignored.
Mechatrolink	4301	001D	The Mechatrolink.xml file specified duplicate default configuration structures for a node type. The first default structure was used, subsequent structures were ignored.
Mechatrolink	4301	001E	A node was detected on the Mechatrolink network, but it is not supported by the software.
Mechatrolink	4301	001F	The Mechatrolink comm board inverter control reference/run control is not enabled. Change the settings in parameters b1-01 and b1-02 to '3' to select PCB reference/run source.
Mechatrolink	4301	0020	The drive returned an invalid watch dog code indicating a possible dropped communication packet.
Mechatrolink	4302	0000	The base code for Sigma-II drive warnings. The drive's warning value is bitwise OR'd in with this base value.
Mechatrolink	4302	0091	This warning occurs before the overload alarms (A.710 or A.720) occur. If the warning is ignored and operation continues, an overload alarm may occur.
Mechatrolink	4302	0092	This warning occurs before the regenerative overload alarm (A.32) occurs. If the warning is ignored and operation continues, a regenerative overload alarm may occur.
Mechatrolink	4302	0093	This warning occurs when the absolute encoder battery voltage is lowered. Continuing the operation in this status may cause an alarm.
Mechatrolink	4302	0094	A value outside the setting range was set using MECHATROLINK-II communications.
Mechatrolink	4302	0095	A command not supported in the product specifications was sent, OR the command reception conditions were not met.
Mechatrolink	4302	0096	A communications error occurred (once).
Mechatrolink	4303	0000	The base code for Sigma-III drive warnings. The drive's warning value is bitwise OR'd in with this base value.
Mechatrolink	4303	0900	Position error pulse exceeded the parameter settings (Pn520 x Pn51E/100).

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
Mechatrolink	4303	0901	When the servo turned ON, the position error pulses exceeded the parameter setting (Pn526 x Pn528/100).
Mechatrolink	4303	0910	This warning occurs before the overload alarms (A.710 or A.720) occur. If the warning is ignored and operation continues, an overload alarm may occur.
Mechatrolink	4303	0911	Abnormal vibration at the motor speed was detected. The detection level is the same as A.520. Set whether to output an alarm or warning by "Vibration Detection Switch" of Pn310.
Mechatrolink	4303	0920	This warning occurs before the regenerative overload alarm (A.320) occurs. If the warning is ignored and operation continues, a regenerative overload alarm may occur.
Mechatrolink	4303	0930	This warning occurs when the absolute encoder battery voltage is lowered. Continuing the operation in this status may cause an alarm.
Mechatrolink	4303	0941	The change of the parameters can be validated only after turning the power ON from OFF.
Mechatrolink	4303	094A	Incorrect command parameter number was set.
Mechatrolink	4303	094B	Command input data is out of range.
Mechatrolink	4303	094C	Calculation error was detected.
Mechatrolink	4303	094D	Data size does not match.
Mechatrolink	4303	095A	Command was sent though command sending condition was not satisfied.
Mechatrolink	4303	095B	Unsupported command was sent.
Mechatrolink	4303	095C	Command condition is not satisfied for parameter settings.
Mechatrolink	4303	095D	Command, especially latch command, interferes.
Mechatrolink	4303	095E	Subcommand and main command interfere.
Mechatrolink	4303	0960	Communications error occurred during MECHATROLINK communications.
Mechatrolink	4304	0000	The base code for io warnings. The io's warning value is bitwise OR'd in with this base value.
DPRAM	4309	1000	Error code prefix for data link errors
DPRAM	4309	1011	Invalid register
DPRAM	4309	1012	Value exceeded data limit
DPRAM	4309	1013	Data math error
DPRAM	4309	1014	Register number and data size do not agree
DPRAM	4309	1015	Invalid data size
DPRAM	4309	1030	Servo and option card accessed data link channel at the same time
DPRAM	4309	10FF	Unknown data link error
DPRAM	4309	2000	Error code prefix for message errors
DPRAM	4309	2002	Invalid register
DPRAM	4309	2003	Message size and data quantity do no match
DPRAM	4309	2030	Invalid register
DPRAM	4309	2031	Register access not allowed
DPRAM	4309	2032	Setting value is out of range
DPRAM	4309	2033	Messaging accessed only part of a register group or spanned register groups
DPRAM	4309	2034	Message command could not be processed because pre-conditions have not been met
DPRAM	4309	2035	Command processing is not possible due to conflict
DPRAM	4309	20A1	Controller option card received an empty message response

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
		GroupErrorID (UINT)	
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
Mechatrolink	4312	0000	The base code for inverter warnings. The inverters warning value is bit-wise OR'd in with this base value.
app	4401	0008	Each call to groupAxes() must be matched by a corresponding call to ungroupAxes(). If a script exits without such a matching call (thus leaving an 'orphaned' group behind), this warning is issued. Clearing the warning also ungroups the orphaned group.
app	4401	0009	The debug stack trace was longer than expected. It may be clipped.
app	4403	0001	The event queue for the remote client was full, and an event was dropped. This is generally caused either by exceeding the network bandwidth or exceeding the general system processing power (starving the connection). When an event is dropped in this manner, the connection is terminated.
app	4403	0005	An RMI connection was attempted by an external client and rejected due to the concurrent connection limit.
app	4407	0001	The configuration file directory is read-only or resides on a read-only file system. Attempts to update the configuration or create directories will fail.
app	4407	0002	An attempt was made to write to a read-only configuration file. The write failed.
app	4407	0105	There was an indication that the SRAM battery backup power may have failed temporarily. SRAM data may have been compromised.
app	4408	0001	The alarm history was configured to use NVRAM storage, but either the available NVRAM was not sufficient to contain the configured buffer size, or the configured buffer size was not large enough to contain the configured number of records. The alarm history will contain fewer records than configured.
app	4408	0002	The alarm history was configured to use NVRAM storage and the data was found to be corrupted. The alarm history has been lost. NOTE: this alarm also occurs if the configured size of the alarm history has been changed.
app	440A	000C	The position and torque scales specified in the configuration file have different signs. As a result, a positive acceleration results in a negative torque, and position limits are opposite in sign as the torque limits.
app	440A	000F	The axis was temporarily disconnected from the servo network during reset. During this time, the feedback data is not valid and the axis cannot be moved.
app	440A	0011	The network I/O was temporarily disconnected from the servo network during reset. During this time, any network I/O state change will be unobservable to the controller.
app	440A	0019	The system was rebooted by the user.
app	440A	001A	The system failed to shut down gracefully during a reboot, although the reboot did occur. This does not necessarily indicate that the software is damaged.
app	440B	0001	The controller is running out of memory. Memory should be freed as soon as possible. Try closing connections to the controller or stopping scripts.
app	440B	0003	The largest free memory block is approaching the critical level. Memory should be freed as soon as possible. Try closing connections to the controller or stopping scripts.
Axes Group	440C	0103	Unable to add AxesGroup to groupIODriver. Check the validity of the AXES_GROUP_REF.Handle.

	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
app	440C	0105	Reserved
app	440C	0207	If the minor version on the controller is less than the one in the IDE, then some function blocks will not be supported. However, since the major version matches, those that are supported have identical interfaces.
app	440C	1032	The configuration file version is not compatible with the firmware version. Please use the configuration tool to update the configuration files to match the firmware version.
app	440C	1034	Some function blocks are not supported by the controller firmware. If these function blocks are used in the PLC application, then their ErrorID will always equal 60909. If these function blocks are needed, then please upgrade the controller's firmware.
app	4501	0000	A user script task posted a warning directly.