Toolbox Introduction

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

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.
- Gantry Toolbox provides functions useful for operating an XY table with or without a Z (vertical) axis.
- Group Toolbox is the successor to Gantry Toolbox and provides enhancements to PLCopen Part 4 for interpolation, including G Code support.
- 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.
- Pendant Toolbox makes it easy to add manual mode and position teaching support for any application involving groups such as robots. It is specifically designed to interface via Modbus TCP to Yaskawa's teach Pendant, but other pendants or HMI's can also take advantage of this Toolbox.
- PLCopen Toolbox contains functions that build upon the PLCopen standard functions. It can serve as a starting point for every project.
- 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.

Cam Toolbox

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

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 on cam applications like random rotary knife, linear flying shear, labeler, bottle filler etc.

Requirements for v302

To use the Cam Toolbox, your 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

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 your 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.

Current Version:

(*****************************   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.

Previous Versions:

(*****************************   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 occuring.

12) CamControl, CamShift_Control - Added support for Multi Use Latches, which is a new feature of the PLCopen Toolbox v206 ProductBuffer function block.

(*************** 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 >= 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

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

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 standstill. (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 descried 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.	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..20] 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.	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..200] OF [[[Undefined variable Primary.DataType_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**		
U	**CamParameters**	**CamSegmentArray**		
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 auto-	MyCamSegmentStruct.SplineStartSlope

			matically determined by the CamGenerator.	
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
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=r-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 stop synchronization (Reference prm 1502 - CamMasterShiftedCyclic)	MyCamStruct.DisengagePosition
U	DisengageData	Y_DISENGAGE_DATA		MyCamStruct.
	Window	LREAL	Size of the window in master units where the engage or disengage will take place	MyCamStruct.Window
U	MasterCycle	LREAL		MyCamStruct.MasterCycle
U	SlaveCycle	LREAL		MyCamStruct.SlaveCycle

Data Type: CamSyncStruct

For use with the CamControl and CamShift_Control 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 func-	MyCamSyncStruct.Shifting

			tion block.	
C	Pause	BOOL	Status flag set by the CamControl function block to signal theCamShift_Control function block.	MyCamSyncStruct.Pause

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 be either the user or a function block depending on whether this datatype is used with Y_ReadCamTable or Y_WriteCamTable in the PLCopen Plus firmware library.

Code Example

Enumerated Types for Cam Toolbox

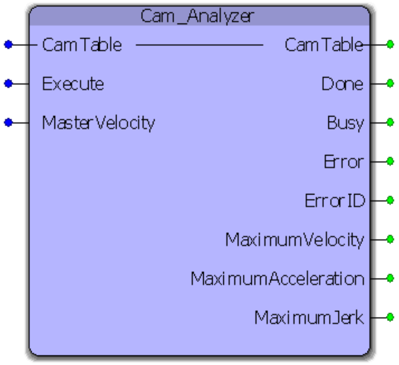
Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to a zero-based integer (INT) list.

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	

TB_CurveType	Indicates the Cam formula to be applied between to positions.		
	0	n/a	Not a valid CurveType
	1	StraightLine	
	2	Parabolic	
	3	SimpleHarmonic	
	4	Cycloidal	
	5	ModifiedTrapezoid	
	6	ModifiedSine	
	7	ModifiedConstVelocity	
	8	AsymmetricalCycloidal	
	9	AsymmetricalModifiedTrapezoid	
	10	Trapecloid	
	11	OneDwellCycloidal_1	
	12	OneDwellCycloidal_2_3	
	13	OneDwellTrapezoid_1	
	14	OneDwellTrapezoid	
	15	OneDwellTrapezoid_2_3	
	16	OneDwellModifiedSine	
	17	OneDwellTrapecloid	
	18	NoDwellSimpleHarmonic	
	19	NoDwellModifiedTrapezoid	
	20	NoDwellModifiedConstVelocity	
	21	NC2Curve	
	22	TangentMatching	
	23	ReverseTrapecloid	
	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
B	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.	

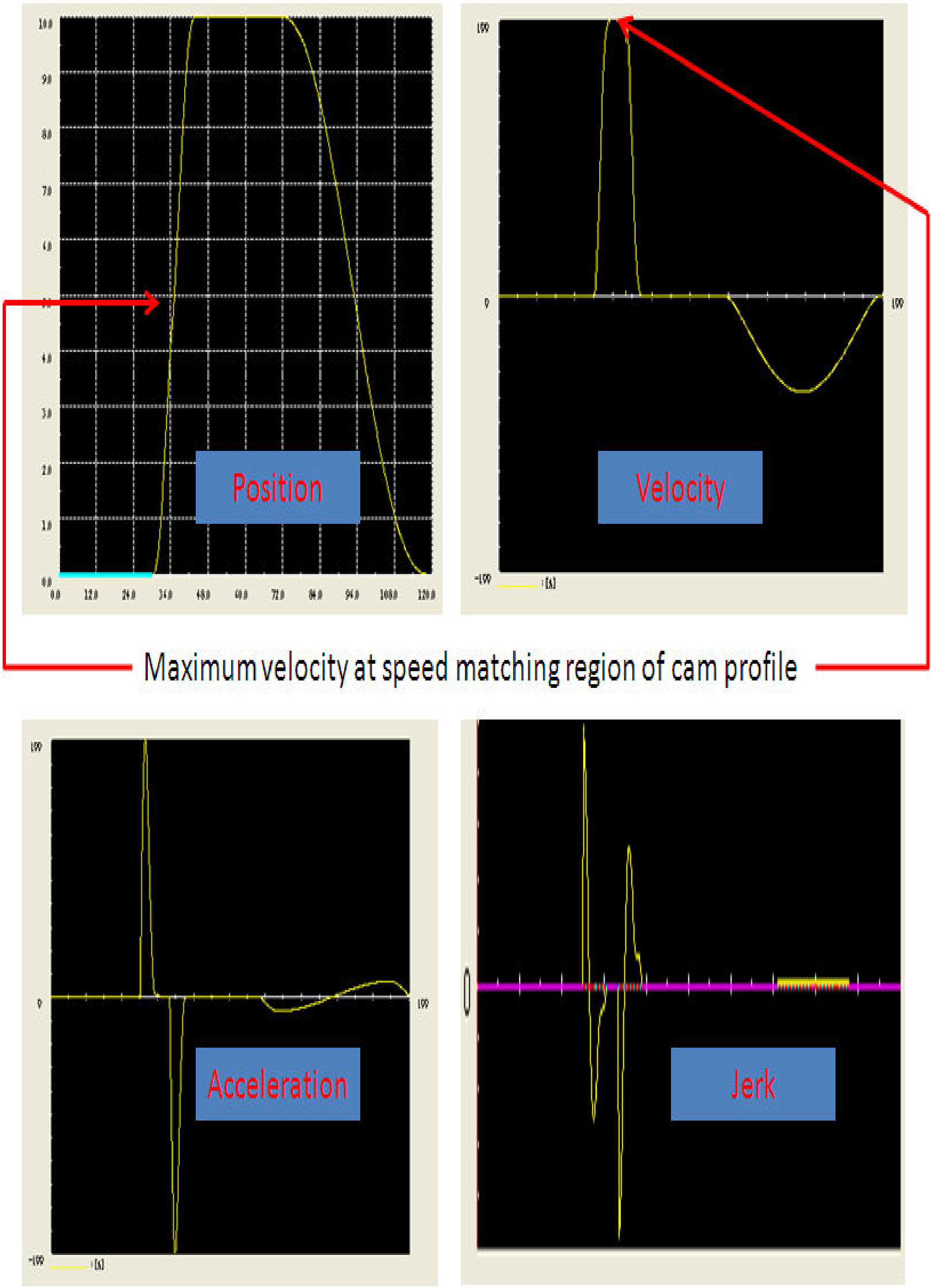
			This output is cleared when 'Execute' or 'Enable' goes low.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
B	MaximumVelocity	LREAL	Peak slave velocity for the given cam profile at the maximum master velocity.
B	MaximumAcceleration	LREAL	Peak slave acceleration for the given cam profile at the maximum master velocity.
B	MaximumJerk	LREAL	Peak slave jerk for the given cam profile at the maximum master velocity.

Error Description

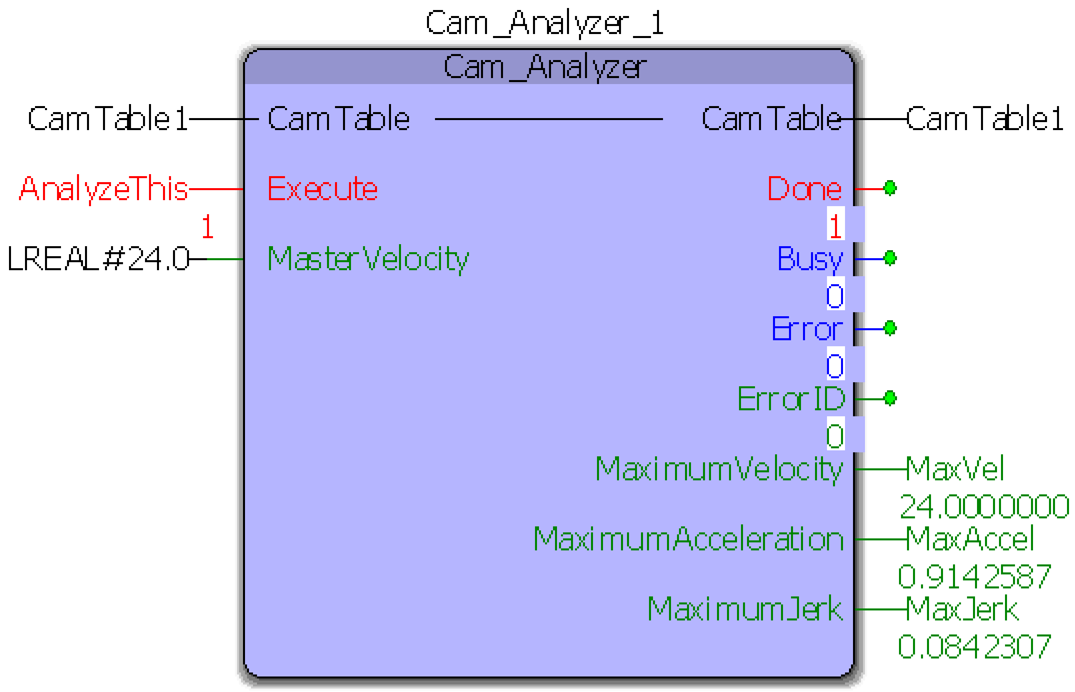
ErrorID	Meaning
0	No error.
10113	Incorrect cam table size (check the CamTable.Header.Datasize).

Example

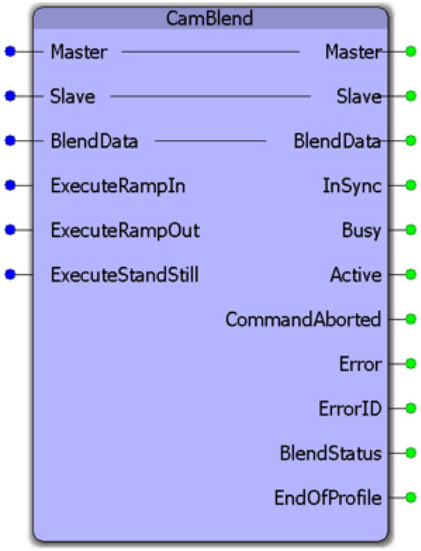
Consider a linear flying shear application. The maximum slave velocity of the profile is in the speed matching region. The master maximum velocity was given as 24 units/sec and the maximum velocity output of the CamAnalyzer is 24.



Maximum velocity at speed matching region of cam profile


Cam_Analyzer_1
Cam_Analyzer
CamTable1
CamTable
CamTable
CamTable1
AnalyzeThis
Execute
Done
1
1
LREAL#24.0
MasterVelocity
Busy
0
Error
0
ErrorID
0
MaximumVelocity
MaxVel
24.0000000
MaximumAcceleration
MaxAccel
0.9142587
MaximumJerk
MaxJerk
0.0842307

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.	
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	FALSE

			(calculated after an E-Stop recovery routine) in the BlendData structure	
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.	
B	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 operating, 2 indicates the Running cam is operating, and 3 indicates the RampOut cam is operating.	
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 that is 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 motion performance by allowing as much time as possible for the slave to accelerate up to the master speed.
- 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.

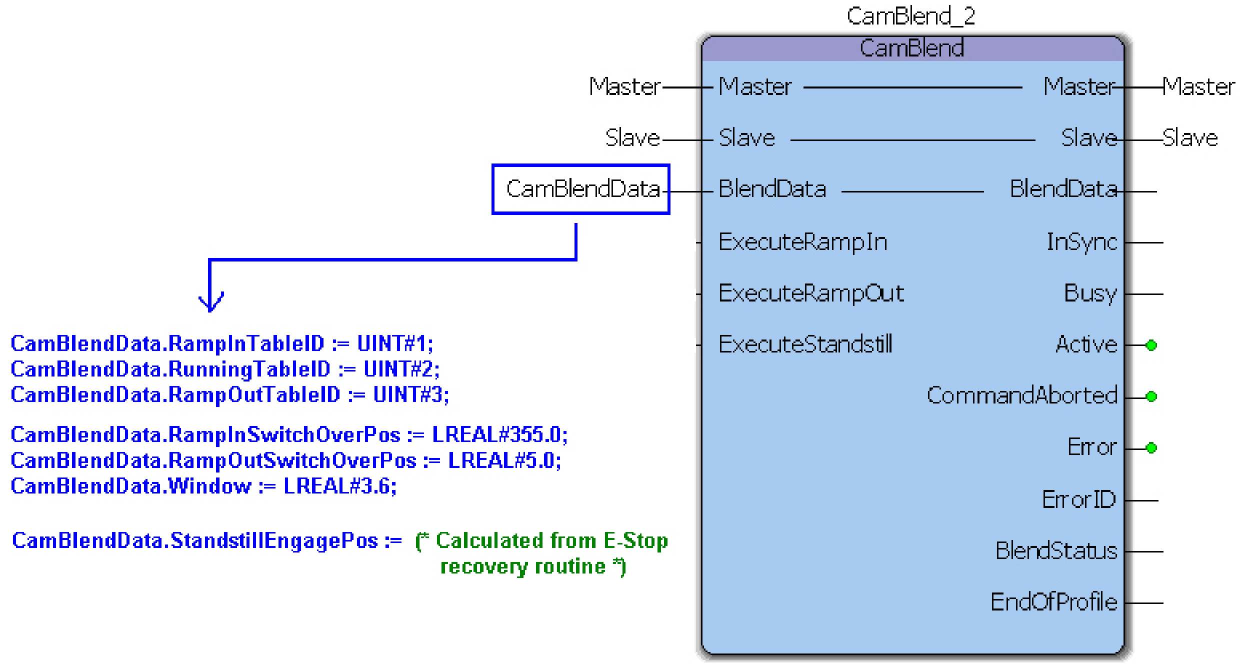
See the CamBlend eLearning Module on Yaskawa's YouTube Channel.

Error Description

ErrorID	Meaning
0	No error.
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.
4375	CamOut called while not camming.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.)
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 POUs.
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.
4643	Start mode does not correspond to a valid enumeration value.
4669	Engage position is outside the cam table domain.
4670	Engage window is less than zero.
467/1	Disengage position is outside the cam table domain.

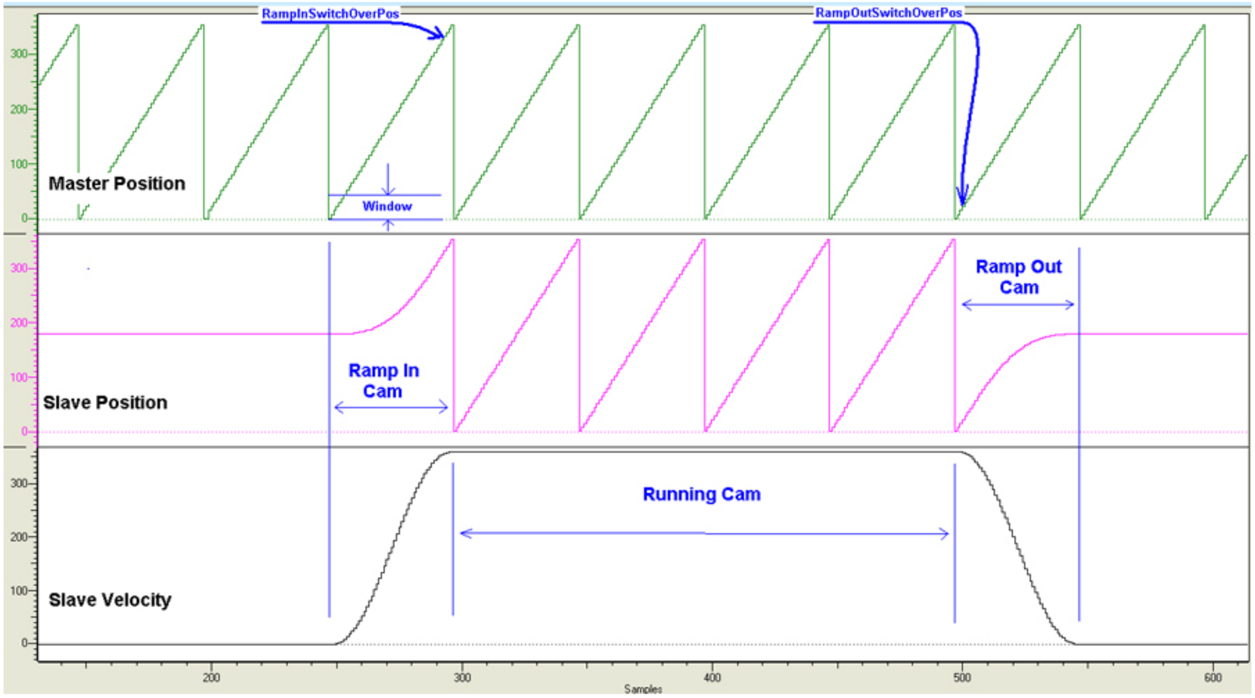
4672	Negative Disengage Window.
4887	CamTableID does not refer to a valid cam table.
4891	The slave axis can not be the same as the master axis.
10084	One of the Cam Tables has an invalid TableID.
57620	The DataType connected to a function block parameter specified as ANY type does not match the required 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. This error may occur because data passed to an 'Axis' input on a PLCopen function block is not an AXIS_REF. If you have included a data element into a user structure which includes an AXIS_REF, be sure that the input to the function block is entered correctly.

Example 1

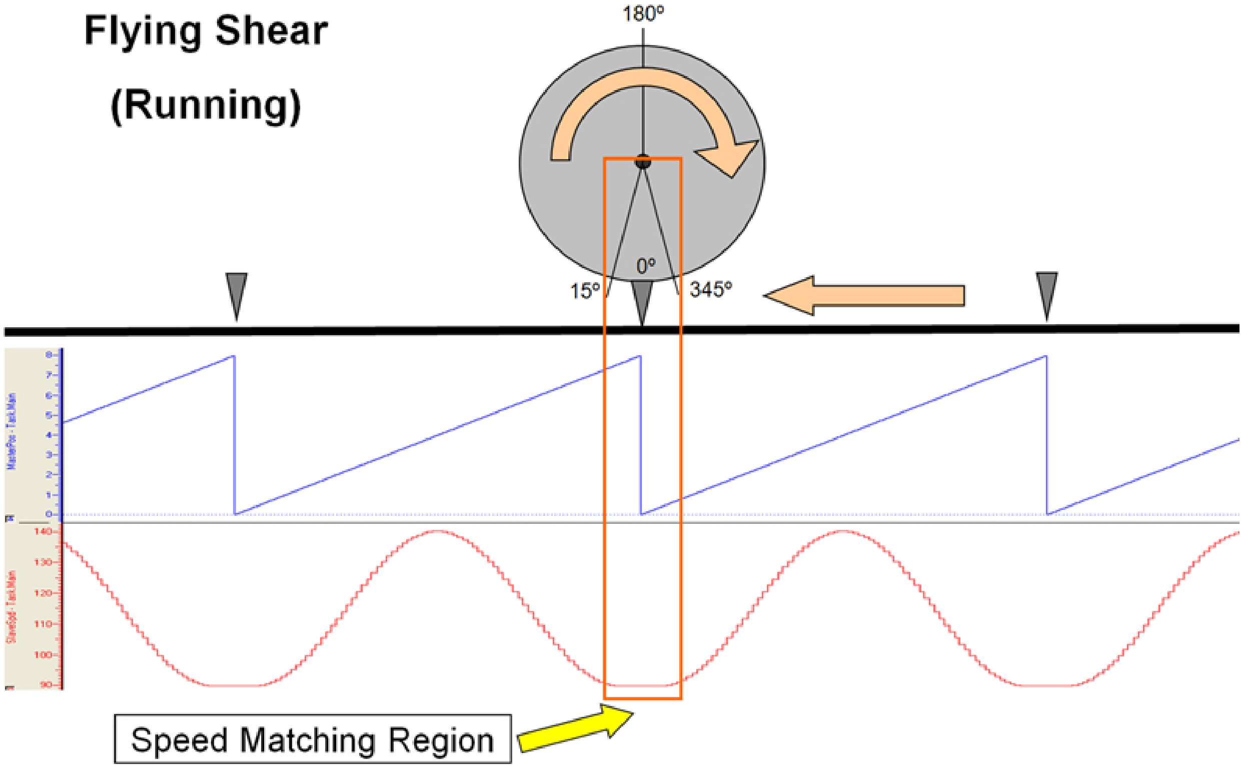


```
CamBlendData.RampInTableID := UINT#1;
CamBlendData.RunningTableID := UINT#2;
CamBlendData.RampOutTableID := UINT#3;

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

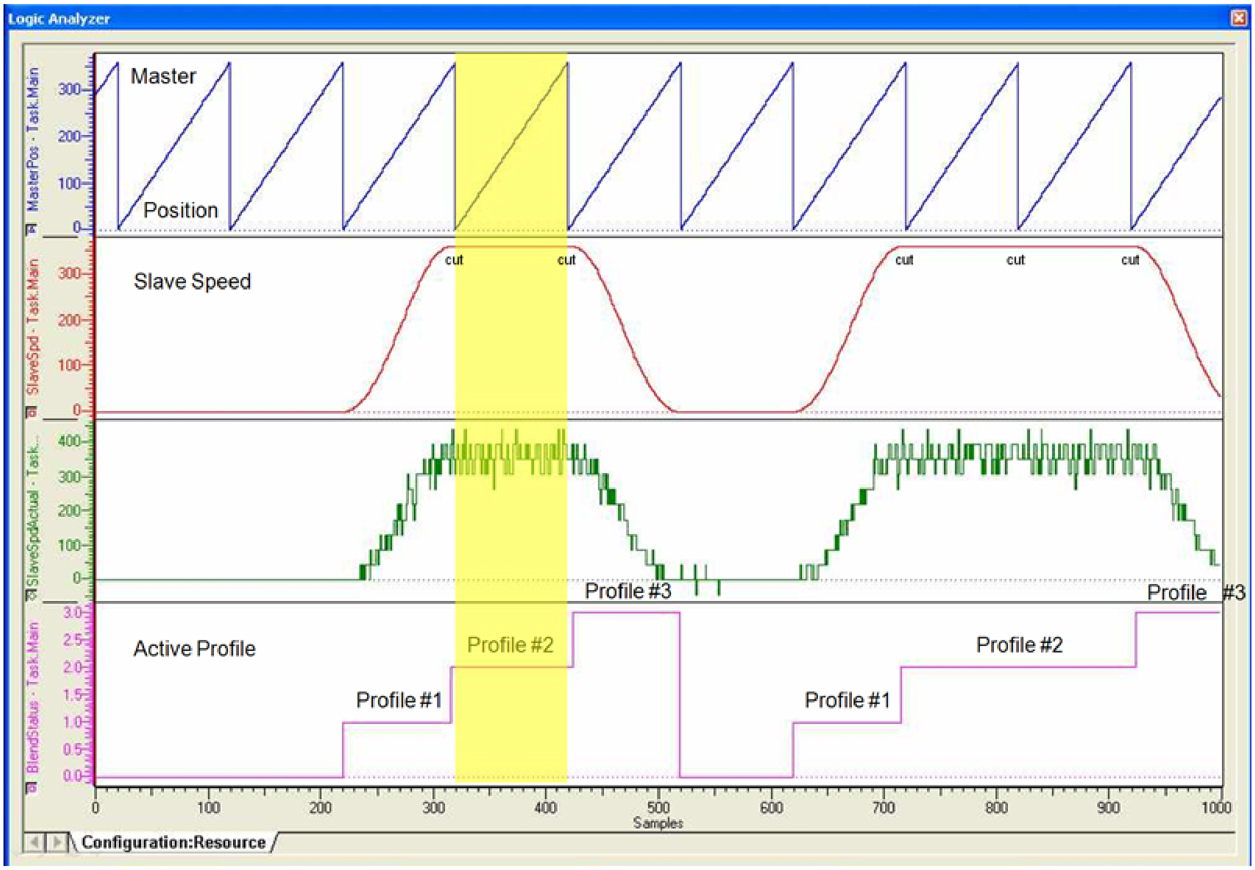


Application Example

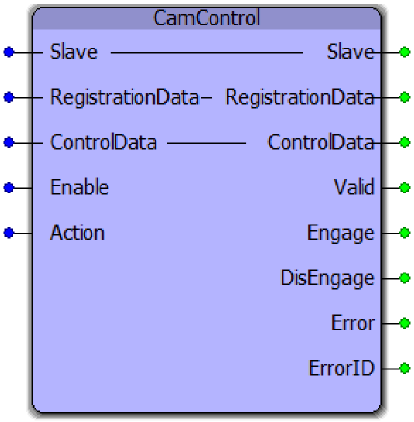


Timing Diagram

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 uleft 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.

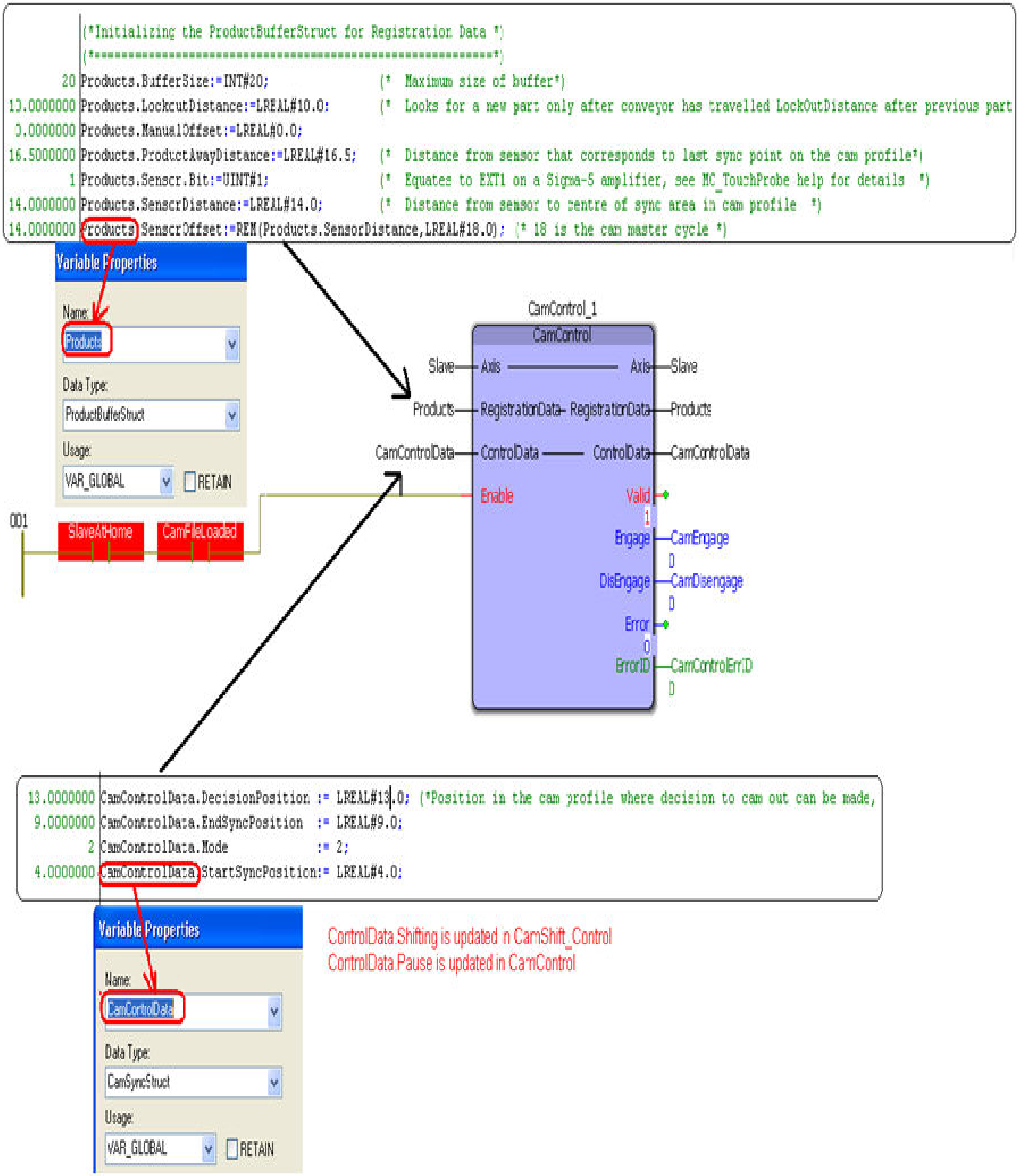
• This function block is designed to work with the CamShift_Control function block. It waits for an initial Camshift will occur before the first Engage event should take place. If the application requires the slave to become synchronized with the master without a Camshift, simply use an R_TRIG of the CamControl.Valid to cause the CamData.Shifting bit to go high and low.

Error Description

ErrorID	Meaning
0	No error.
10081	ControlData.DecisionPosition is <= 0. The position to determine when to disengage the cam cannot be less than or equal to zero.

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;
```

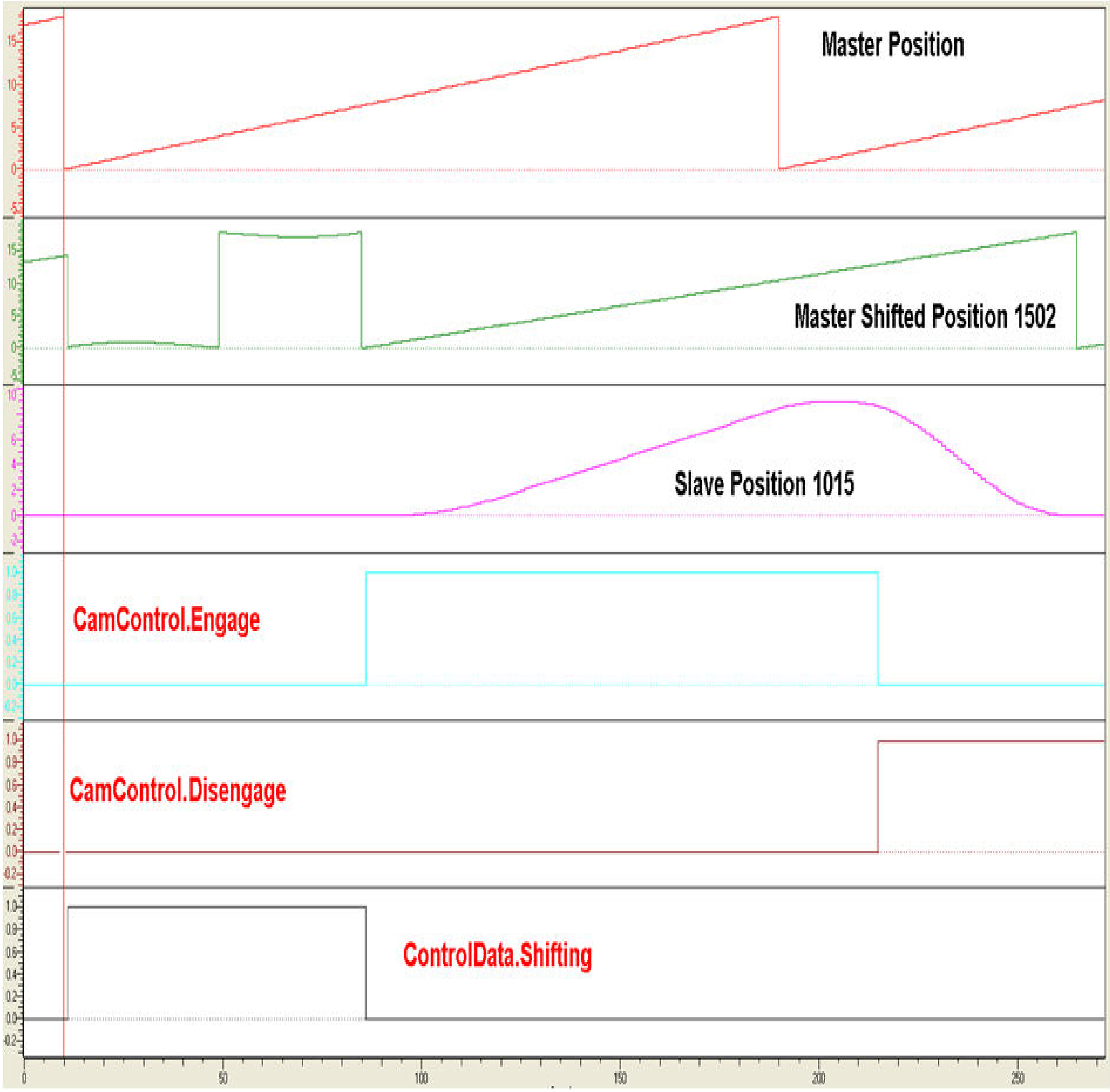
Variable Properties

Name: CamControlData

Data Type: CamSyncStruct

Usage: VAR_GLOBAL □ RETAIN

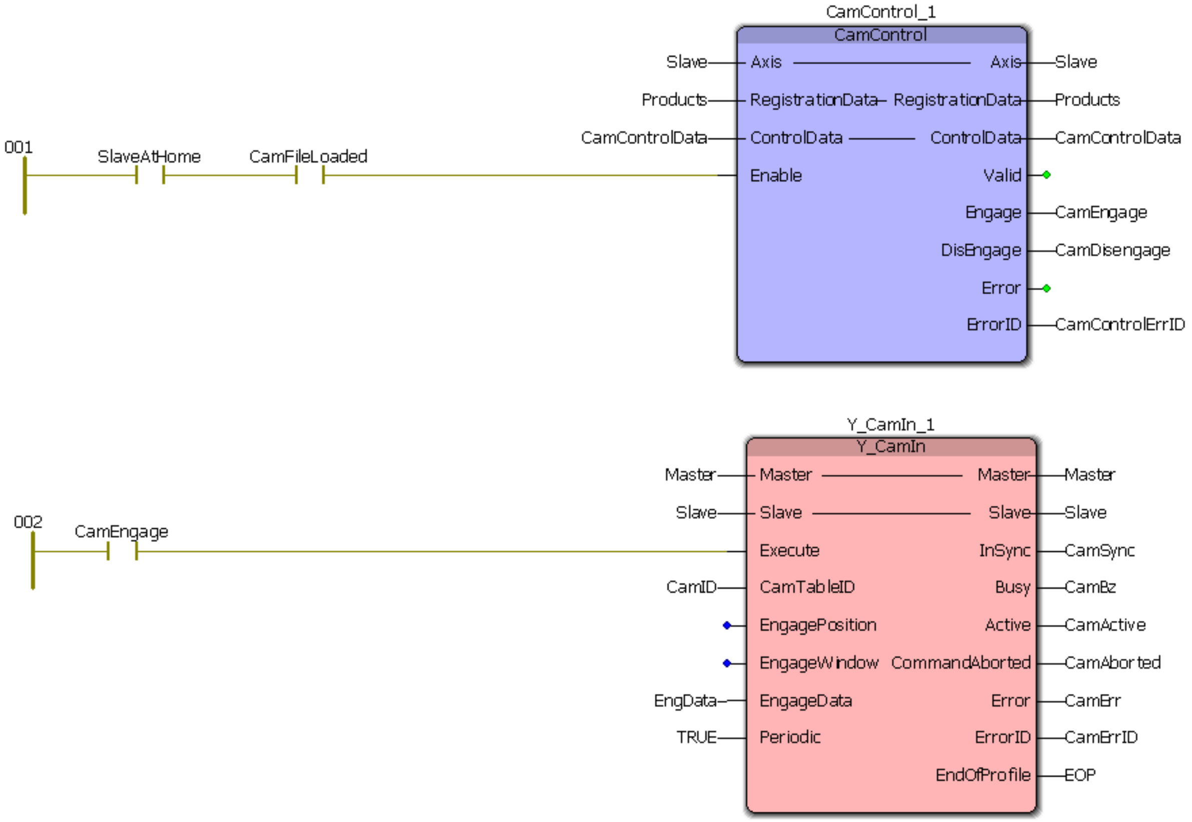
ControlData.Shifting is updated in CamShift_Control
ControlData.Pause is updated in CamControl

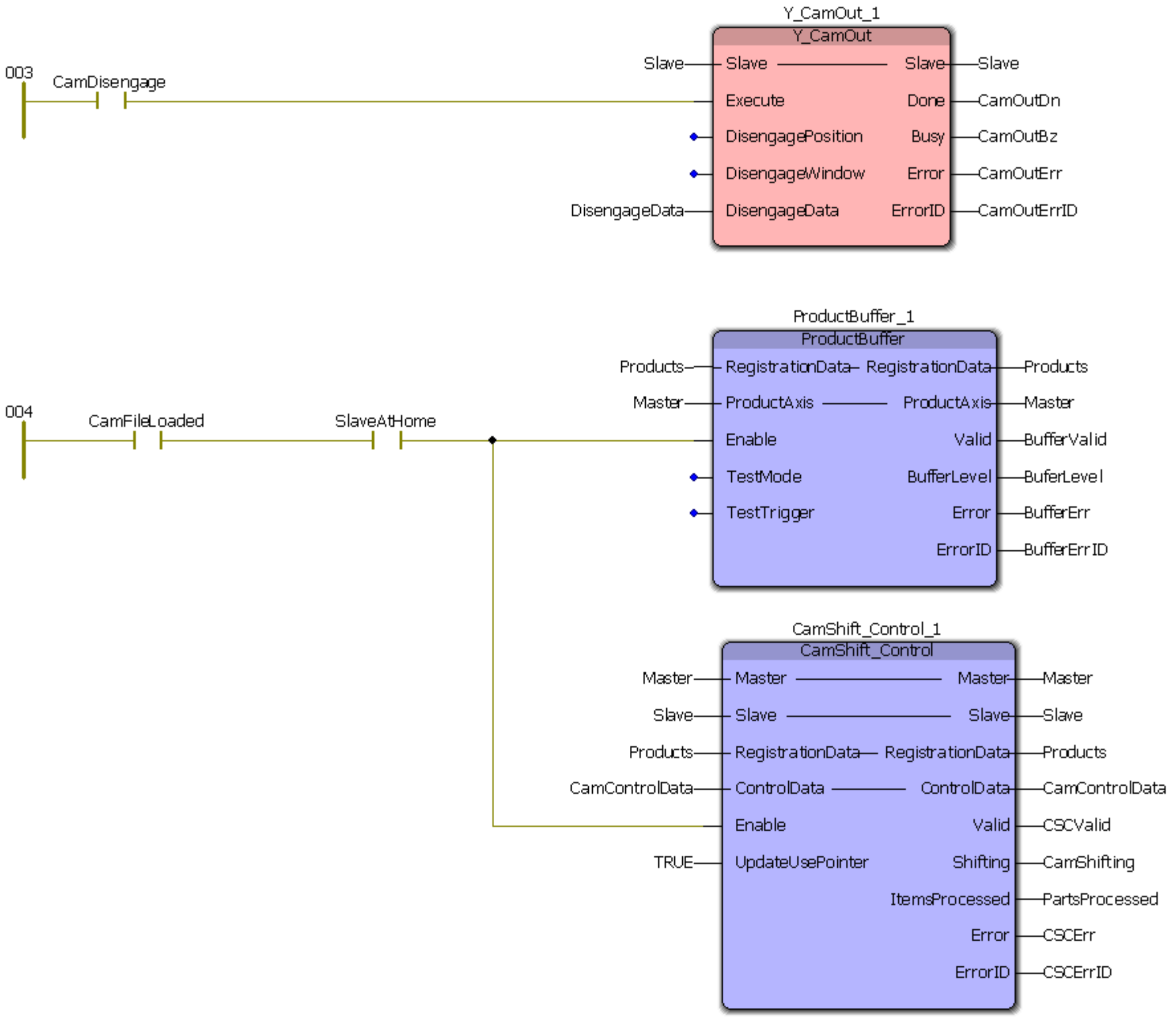


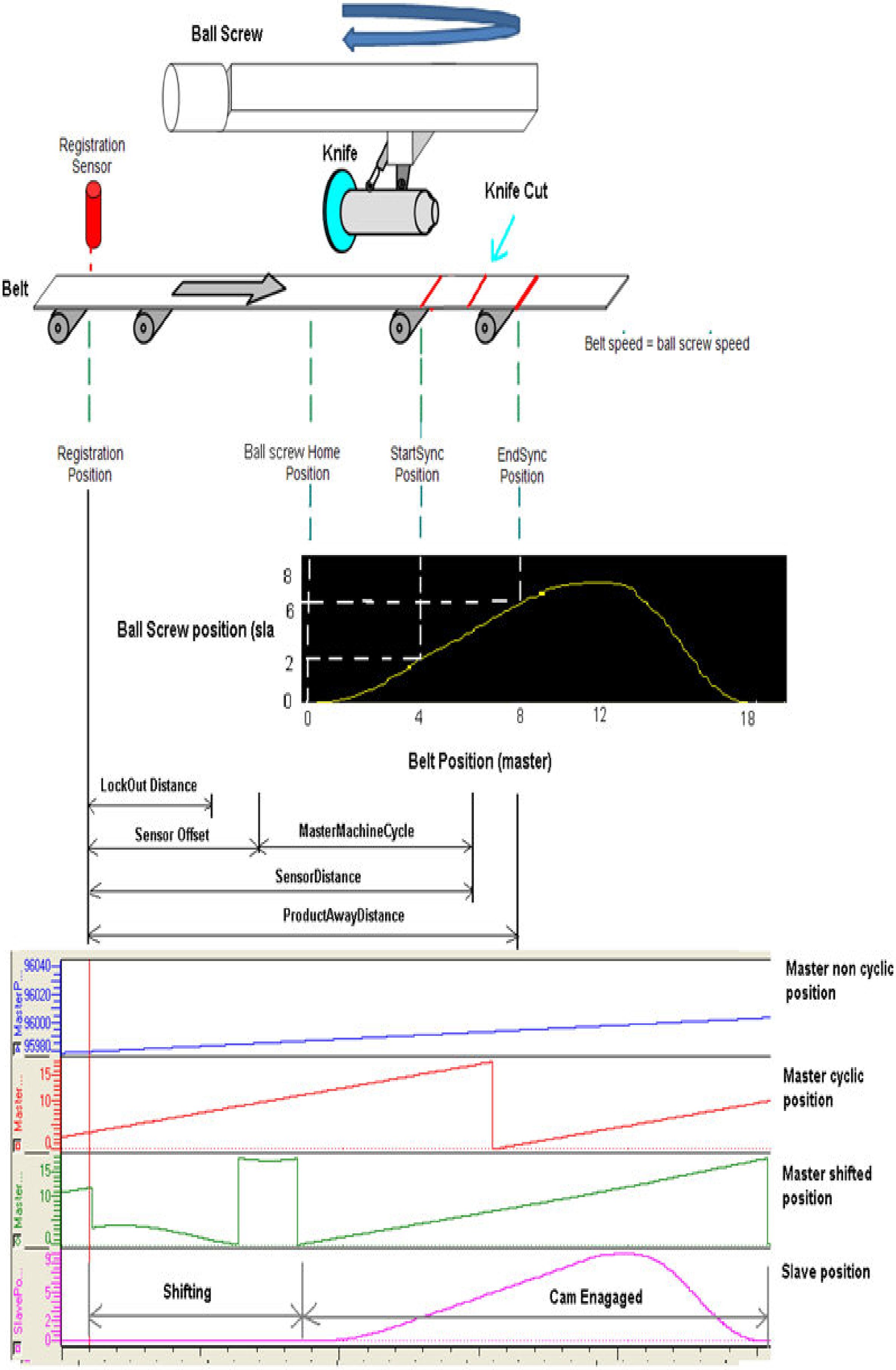
Application Example

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:

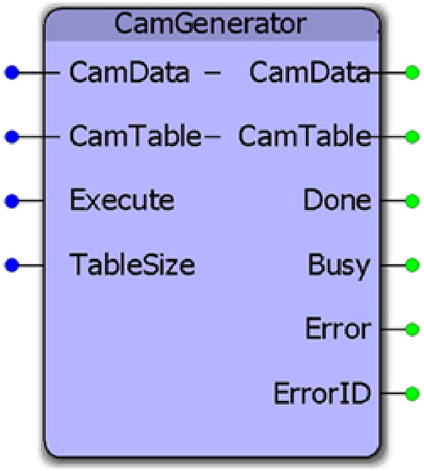
1. The ProductBuffer stores the position of each defect where a cut must be made.
2. The CamShift_Control synchronizes the master (conveyor moving the wood) and slave (saw).
3. The CamControl.Engage output must be connected to Y_CamIn.Execute. (Other logic requirements may be included if necessary.)
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_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.


001
SlaveAtHome
CamFileLoaded
CamControl_1
CamControl
Slave
Axis
Axis
Slave
Products
RegistrationData
RegistrationData
Products
CamControlData
ControlData
ControlData
CamControlData
Enable
Valid
Engage
CamEngage
DisEngage
CamDisengage
Error
ErrorID
CamControlErrID
002
CamEngage
Y_CamIn_1
Y_CamIn
Master
Master
Master
Master
Slave
Slave
Slave
Slave
Execute
InSync
CamSync
CamID
CamTableID
Busy
CamBz
EngagePosition
Active
CamActive
EngageWindow
CommandAborted
CamAborted
EngData
EngageData
Error
CamErr
TRUE
Periodic
ErrorID
CamErrID
EndOfProfile
EOP


Y_CamOut_1
Y_CamOut
003
CamDisengage
Slave
Slave
Slave
Slave
Execute
Done
CamOutDn
DisengagePosition
Busy
CamOutBz
DisengageWindow
Error
CamOutErr
DisengageData
DisengageData
ErrorID
CamOutErrID
ProductBuffer_1
ProductBuffer
Products
RegistrationData
RegistrationData
Products
Master
ProductAxis
ProductAxis
Master
004
CamFileLoaded
SlaveAtHome
Enable
Valid
BufferValid
TestMode
BufferLevel
BuferLevel
TestTrigger
Error
BufferErr
ErrorID
BufferErrID
CamShift_Control_1
CamShift_Control
Master
Master
Master
Master
Slave
Slave
Slave
Slave
Products
RegistrationData
RegistrationData
Products
CamControlData
ControlData
ControlData
CamControlData
Enable
Valid
CSCValid
TRUE
UpdateUsePointer
Shifting
CamShifting
ItemsProcessed
PartsProcessed
Error
CSCErr
ErrorID
CSCErrID


Ball Screw
Knife
Registration Sensor
Knife Cut
Belt
Belt speed = ball screw speed
Registration Position
Ball screw Home Position
StartSync Position
EndSync Position
Ball Screw position (sla
8
6
2
0
0
4
8
12
18
Belt Position (master)
LockOut Distance
Sensor Offset
MasterMachineCycle
SensorDistance
ProductAwayDistance
96040
96020
96000
95980
MasterP...
Master non cyclic position
15
10
0
Master...
Master cyclic position
15
10
0
Master...
Master shifted position
5
2
0
SlavePo...
Slave position
Shifting
Cam Enagaged

CamGenerator



This function can calculate the information required for various master / slave motion profiles. It was designed to replicate the formulas available in Yaskawa's CamTool 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 which is used to generate 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 value must be the same as the definition of the ARRAY size of the MS_Array_Type in the MotionInfo DataTypes folder of either the PLCopen or DataTypes Toolbox.	UDINT#2880
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.
B	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, certain information must be hard coded at design time, such as the size of an array. Because of this, we selected a default size of 200 for the CamSegmentArray DataType. If more segments are required, edit the Cam Toolbox’s DataType definition by changing this value. There is no practical limit on the number of segments, however the IEC code uses INT datatype for array definitions associated with this function. There is also a hard coded check for the number of segments inside the CamGenerator function block. If you change the array size, also change the line that reads:

SegmentSizeError:=(CamData.LastSegment = INT#0) OR (CamData.LastSegment > INT#200).

• 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 your cam profile requires more than 2880 master / slave pairs, this value can be increased by editing the DataTypes Toolbox > DataTypes > MotionBlockTypes definition. When changing the value, don’t forget to change the TableSize input to CamGenerator.

- 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.

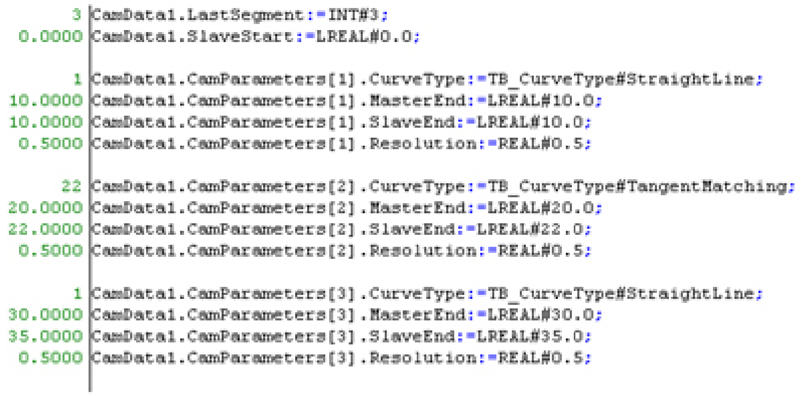
Error Description

ErrorID	Meaning
0	No error.
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..
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.
10077	Cubic Spline maximum number of consecutive segments exceeded. DataType definition for the Matrix could be increased if necessary.
10083	Unsupported Cubic Spline Sequence.
10089	Bezier Error. There should be a straight line segment before and after the bezier segment.
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.

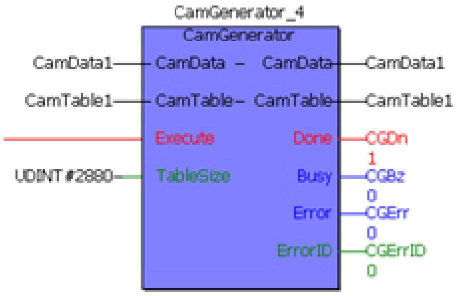
Examples

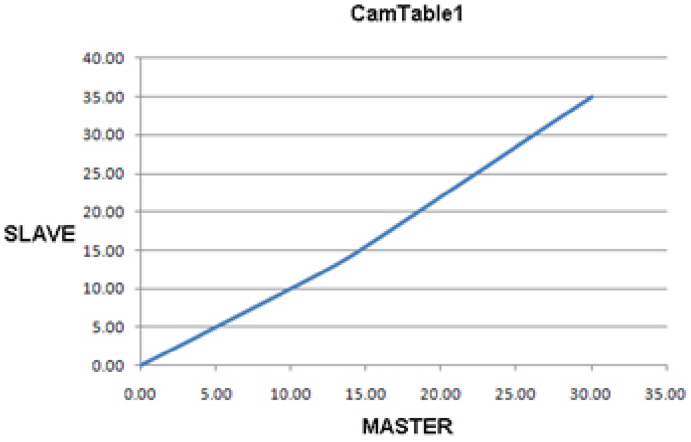
Structured text to load a CamSegmentStruct:

Example 1

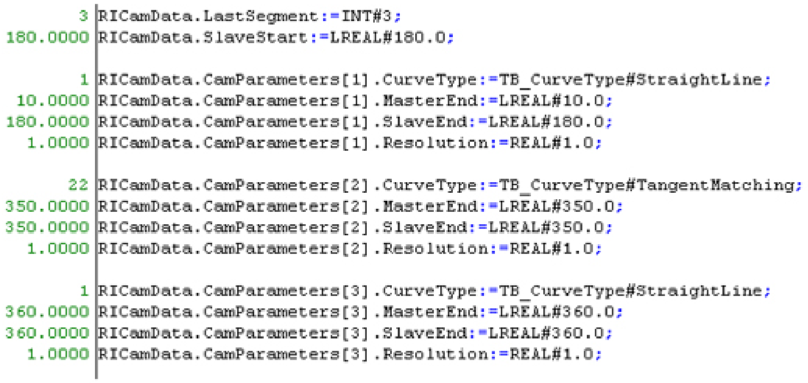


```
      3 CamData1.LastSegment:=INT#3;
 0.0000 CamData1.SlaveStart:=LREAL#0.0;

      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;
```

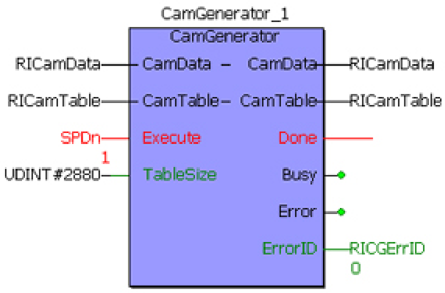


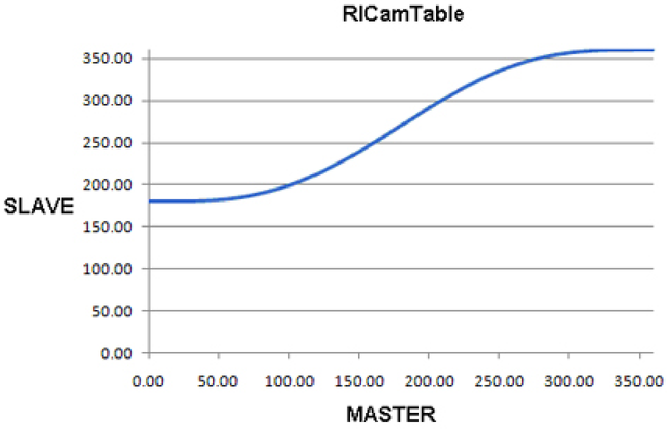


Example 2

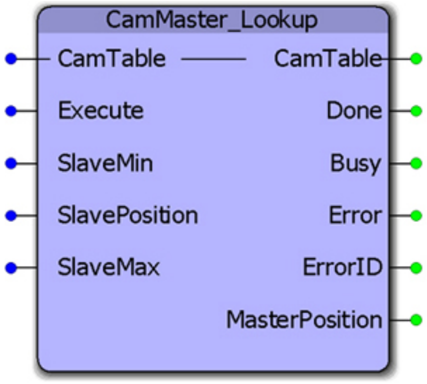


```
       3 RICamData.LastSegment:=INT#3;
180.0000 RICamData.SlaveStart:=LREAL#180.0;

       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	Default
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 retrigger 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.
B	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:

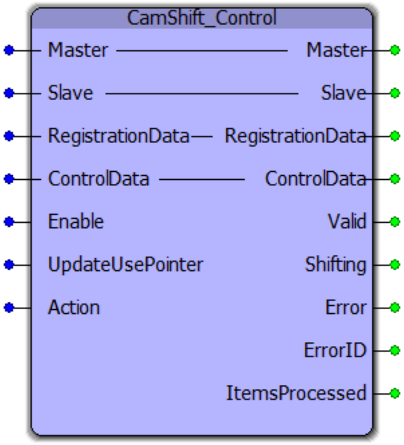
M	S
0	0
10	0
20	5
30	10
40	20

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

Error Description

ErrorID	Meaning
0	No error.
10045	SlavePosition not found in Y_MS_CAM_STRUCT.

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 uleft 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

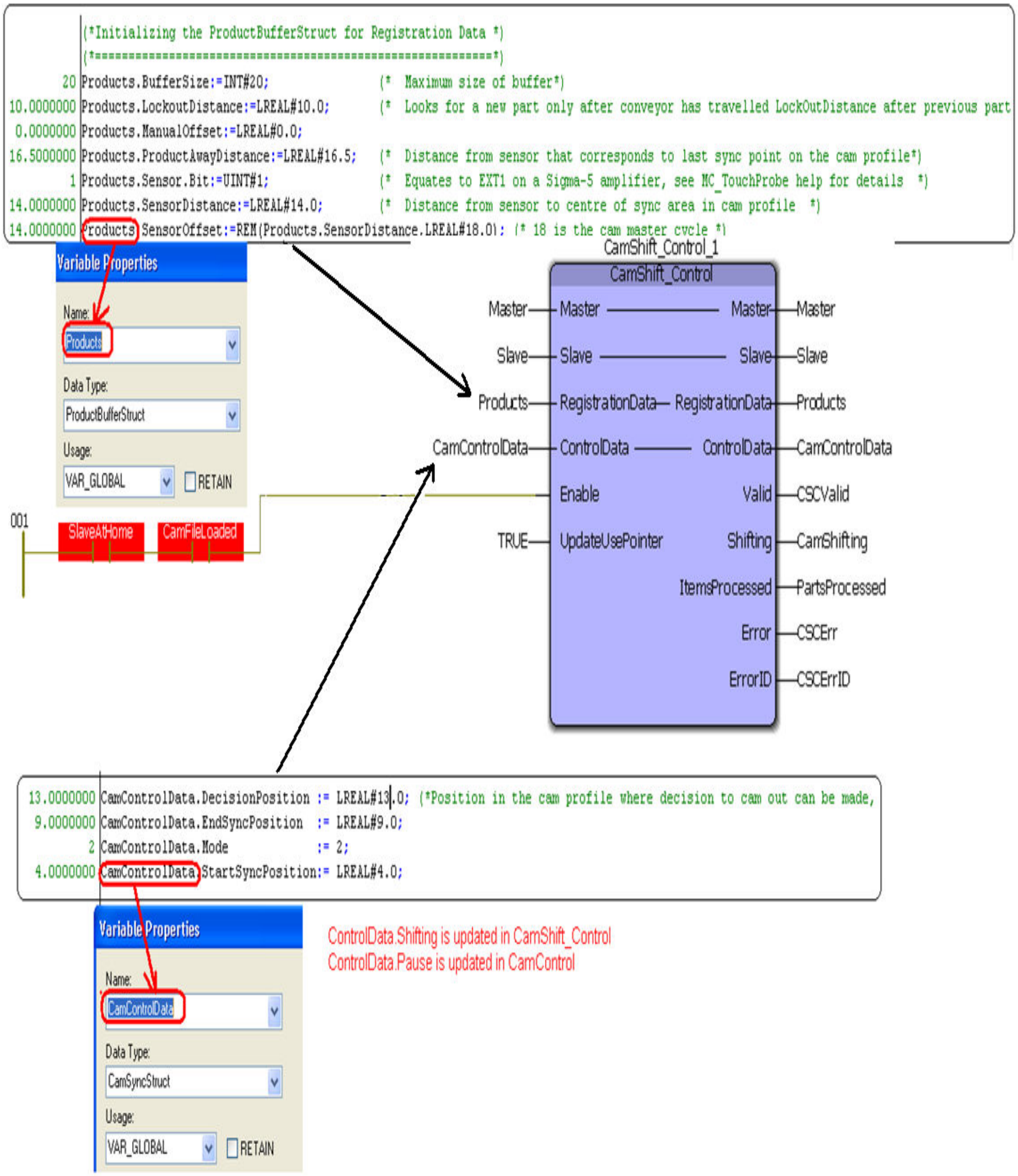
- This function block includes a Y_CamShift block, and will execute shifts at the appropriate position based on data provided by the user via the ControlData structure.
- The shifted master position is available by reading slave axis parameter 1502.
- This function block requires the ProductBuffer function 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

ErrorID	Meaning
0	No error.
11050	Cam correction (shift/offset) has been aborted by another function block.
10082	Mode Error. ControlData.Mode can only be 1 (one way cam) or 2 (two way cam).

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:=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;
```

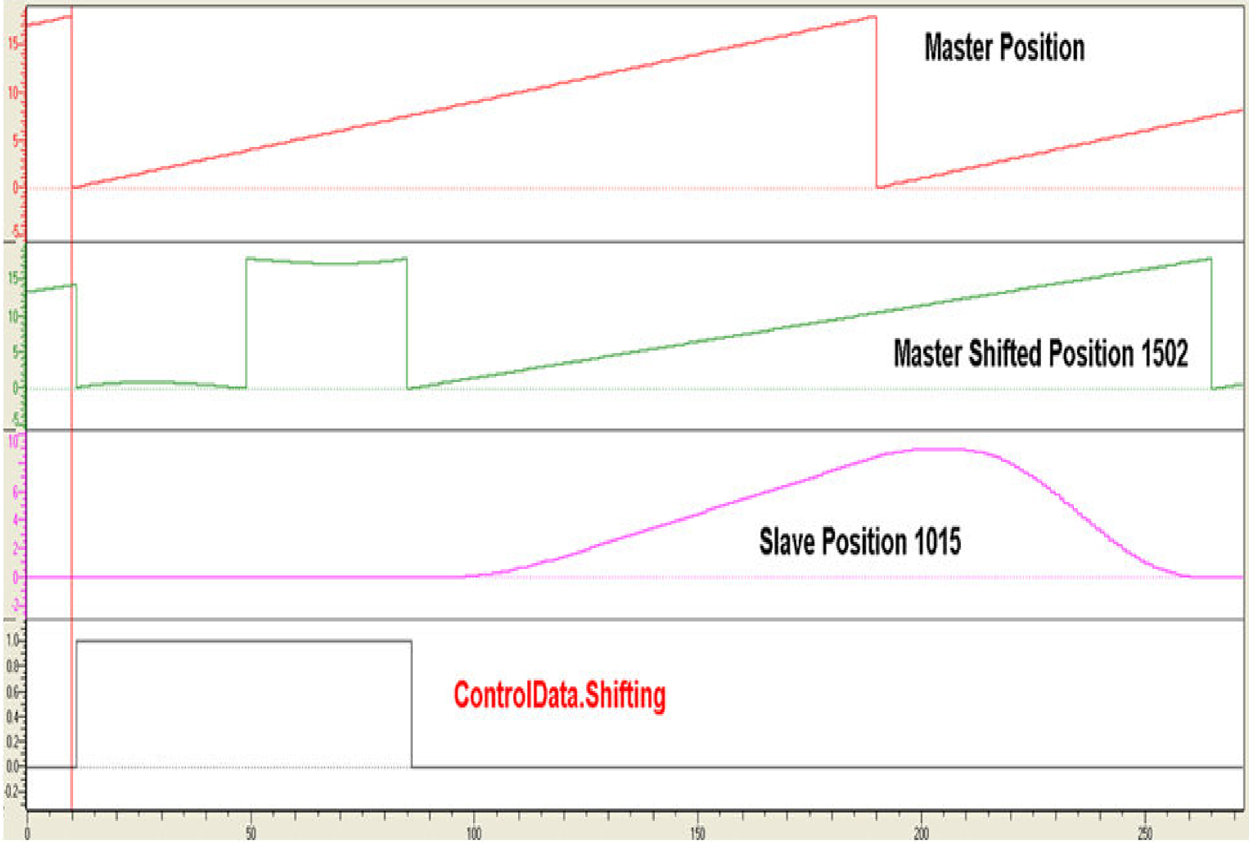
Variable Properties

Name: CamControlData

Data Type: CamSyncStruct

Usage: VAR_GLOBAL □ RETAIN

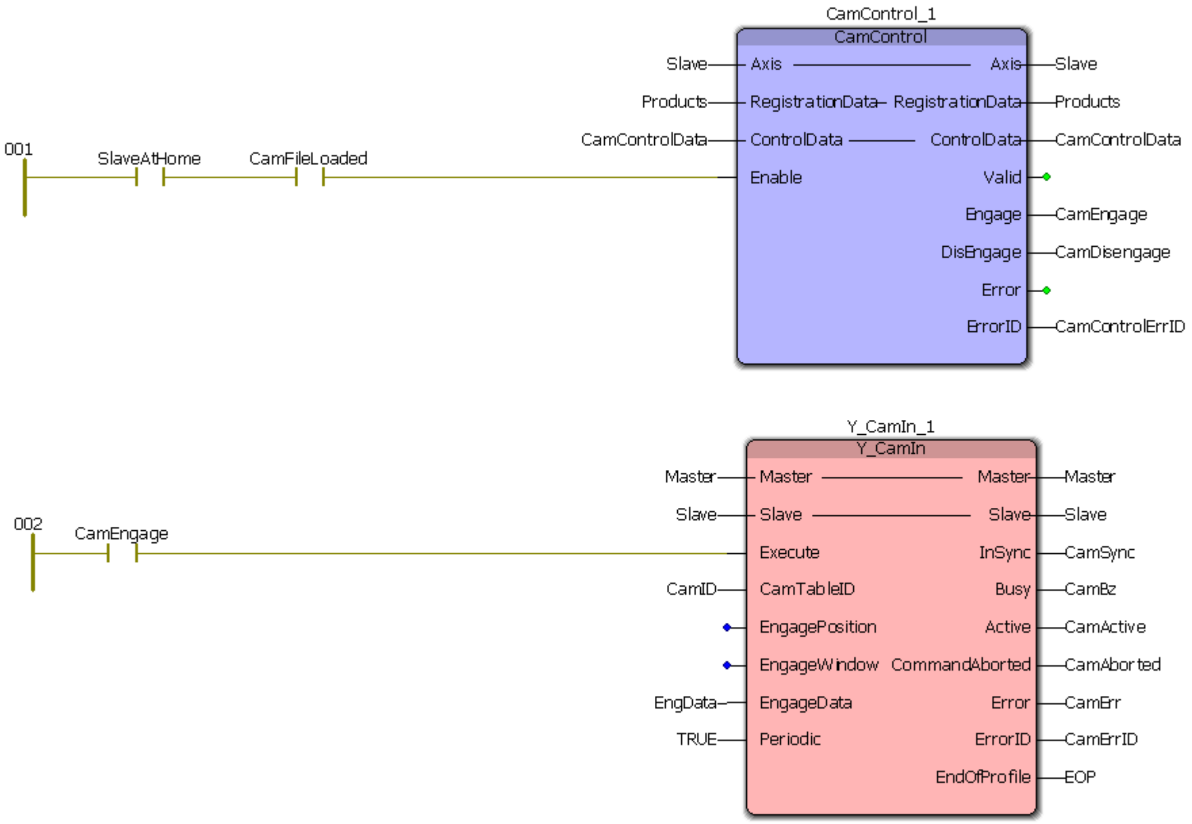
ControlData.Shifting is updated in CamShift_Control
ControlData.Pause is updated in CamControl

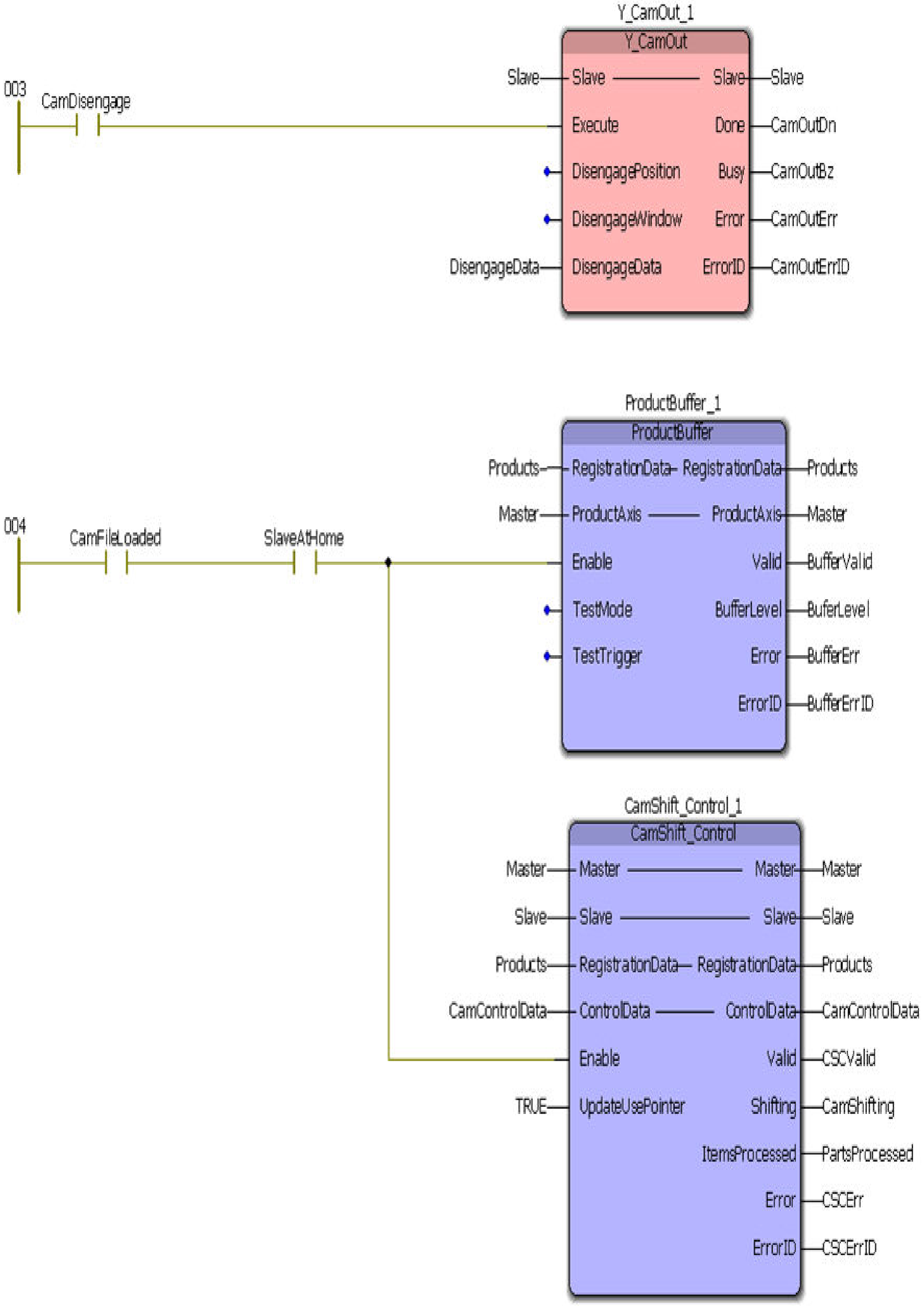


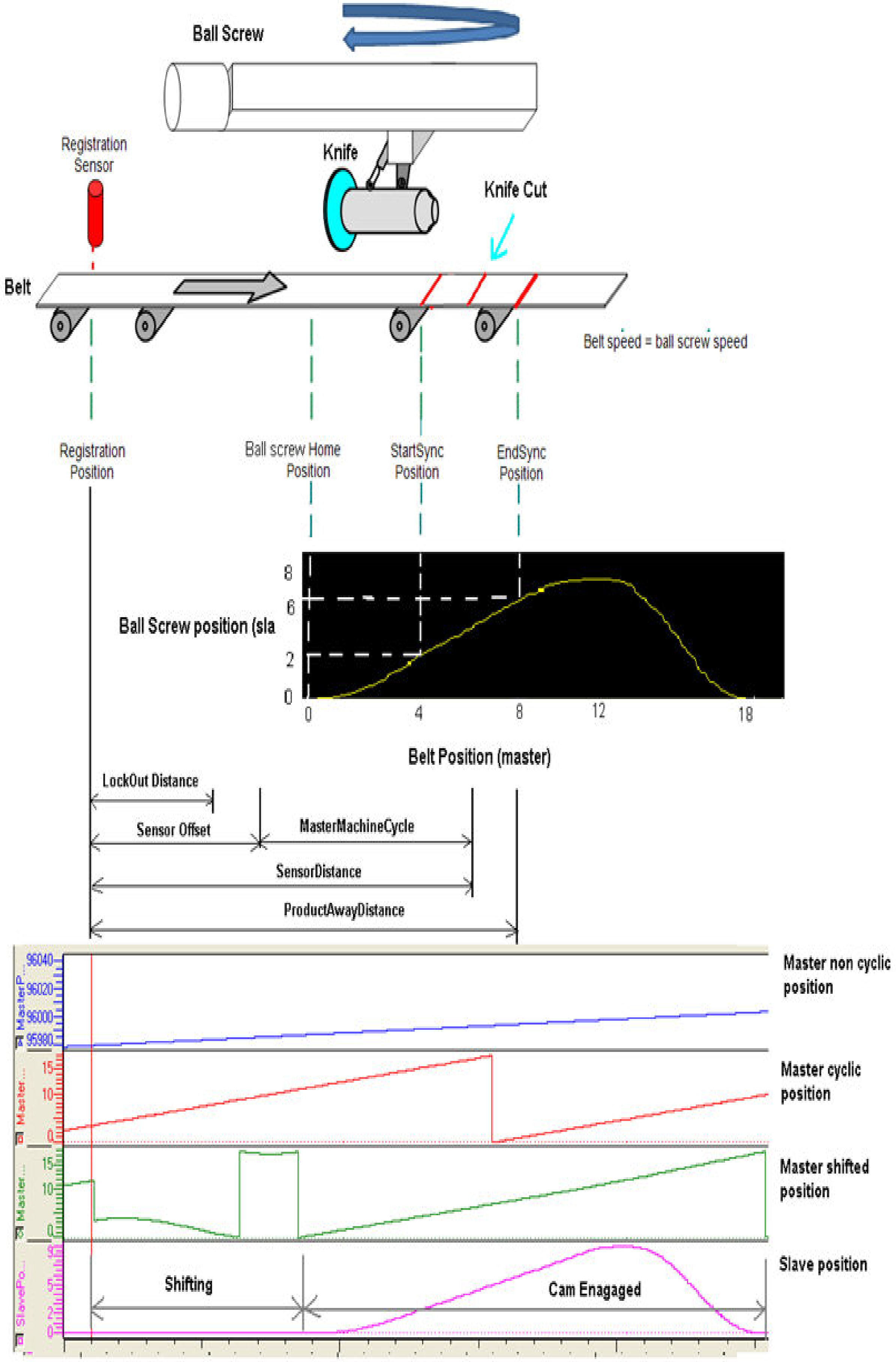
Application Example

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:

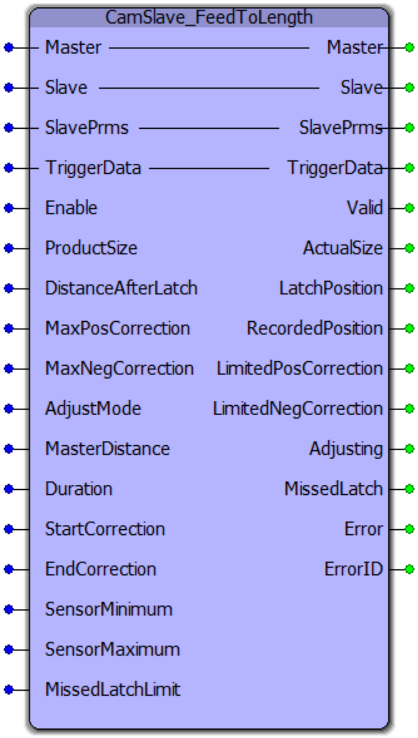
1. The ProductBuffer stores the position of each defect where a cut must be made.
2. The CamShift_Control synchronizes the master (conveyor moving the wood) and slave (saw).
3. The CamControl.Engage output must be connected to Y_CamIn.Execute. (Other logic requirements may be included if necessary.)
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_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.


001
SlaveAtHome
CamFileLoaded
CamControl_1
CamControl
Slave
Axis
Axis
Slave
Products
RegistrationData
RegistrationData
Products
CamControlData
ControlData
ControlData
CamControlData
Enable
Valid
Engage
CamEngage
DisEngage
CamDisengage
Error
ErrorID
CamControlErrID
002
CamEngage
Y_CamIn_1
Y_CamIn
Master
Master
Master
Master
Slave
Slave
Slave
Slave
Execute
InSync
CamSync
CamID
CamTableID
Busy
CamBz
EngagePosition
Active
CamActive
EngageWindow
CommandAborted
CamAborted
EngData
EngageData
Error
CamErr
TRUE
Periodic
ErrorID
CamErrID
EndOfProfile
EOP


003
CamDisengage
Y_CamOut_1
Y_CamOut
Slave
Slave
Slave
Slave
Execute
Done
CamOutDn
DisengagePosition
Busy
CamOutBz
DisengageWindow
Error
CamOutErr
DisengageData
DisengageData
ErrorID
CamOutErrID
004
CamFileLoaded
SlaveAtHome
ProductBuffer_1
ProductBuffer
Products
RegistrationData
RegistrationData
Products
Master
ProductAxis
ProductAxis
Master
Enable
Valid
BufferValid
TestMode
BufferLevel
BuferLevel
TestTrigger
Error
BufferErr
ErrorID
BufferErrID
CamShift_Control_1
CamShift_Control
Master
Master
Master
Master
Slave
Slave
Slave
Slave
Products
RegistrationData
RegistrationData
Products
CamControlData
ControlData
ControlData
CamControlData
Enable
Valid
CSCValid
TRUE
UpdateUsePointer
Shifting
CamShifting
ItemsProcessed
PartsProcessed
Error
CSCErr
ErrorID
CSCErrID


Ball Screw
Registration Sensor
Knife
Knife Cut
Belt
Belt speed = ball screw speed
Registration Position
Ball screw Home Position
StartSync Position
EndSync Position
8
6
2
0
Ball Screw position (sla
0
4
8
12
18
Belt Position (master)
LockOut Distance
Sensor Offset
MasterMachineCycle
SensorDistance
ProductAwayDistance
96040
96020
96000
95980
MasterP...
Master non cyclic position
15
10
0
Master...
Master cyclic position
15
10
0
Master...
Master shifted position
9
5
2
0
SlavePo...
Shifting
Cam Enagaged
Slave position

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#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.	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 needs to 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

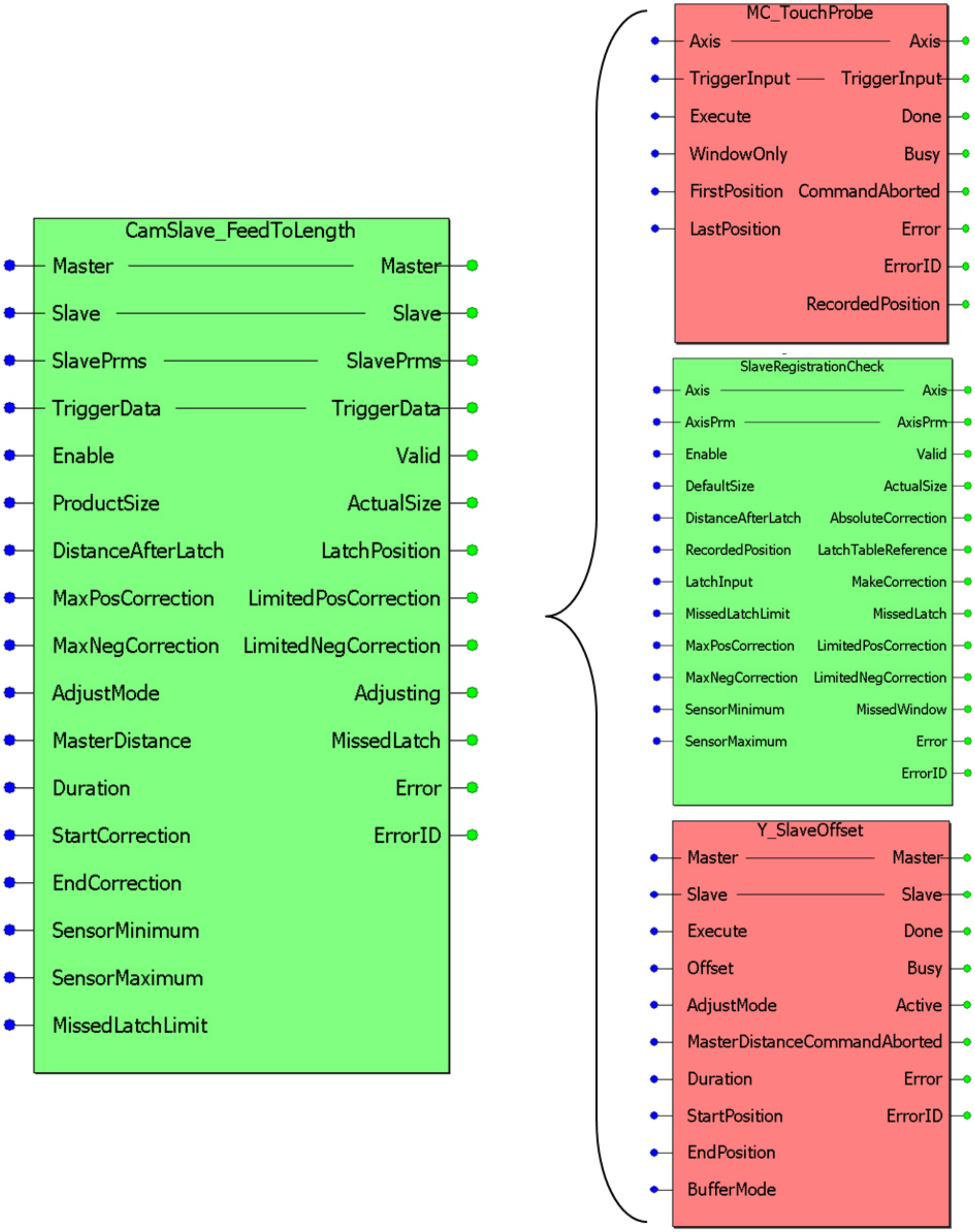
ErrorID	Meaning
0	No error.
4370	The move could not be started because motion is prohibited. Possible causes: 1) The axis may not be enabled. Check MC_Power.Status output. 2) An MC_Stop.Execute might be held high - no other block can override the Stopping state. 3) Verify that the limit switches are 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.
4374	Torque move prohibited while non-torque moves queued or in progress.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
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.
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.
4649	Invalid adjust mode.
4657	Distance parameter is less than or equal to zero.
4663	Specified time was less than zero.
4673	StartPosition is outside of master's range.
4674	EndPosition is outside of master's range.

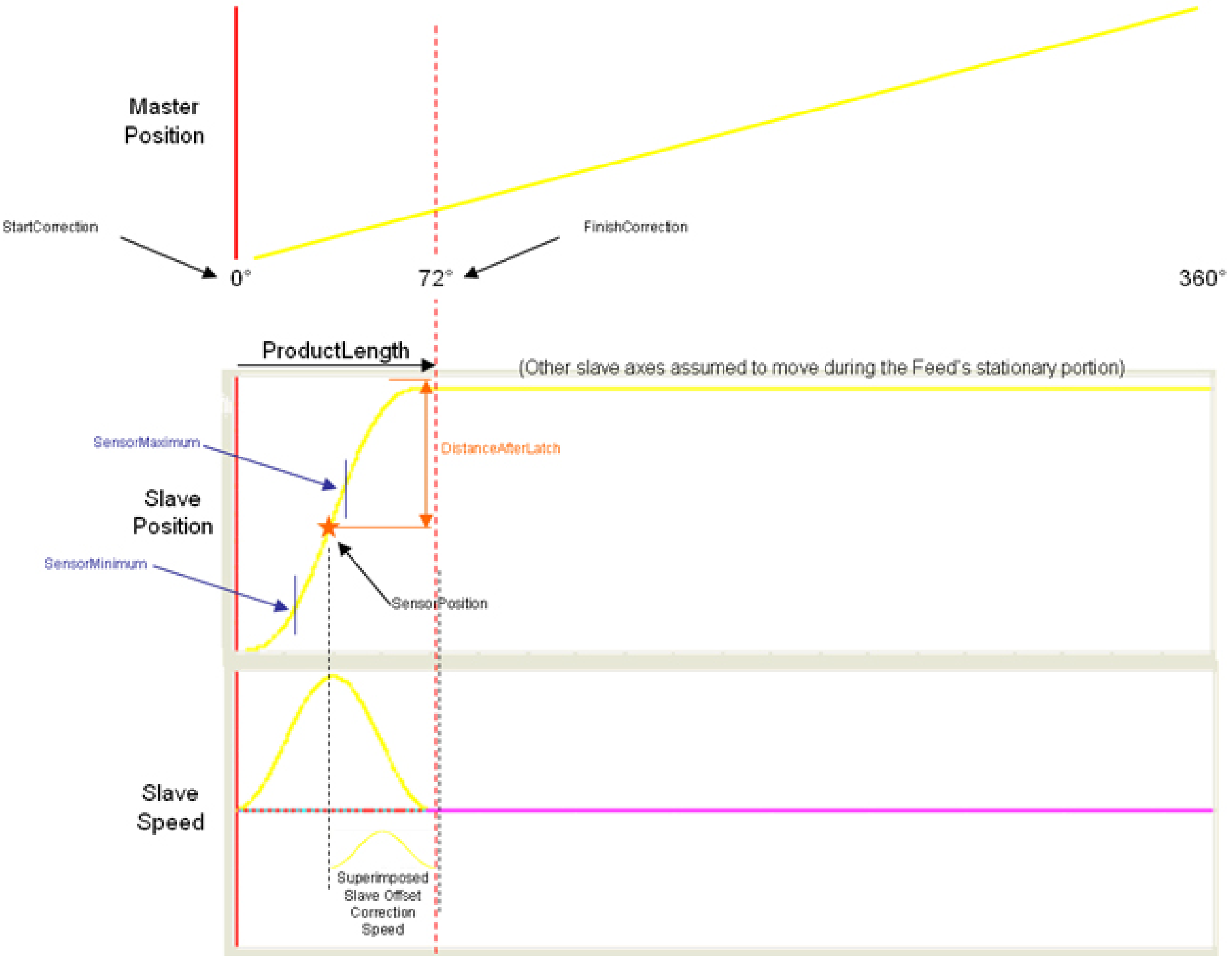
10020	ProductSize cannot be less than or equal to zero.
10021	Maximum allowed consecutive missed registration marks reached.
10025	SensorMinimum must be less than SensorMaximum.
10053	DataPoint Error.
10086	MaxPosCorrection must be zero or positive, MaxNegCorrection must be or zero or negative.
57620	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.

Applications

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

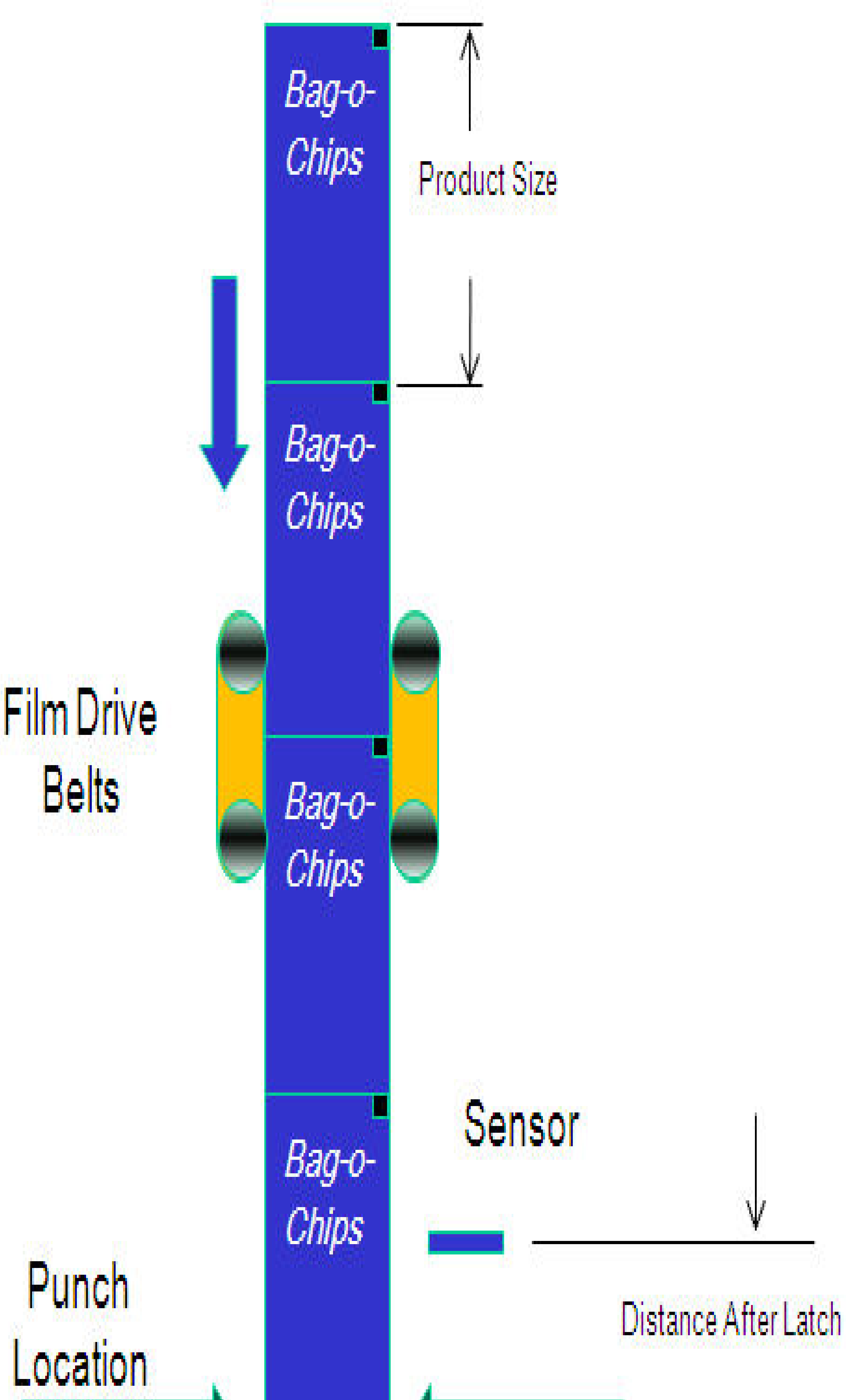
Overview of Supporting Function Blocks



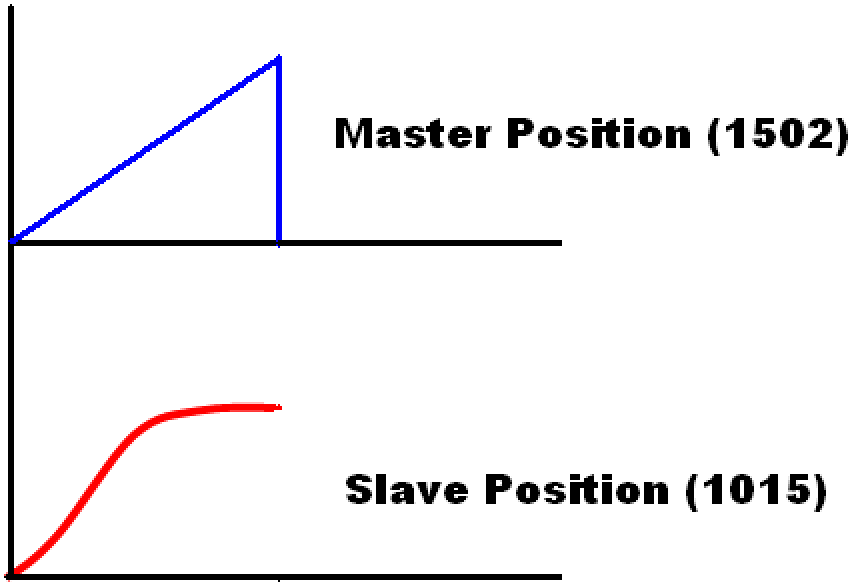


Application Example

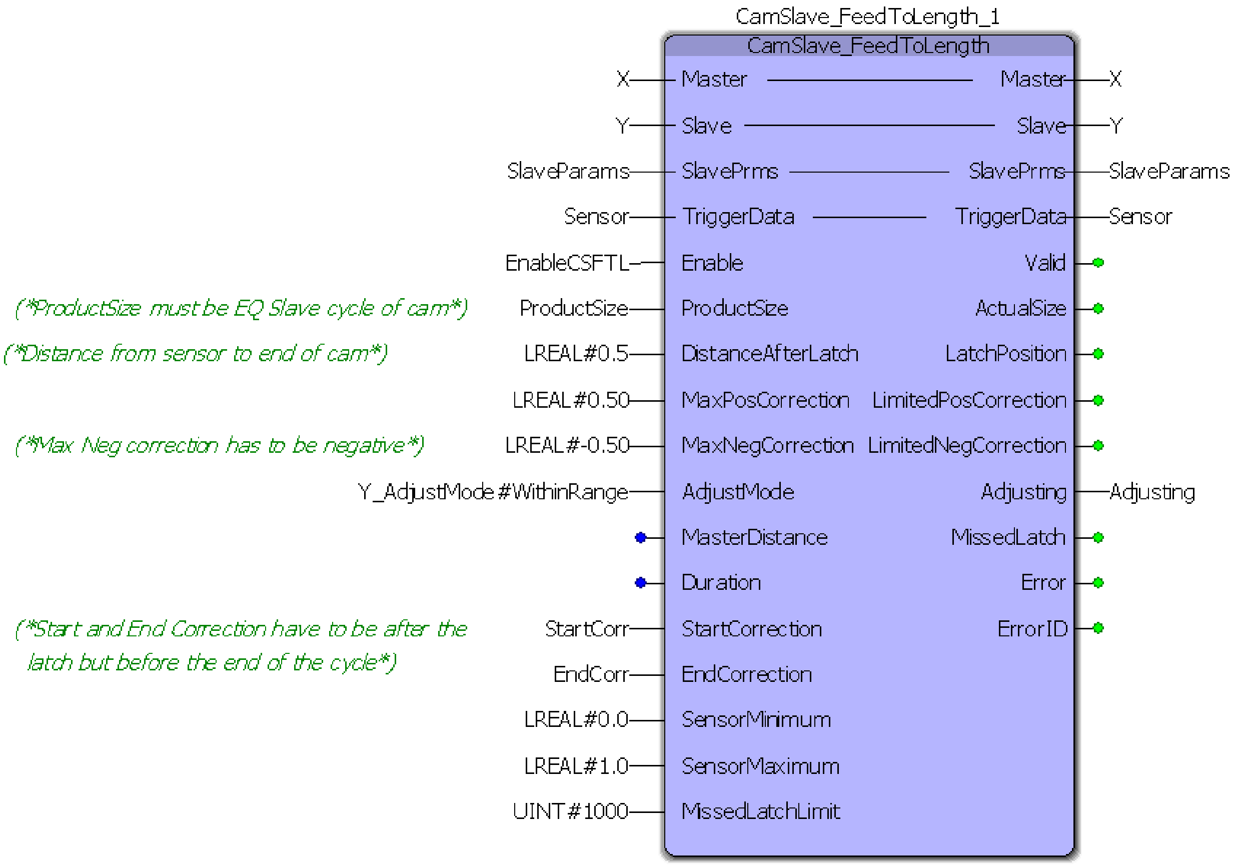
Consider a form fill and seal application as shown below. Feed belts control payout of film for the form fill and seal machine. Distance After Latch is set to align the end of bag with the cutter/punch


Product Size
Bag-o-Chips
Bag-o-Chips
Film Drive Belts
Bag-o-Chips
Sensor
Bag-o-Chips
Punch Location
Distance After Latch

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.



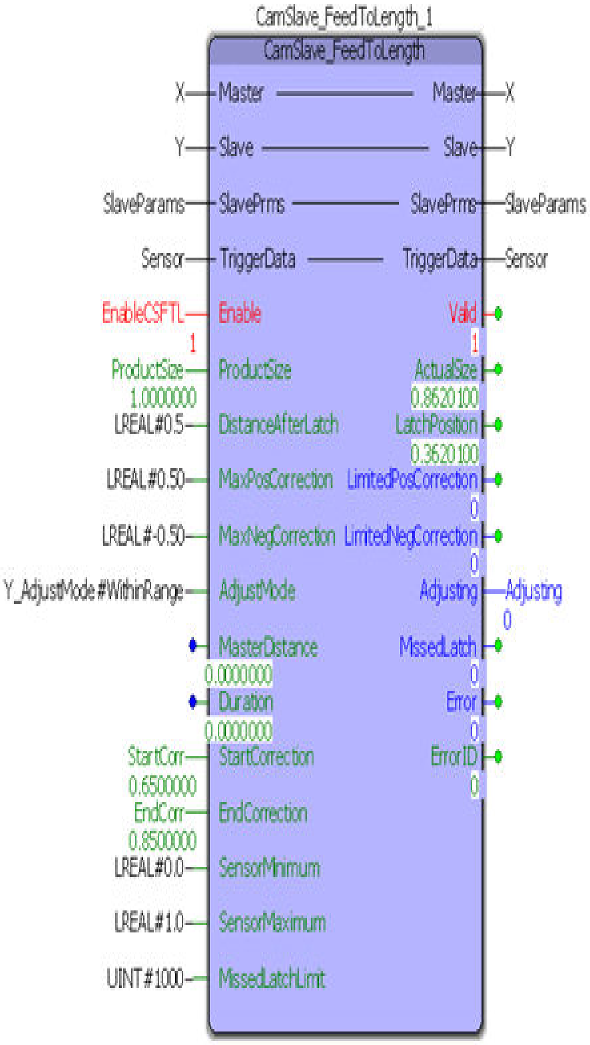
A sample screen shot of data that needs to be entered for the system described above is shown in the figure below. Care should be taken to ensure that the input parameters will generate motion that is physically achiev able and desirable by the slave axis.

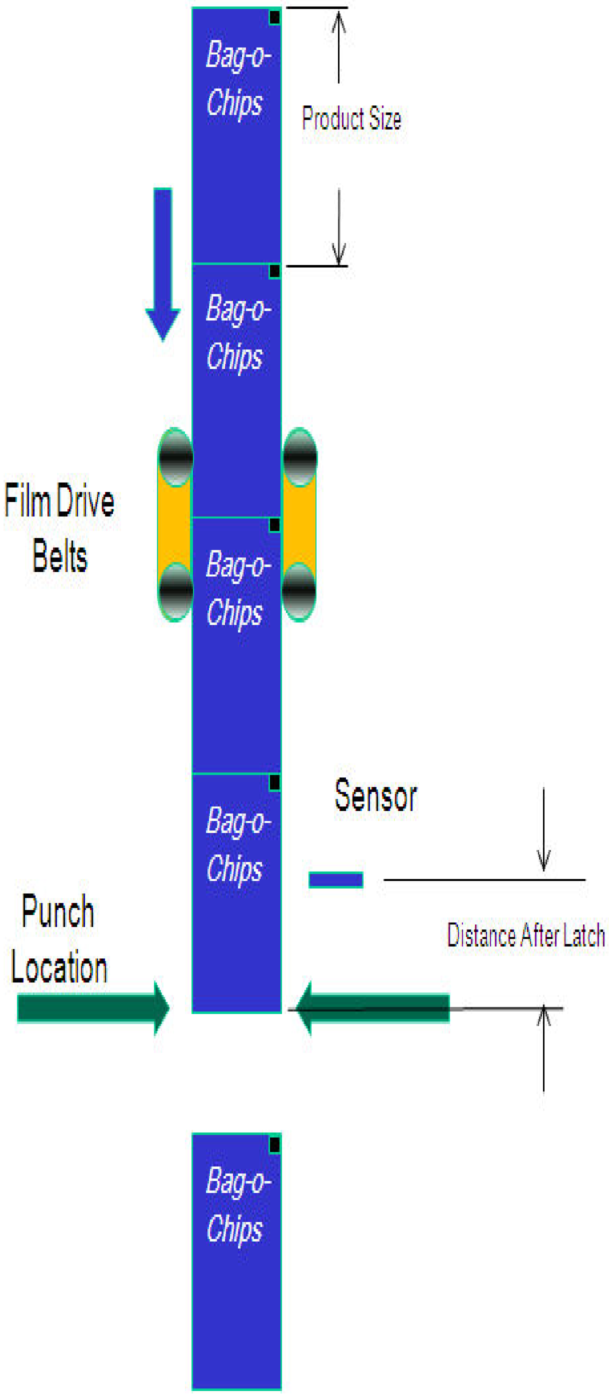


CamSlave_FeedToLength_1
CamSlave_FeedToLength

X — Master — Master — X
Y — Slave — Slave — Y
SlaveParams — SlavePrms — SlavePrms — SlaveParams
Sensor — TriggerData — TriggerData — Sensor
EnableCSFTL — Enable — Valid
(*ProductSize must be EQ Slave cycle of cam*) ProductSize — ProductSize — ActualSize
(*Distance from sensor to end of cam*) LREAL#0.5 — DistanceAfterLatch — LatchPosition
LREAL#0.50 — MaxPosCorrection — LimitedPosCorrection
(*Max Neg correction has to be negative*) LREAL#-0.50 — MaxNegCorrection — LimitedNegCorrection
Y_AdjustMode#WithinRange — AdjustMode — Adjusting — Adjusting
MasterDistance — MissedLatch
Duration — Error
(*Start and End Correction have to be after the latch but before the end of the cycle*) StartCorr — StartCorrection — ErrorID
EndCorr — EndCorrection
LREAL#0.0 — SensorMinimum
LREAL#1.0 — SensorMaximum
UINT#1000 — MissedLatchLimit

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.


CamSlave_FeedToLength_1
CamSlave_FeedToLength
X Master Master X
Y Slave Slave Y
SlaveParams SlavePrms SlavePrms SlaveParams
Sensor TriggerData TriggerData Sensor
EnableCSFTL Enable Valid
1
1
ProductSize ProductSize ActualSize
1.0000000
0.8620100
LREAL#0.5 DistanceAfterLatch LatchPosition
0.3620100
LREAL#0.50 MaxPosCorrection LimitedPosCorrection
0
LREAL#-0.50 MaxNegCorrection LimitedNegCorrection
0
Y_AdjustMode#WithinRange AdjustMode Adjusting Adjusting
0
MasterDistance MissedLatch
0.0000000
0
Duration Error
0.0000000
0
StartCorr StartCorrection ErrorID
0.6500000
0
EndCorr EndCorrection
0.8500000
LREAL#0.0 SensorMinimum
LREAL#1.0 SensorMaximum
UINT#1000 MissedLatchLimit



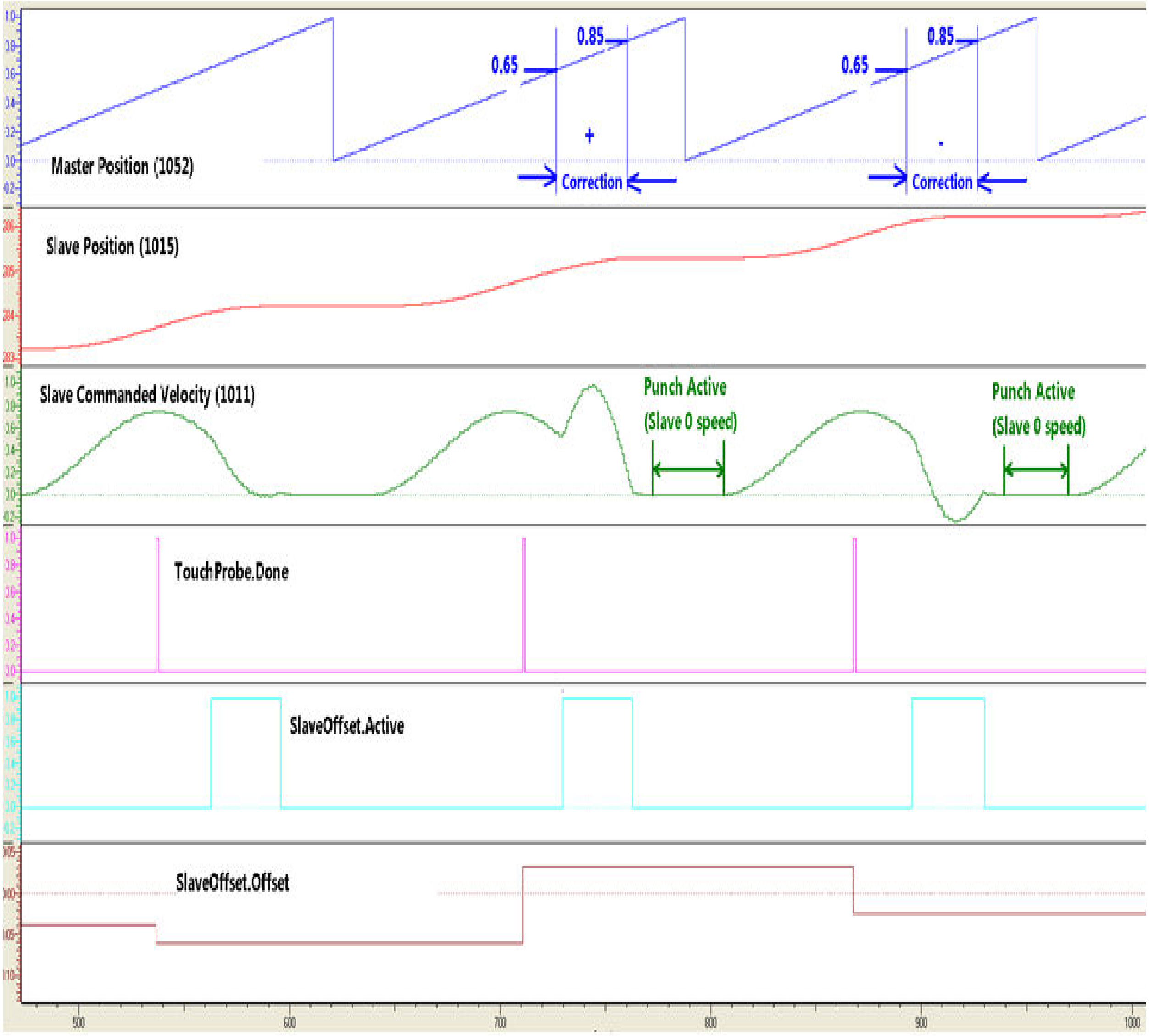
The calculation on how much adjustment needs to be made to make the slave axis (film feed) place the film exactly at the cutter/pinch location is explained below:

Correction = Nominal part size (1.0) Actual bag length (0.86201)= -0.1379

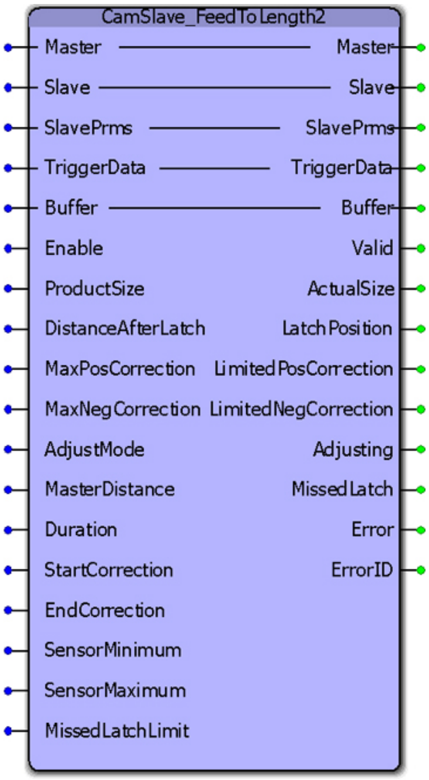
This will be the amount of offset added/subtracted (for this cycle) to any previous offsets in the slave position.

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

The first occurrence of TouchProbe.Done in the figure triggers a calculation that shows 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. This is seen as the spike in slave velocity between 0.65 and 0.86 units of the master cycle. The last registration mark in the figure (in the middle of the third master cycle) triggers a calculation that results in a short bag and a negative offset. This is seen as the dip in slave velocity between 0.65 and 0.86 units of the master cycle.


Master Position (1052)
0.65
0.85
+
Correction
0.65
0.85
-
Correction
Slave Position (1015)
Slave Commanded Velocity (1011)
Punch Active
(Slave 0 speed)
Punch Active
(Slave 0 speed)
TouchProbe.Done
SlaveOffset.Active
SlaveOffset.Offset
500
600
700
800
900
1000

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

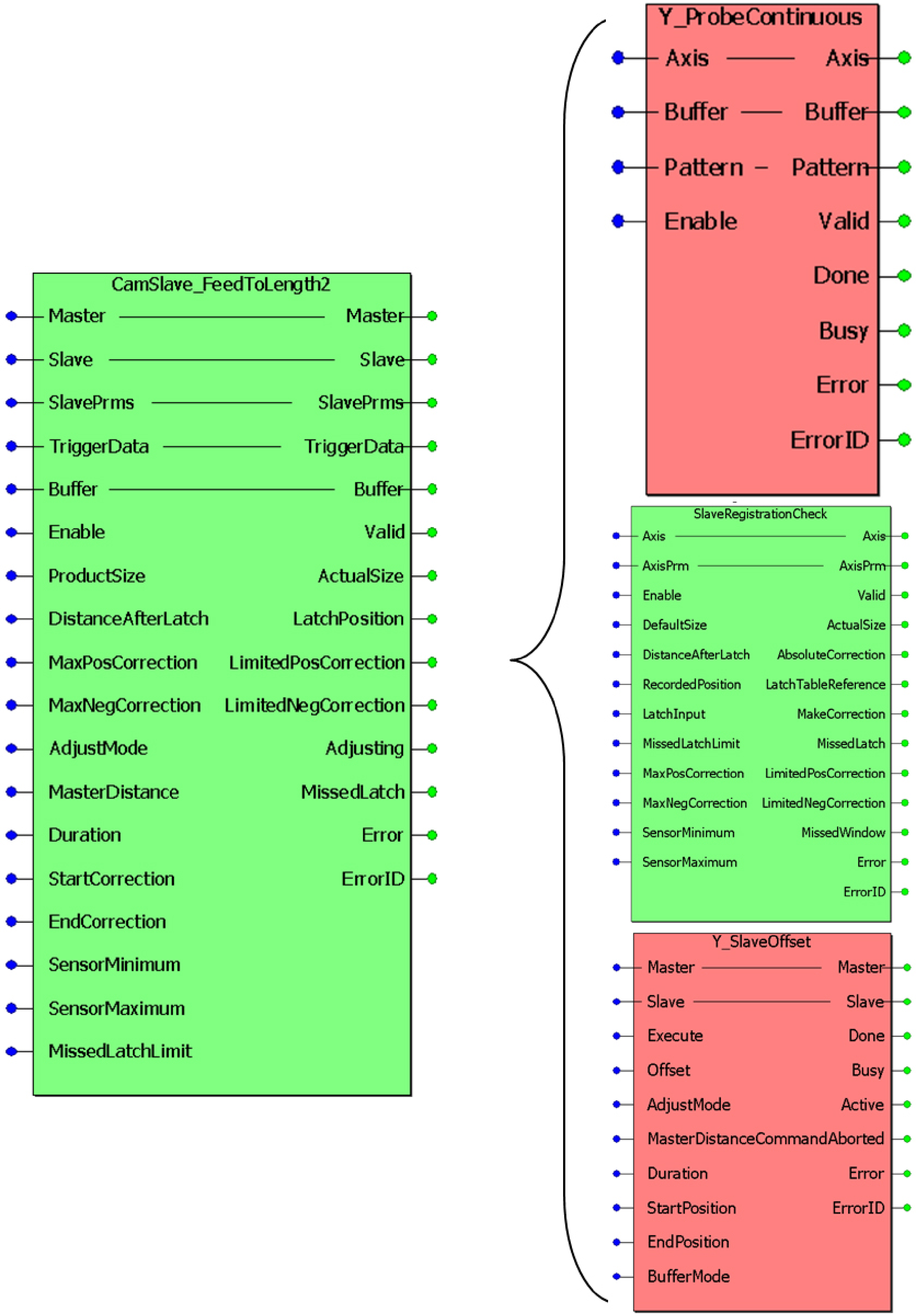
Error Description

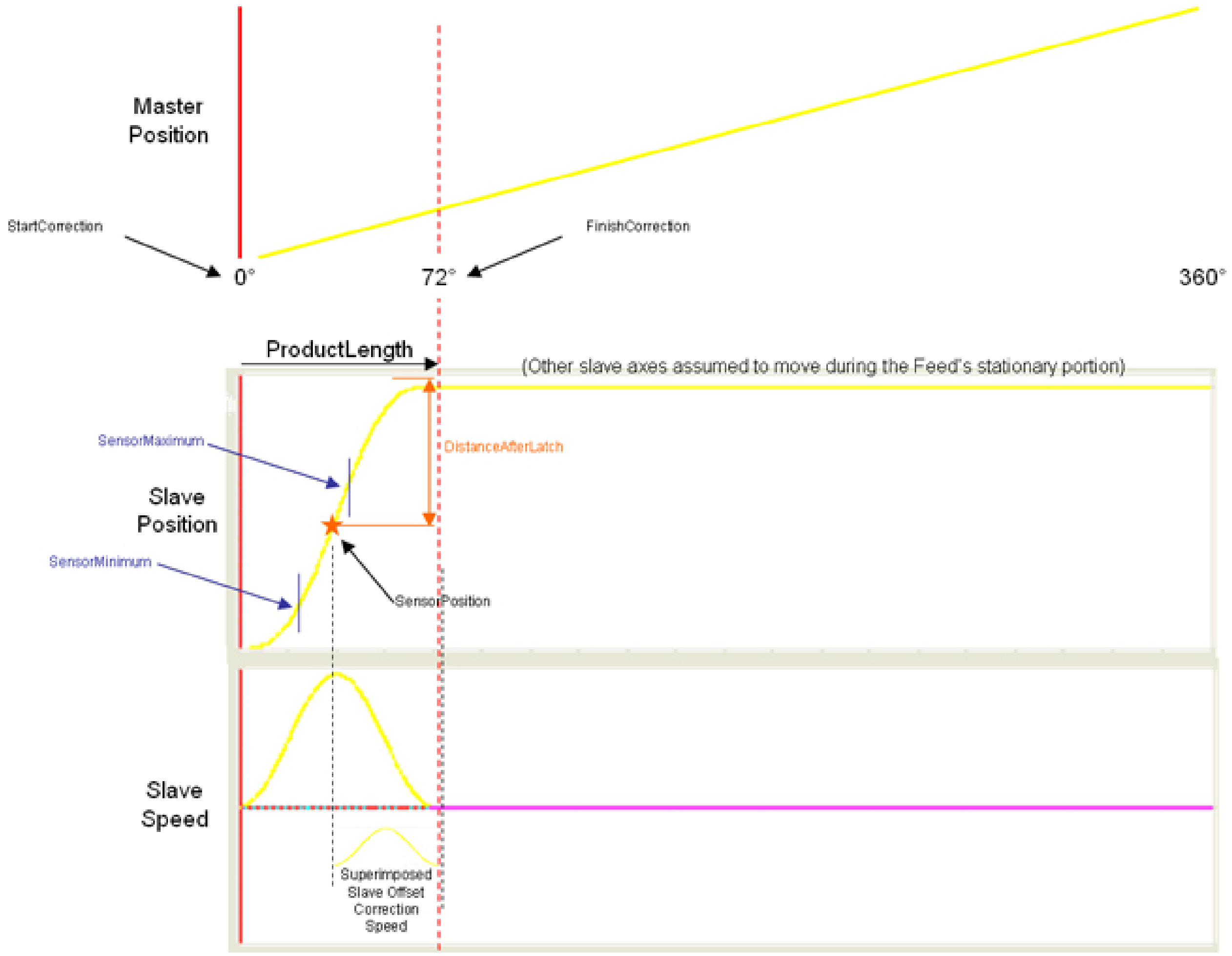
ErrorID	Meaning
0	No error.
4370	The move could not be started because motion is prohibited. Possible causes: 1) The axis may not be enabled. Check MC_Power.Status output. 2) An MC_Stop.Execute might be held high - no other block can override the Stopping state. 3) Verify that the limit switches are 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.
4374	Torque move prohibited while non-torque moves queued or in progress.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
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.
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.
4649	Invalid adjust mode.
4657	Distance parameter is less than or equal to zero.
4663	Specified time was less than zero.
4673	StartPosition is outside of master's range.
4674	EndPosition is outside of master's range.
10020	ProductSize cannot be less than or equal to zero.
10021	Maximum allowed consecutive missed registration marks reached.
10025	SensorMinimum must be less than SensorMaximum.
10053	DataPoint Error.
10086	MaxPosCorrection must be zero or positive, MaxNegCorrection must be or zero or negative.
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.

Applications

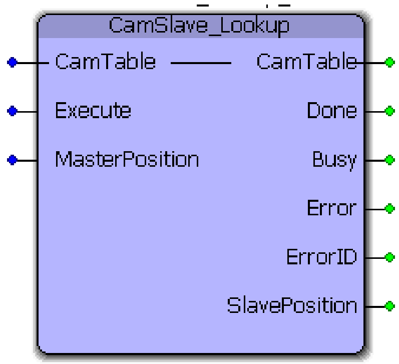
- Label Feeder
- Punch Press

Overview of Supporting Function Blocks


CamSlave_FeedToLength2
Master Master
Slave Slave
SlavePrms SlavePrms
TriggerData TriggerData
Buffer Buffer
Enable Valid
ProductSize ActualSize
DistanceAfterLatch LatchPosition
MaxPosCorrection LimitedPosCorrection
MaxNegCorrection LimitedNegCorrection
AdjustMode Adjusting
MasterDistance MissedLatch
Duration Error
StartCorrection ErrorID
EndCorrection
SensorMinimum
SensorMaximum
MissedLatchLimit
Y_ProbeContinuous
Axis Axis
Buffer Buffer
Pattern Pattern
Enable Valid
Done
Busy
Error
ErrorID
SlaveRegistrationCheck
Axis Axis
AxisPrm AxisPrm
Enable Valid
DefaultSize ActualSize
DistanceAfterLatch AbsoluteCorrection
RecordedPosition LatchTableReference
LatchInput MakeCorrection
MissedLatchLimit MissedLatch
MaxPosCorrection LimitedPosCorrection
MaxNegCorrection LimitedNegCorrection
SensorMinimum MissedWindow
SensorMaximum Error
ErrorID
Y_SlaveOffset
Master Master
Slave Slave
Execute Done
Offset Busy
AdjustMode Active
MasterDistanceCommandAborted
Duration Error
StartPosition ErrorID
EndPosition
BufferMode


Master Position
StartCorrection
FinishCorrection
0°
72°
360°
ProductLength
(Other slave axes assumed to move during the Feed's stationary portion)
SensorMaximum
DistanceAfterLatch
Slave Position
SensorMinimum
SensorPosition
Slave Speed
Superimposed Slave Offset Correction Speed

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.	FALSE
V	MasterPosition	LREAL	The position of the master axis for which the corresponding slave 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.	
B	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 MasterPostion 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.

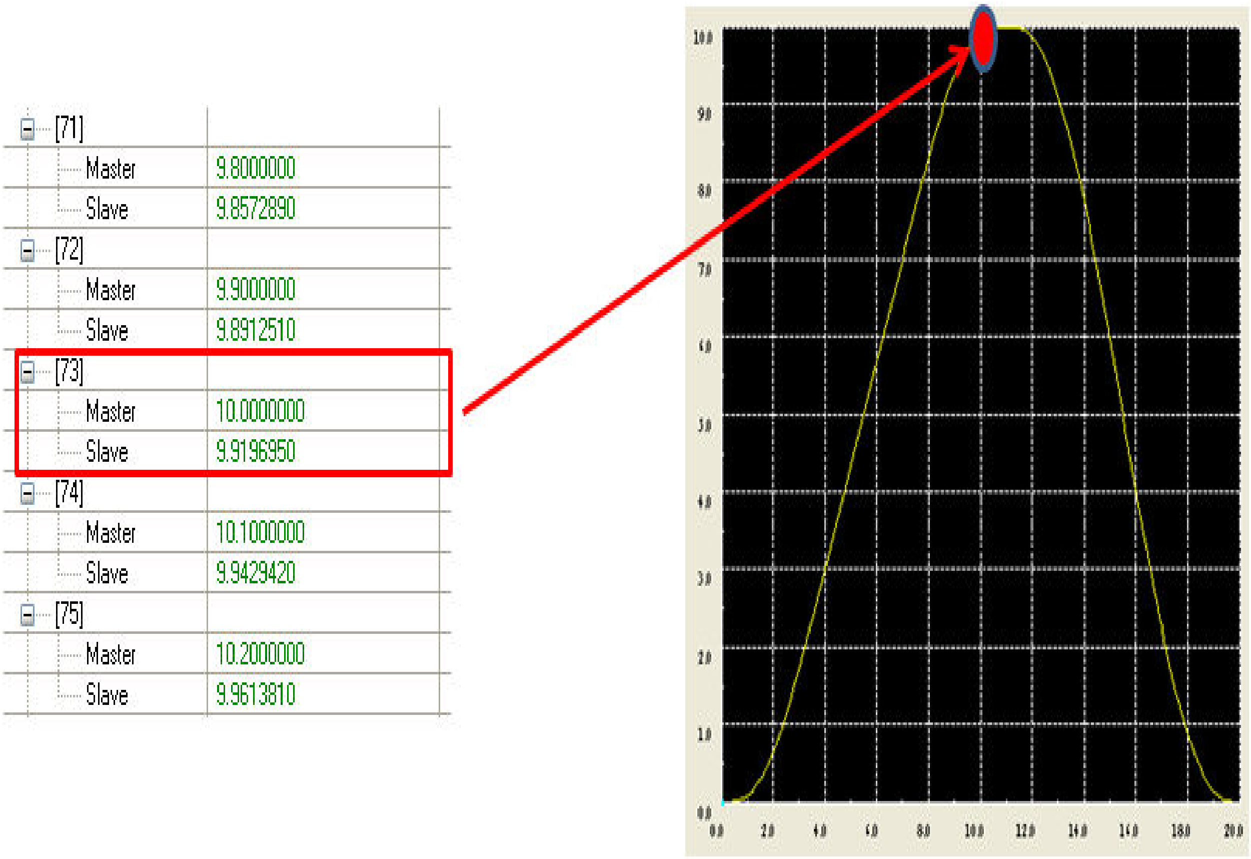
See the CamSlave_Lookup eLearning Module on Yaskawa's YouTube Channel.

Error Description

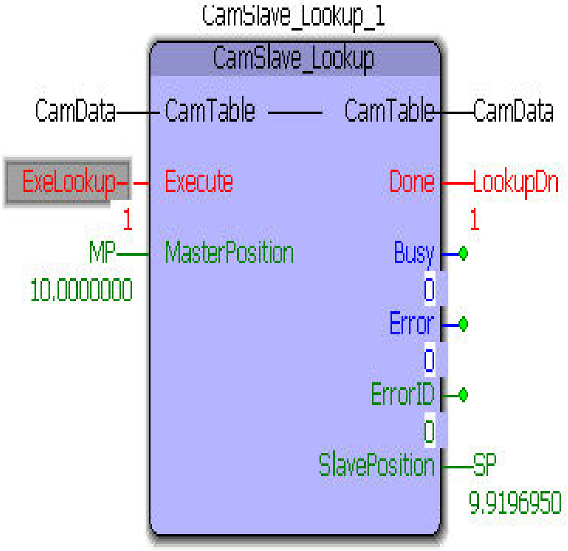
ErrorID	Meaning
0	No error.
10114	Incorrect cam table size (check the CamTable.Header.Datasize).
10045	SlavePosition not found in Y_MS_CAM_STRUCT.

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_Lookup_1

CamSlave_Lookup

CamData — CamTable —— CamTable — CamData

ExeLookup — Execute   Done — LookupDn

1   1

MP — MasterPosition   Busy

10.0000000   0

Error

0

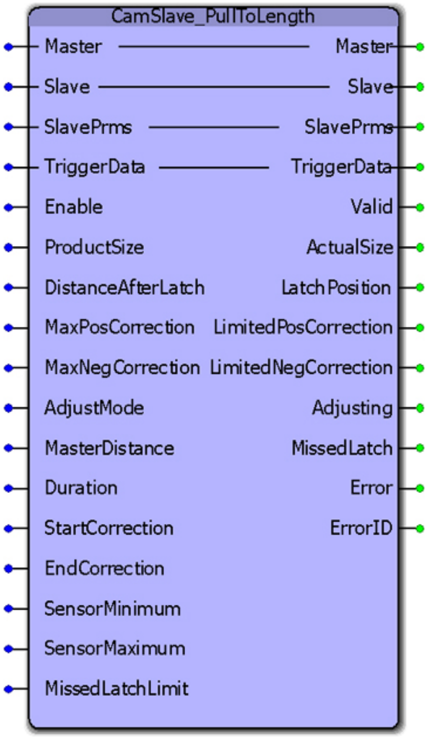
ErrorID

0

SlavePosition — SP

9.9196950

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	An ENUM for TIME or range of master correction, with the following values: Y_AdjustMode#MasterDistance: The adjustment starts immediately and completes when the master has travelled the specified MasterDistance. Y_AdjustMode#ElapsedTime: The adjustment starts immediately and completes within the specified Time. Y_AdjustMode#WithinRange: The adjustment starts when the master first crosses the StartPosition and completes when the master reaches the EndPosition.	INT#0 (Y_AdjustMode#MasterDistance)
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.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

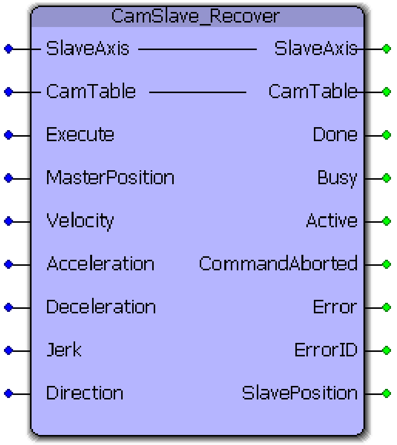
Notes

This function block is 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

ErrorID	Meaning
0	No error.
10020	ProductSize cannot be less than or equal to zero.
10021	Maximum allowed consecutive missed registration marks reached.
10025	SensorMinimum must be less than SensorMaximum.
10053	DataPoint Error.
10086	MaxPosCorrection must be zero or positive, MaxNegCorrection must be or zero or negative.

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	
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	LREAL#0.0

			and moves to the position from the cam table corresponding to the master axis position	
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
B	*Jerk*	*LREAL*	*[[[Undefined variable Primary.ParameterNotSupported]]] Value of the jerk in [user units / second^3].*	
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.

Reccomended 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

ErrorID	Meaning
0	No error.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4893	The specified external axis may not be used. A physical axis is required.
10113	Incorrect cam table size (check the CamTable.Header.datasize)
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.

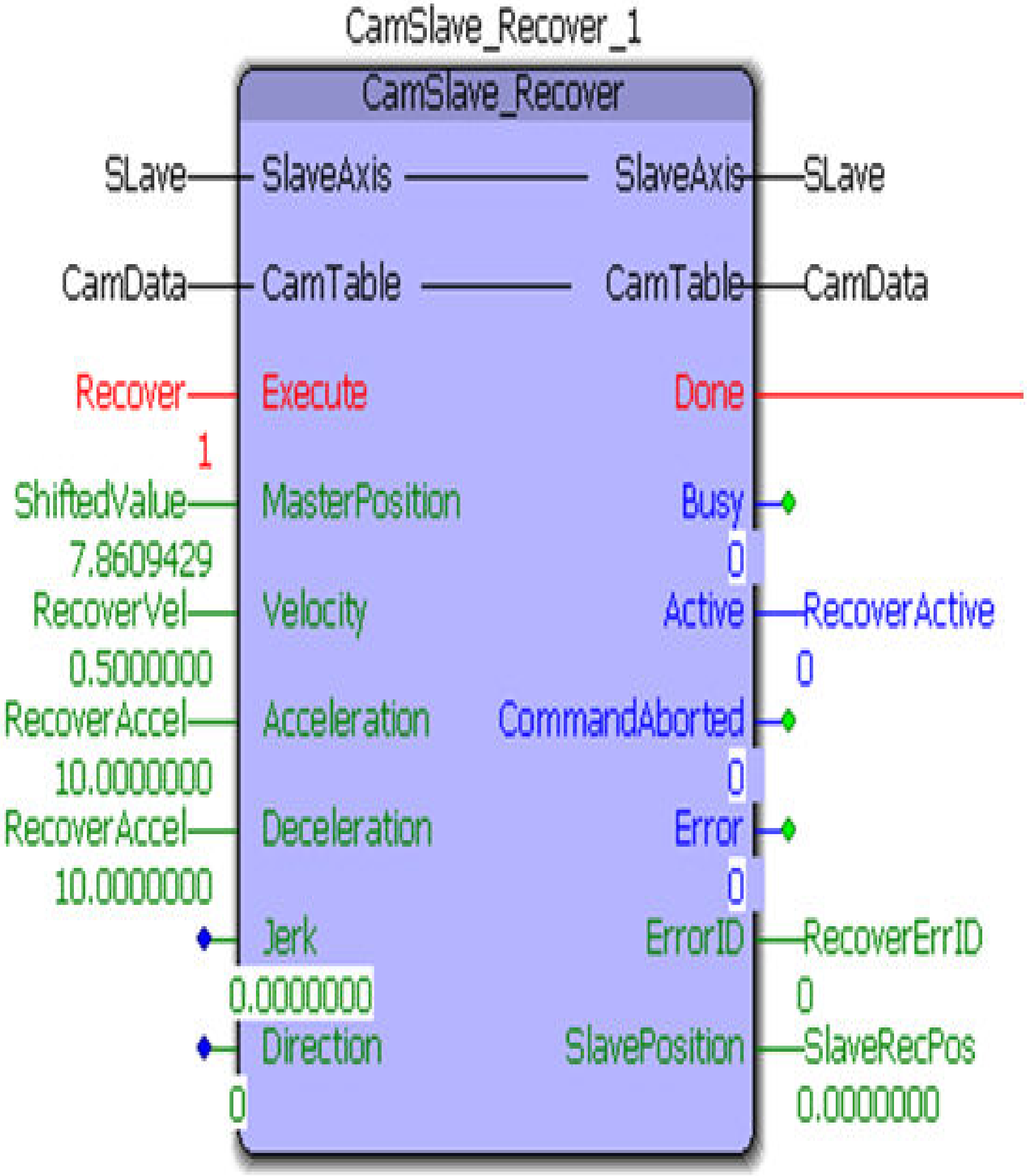
Example

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.



CamSlave_Recover_1

CamSlave_Recover

SLave — SlaveAxis — SlaveAxis — SLave

CamData — CamTable — CamTable — CamData

Recover — Execute — Done

1

ShiftedValue — MasterPosition — Busy

7.8609429 — 0

RecoverVel — Velocity — Active — RecoverActive

0.5000000 — 0

RecoverAccel — Acceleration — CommandAborted

10.0000000 — 0

RecoverAccel — Deceleration — Error

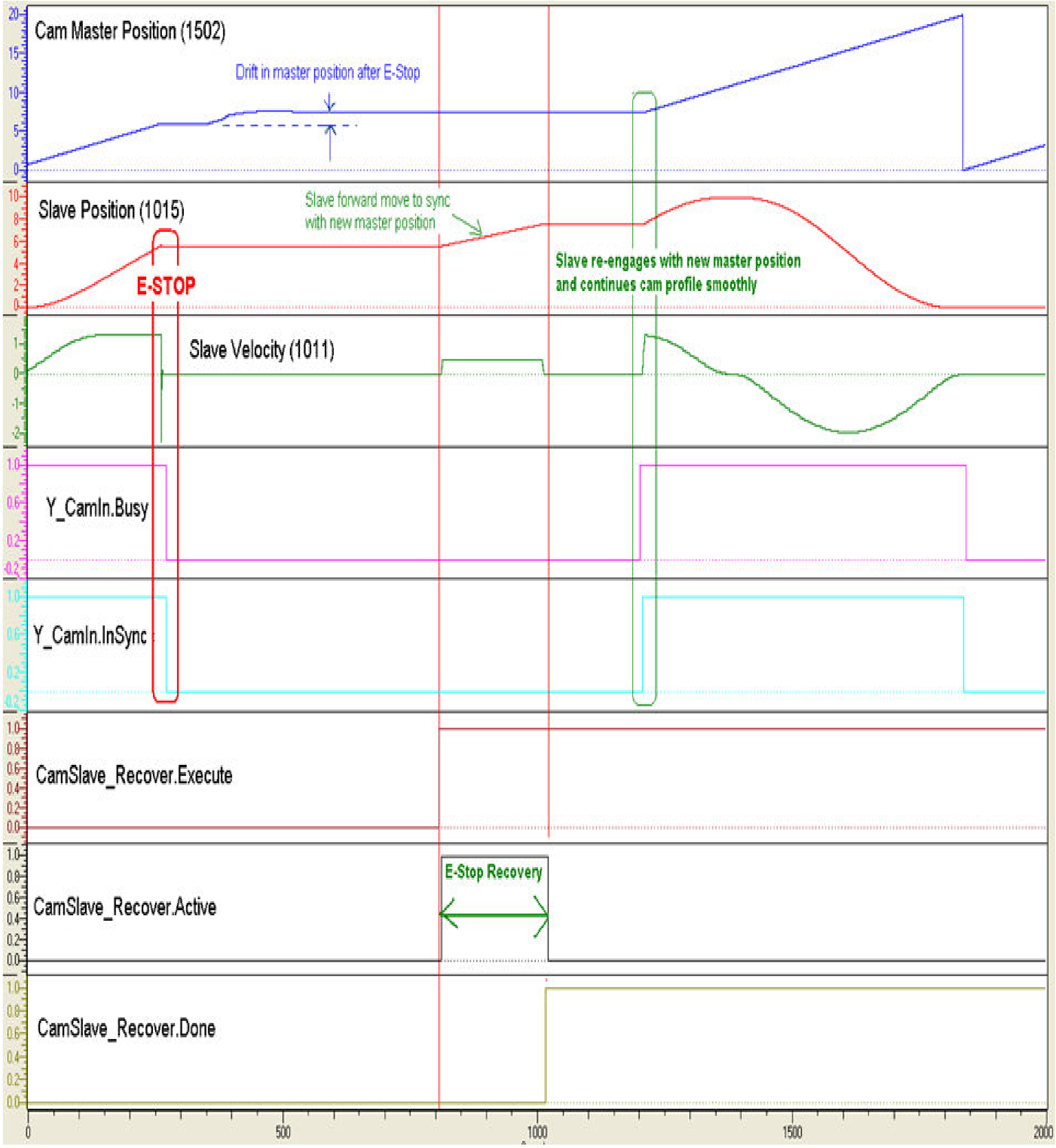
10.0000000 — 0

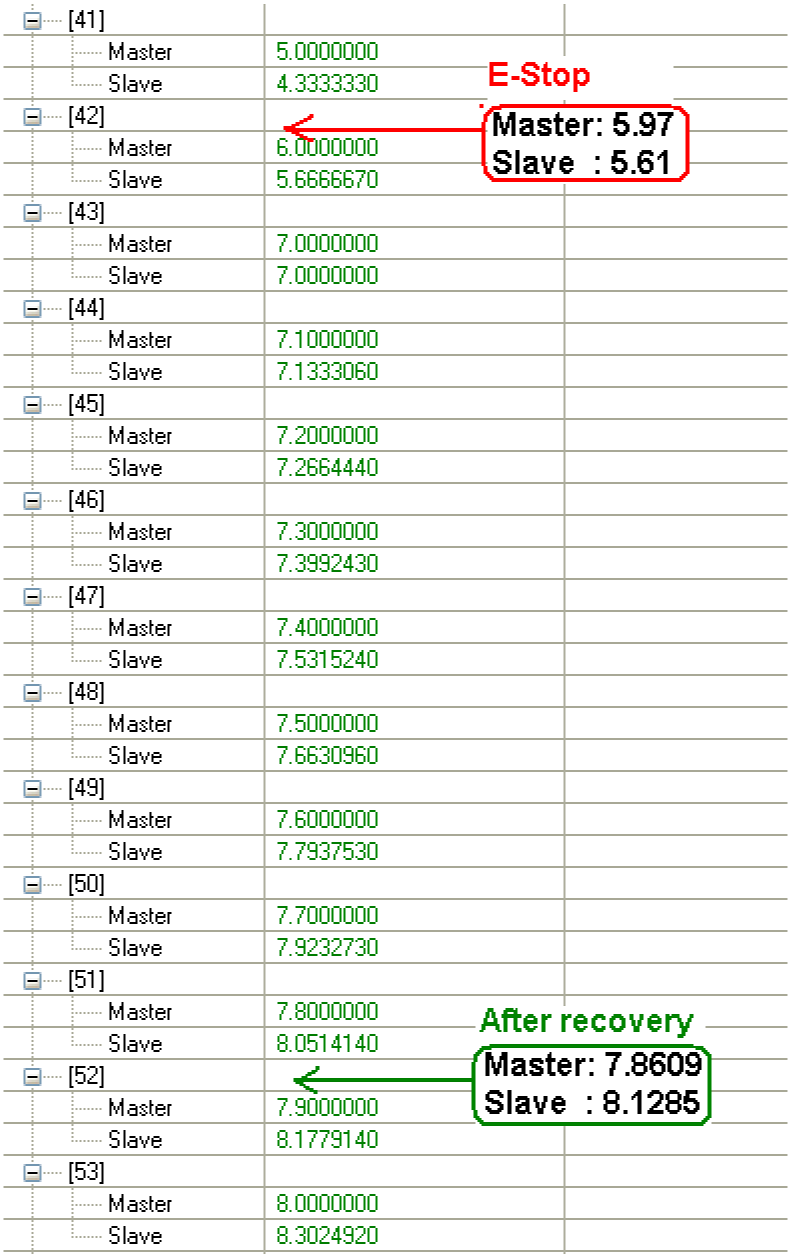
Jerk — ErrorID — RecoverErrID

0.0000000 — 0

Direction — SlavePosition — SlaveRecPos

0 — 0.0000000


Cam Master Position (1502)
Drift in master position after E-Stop
Slave Position (1015)
Slave forward move to sync with new master position
E-STOP
Slave re-engages with new master position and continues cam profile smoothly
Slave Velocity (1011)
Y_CamIn.Busy
Y_CamIn.InSync
CamSlave_Recover.Execute
E-Stop Recovery
CamSlave_Recover.Active
CamSlave_Recover.Done
0
500
1000
1500
2000



[41]	
Master	5.0000000
Slave	4.3333330
[42]	
Master	6.0000000
Slave	5.6666670
[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
Slave	8.0514140
[52]	
Master	7.9000000
Slave	8.1779140
[53]	
Master	8.0000000
Slave	8.3024920

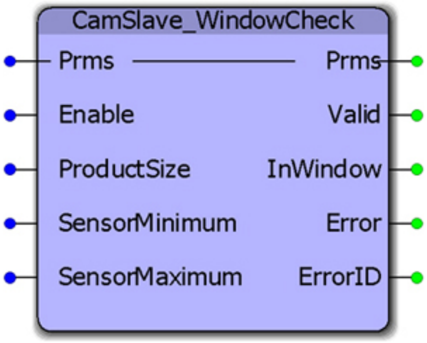
E-Stop

Master: 5.97
Slave : 5.61

After recovery

Master: 7.8609
Slave : 8.1285

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

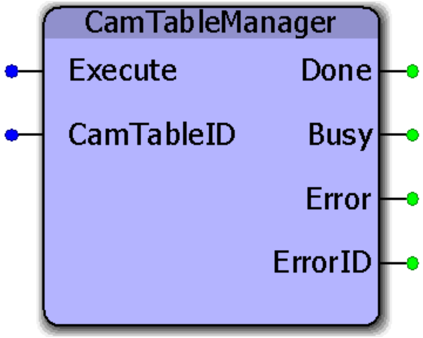
If SensorMinimum and SensorMaximum are both zero, this function does not check for a window and reports InWindow as TRUE.

For the most accurate WindowCheck, this function block must be in a fast application task. Since this function is used by CamSlave_WindowCheck, that block also should be used in a fast (high priority)

Error Description

ErrorID	Meaning
0	No error.
10025	SensorMinimum must be less than SensorMaximum.

CamTableManager



This function block serves as a FIFO buffer for CamTableID's. Each time a new CamTableID is created, it will delete the memory allocated to the oldest CamTable by using the Y_RemoveCamTable function block from the PLCopenPlus firmware library. This function block is used to clean up memory in applications which build 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. The controller can allocate this memory area for new cam tables or application code.

Library

Cam 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
V	CamTableID	UINT	The most recent CamTableID create by Y_CamFileSelect or Y_CamStructSelect	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	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

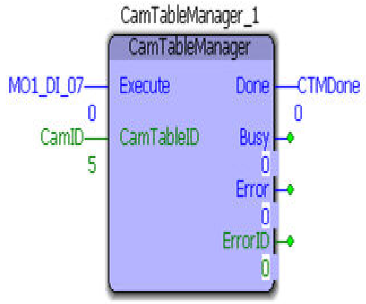
- This function block is unnecessary in applications which use a single, static cam table, or when there are several cams but once they are created, they are used permanently and not recalculated. There is capacity in the controller memory for dozens of cam tables. CamTableManager prevents the situation 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.

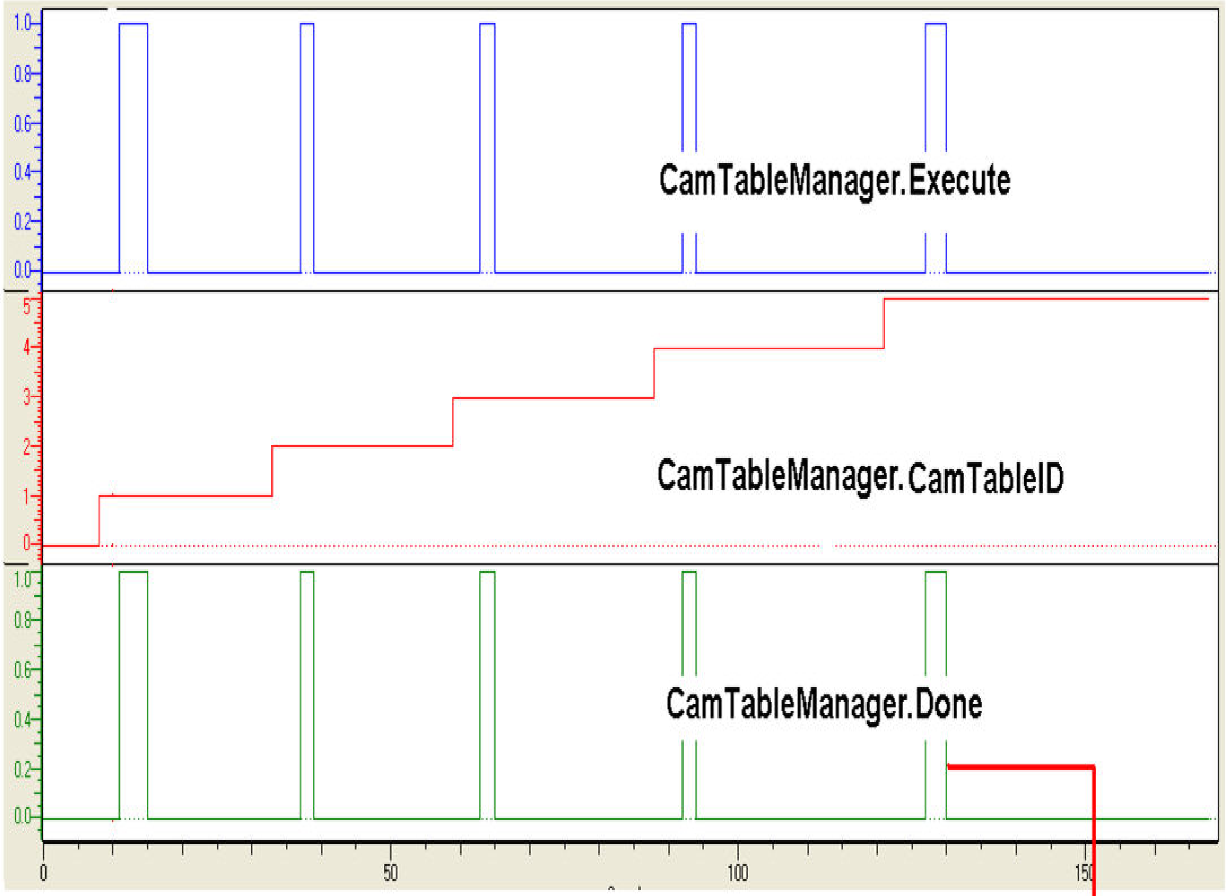
Error Description

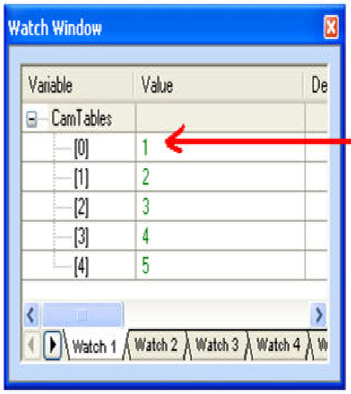
ErrorID	Meaning
0	No error.
4887	CamTableID does not refer to a valid cam table.

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.



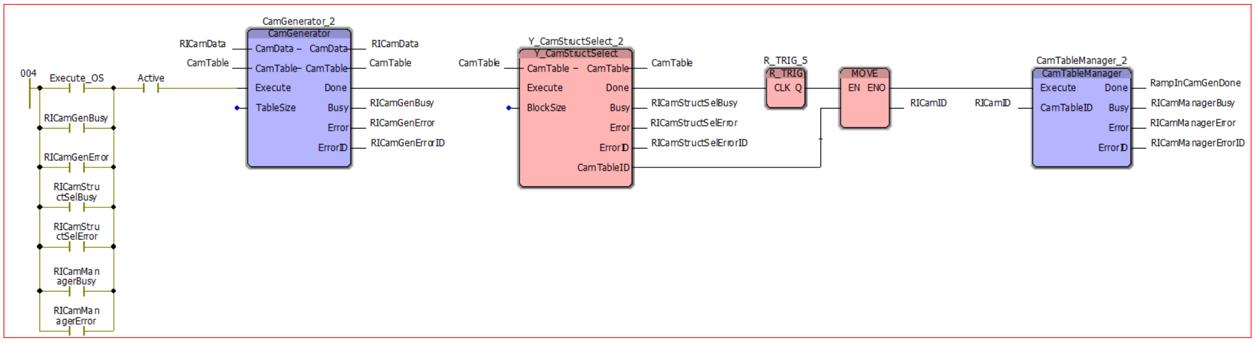




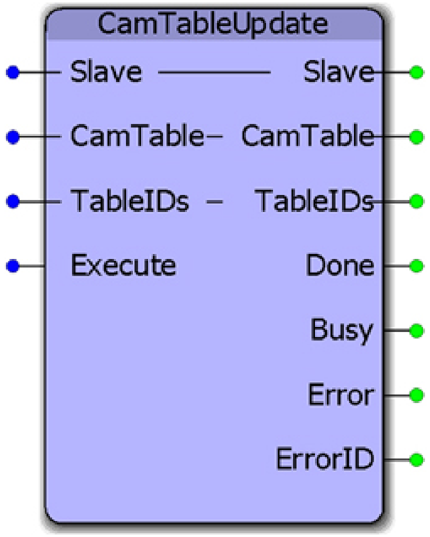
Variable	Value	De
CamTables		
[0]	1	
[1]	2	
[2]	3	
[3]	4	
[4]	5	

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	
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 out-	

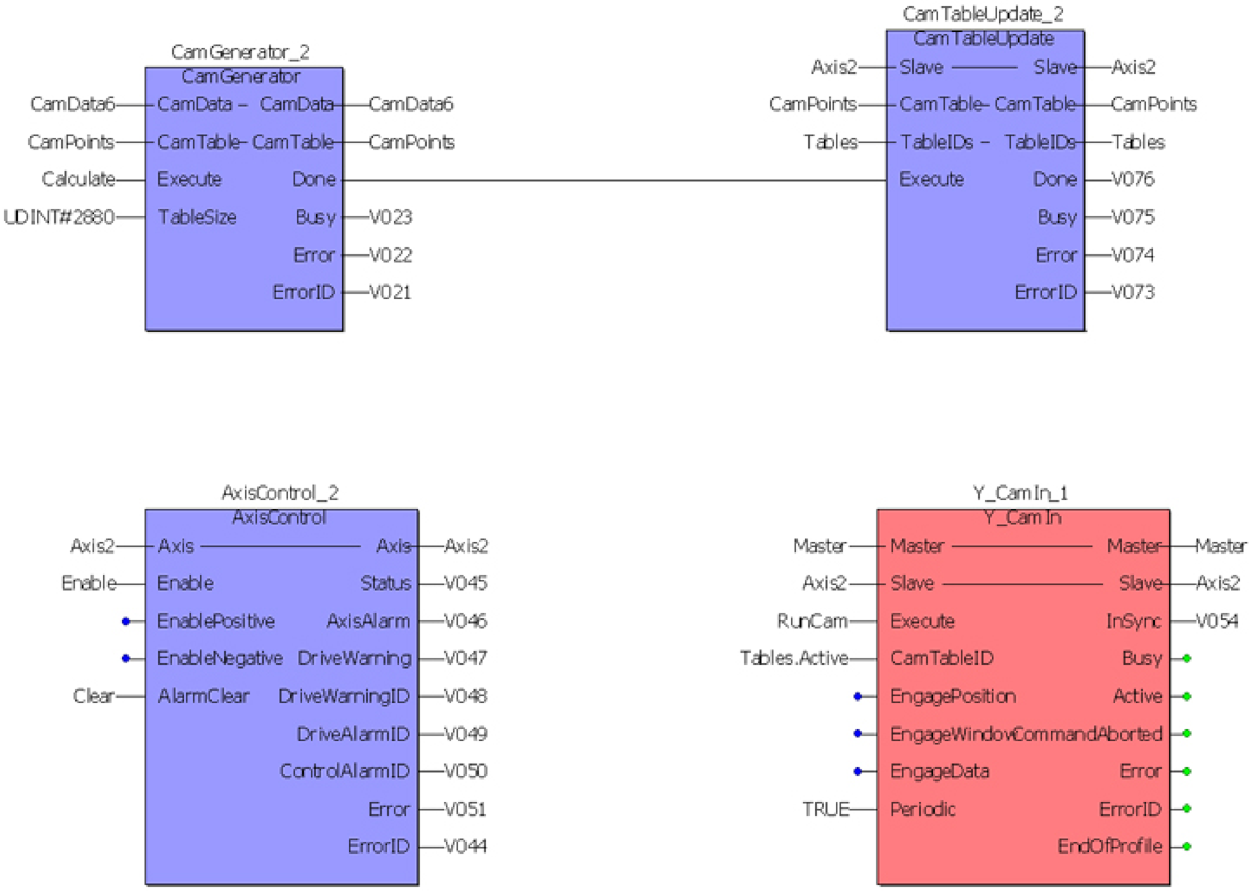
			put is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	BOOL	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

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.

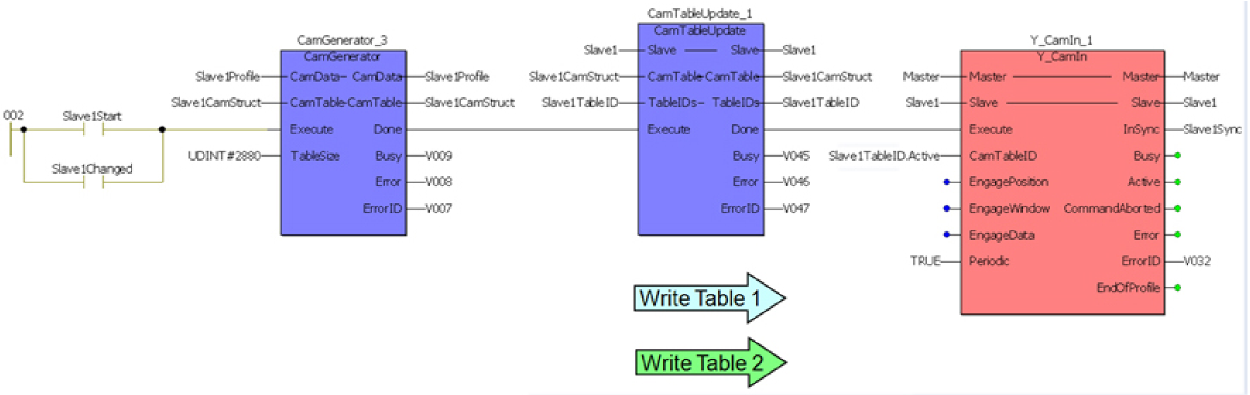
Example 1:

In this example, assume that some event has occurred which triggers the need for a new cam table to be generated using CamGeneator. 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

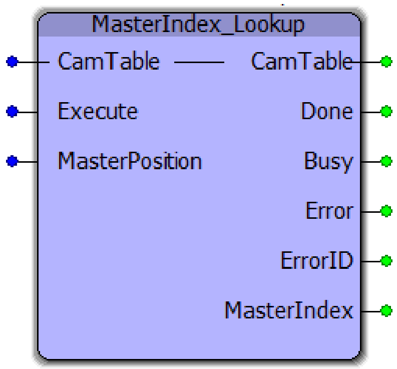
- One will be actively running the motion
- One will be "on deck" to take new changes



Error Description

ErrorID	Meaning
0	No error.
4377	File reading already in progress.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
4387	Already copying cam data (If Execute transition to TRUE while Busy = TRUE).
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.
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.
4887	CamTableID does not refer to a valid cam table.

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	
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.	
B	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.	

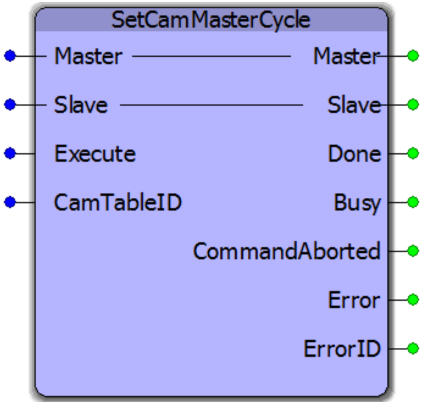
Notes

- 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

ErrorID	Meaning
0	No error.

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

Notes

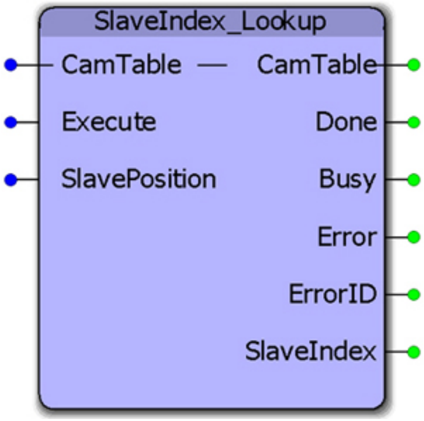
1) Although there will be no slave motion, the slave axis must be enabled using MC_Power before executing this function block.

2) A valid CamTableID must be input before executing SetCamMastercycle.

Error Description

ErrorID	Meaning
0	No error.
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.
4374	Torque move prohibited while non-torque moves queued or in progress.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
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.
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.
4887	CamTableID does not refer to a valid cam table.
4891	The slave axis can not be the same as the master axis.
4893	The specified external axis may not be used. A physical axis is required.
10114	Incorrect cam table size (check the CamTable.Header.Datasize).
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.

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	
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.	
B	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.

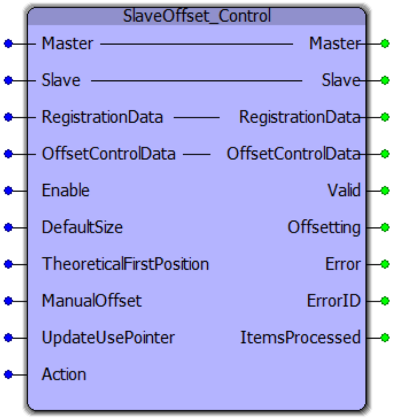
Notes

- 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

ErrorID	Meaning
0	No error.
10045	SlavePosition not found in Y_MS_CAM_STRUCT.

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				**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 pitch between consecutive parts	LREAL#0.0
V	TheoreticalFirstPosition	LREAL	Ideal absolute position of the first part that will be detected by the sensor after homing is done.	LREAL#0.0
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.	LREAL#0.0
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.	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 uleft 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	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.

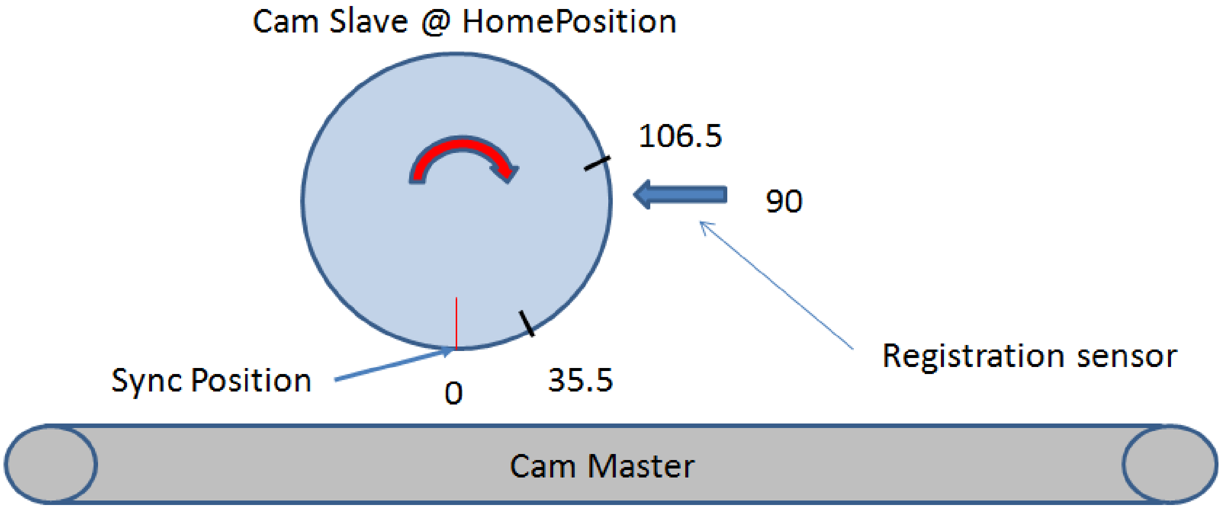
	CamSlave_FeedToLength	**SlaveOffset_Control**
BufferProducts (SensorDistance > 1 part length)	Not supported	Supported
Successive triggers > 2 part lengths	Reports missed latch	Makes large correction
Missed Part Indicator	Supported	Not supported
Registration within window check	Supported	Not supported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

Error Description

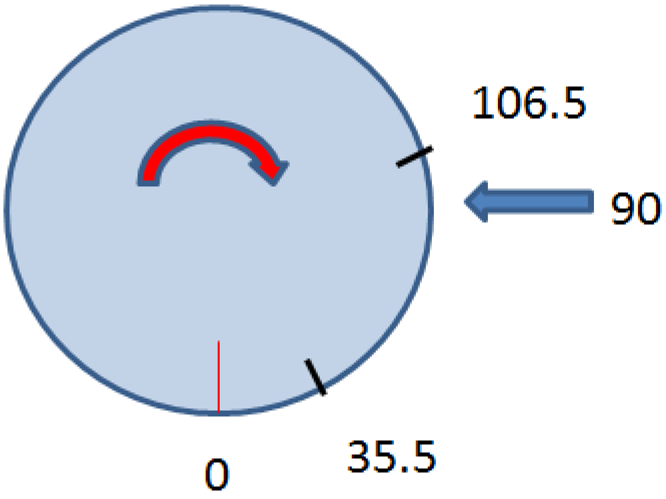
ErrorID	Meaning
0	No error.
11050	Cam correction (shift/offset) has been aborted by another function block.

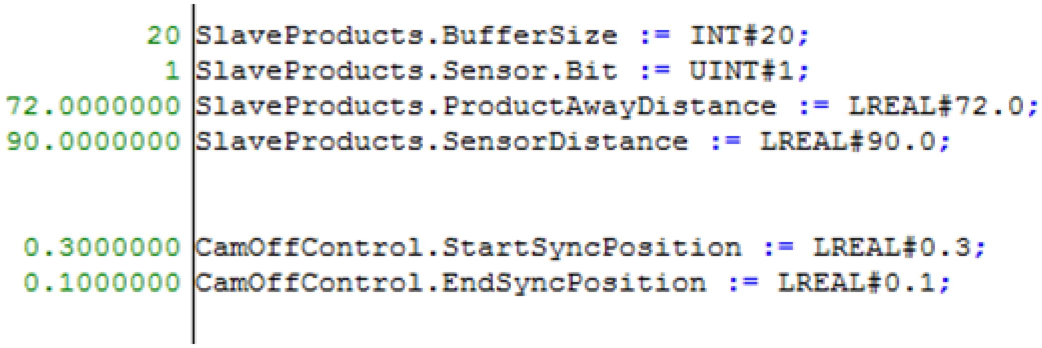
Application Example 1

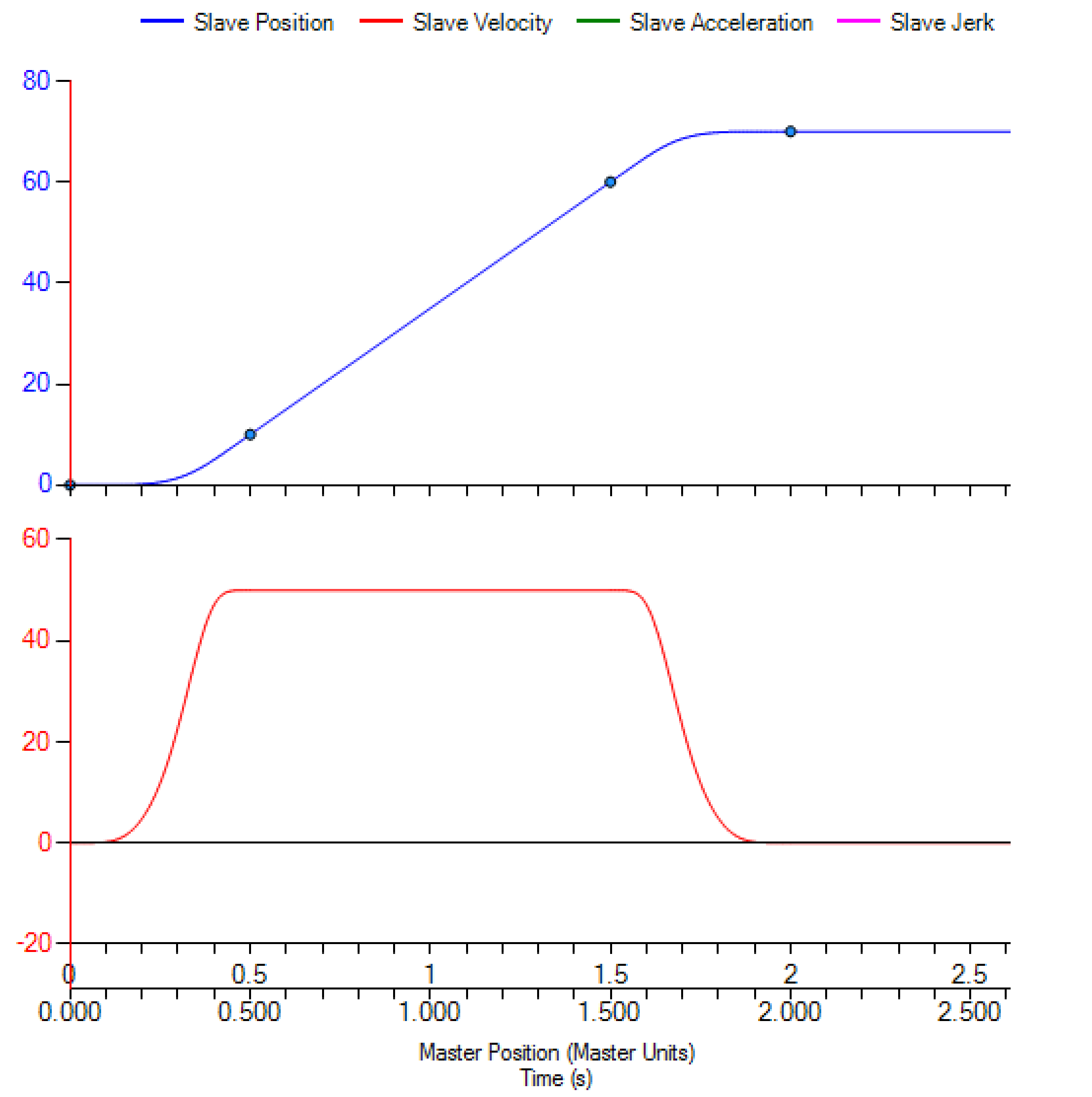
Consider a rotary disc cammed to a master conveyor running a one way cam in the clockwise direction as shown. The home position is defined as shown below. Product length (nominal distance between parts) on the slave is 71 degrees. The nominal cam slave travel is 71 degrees. Position 0 (bottom dead center) 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.



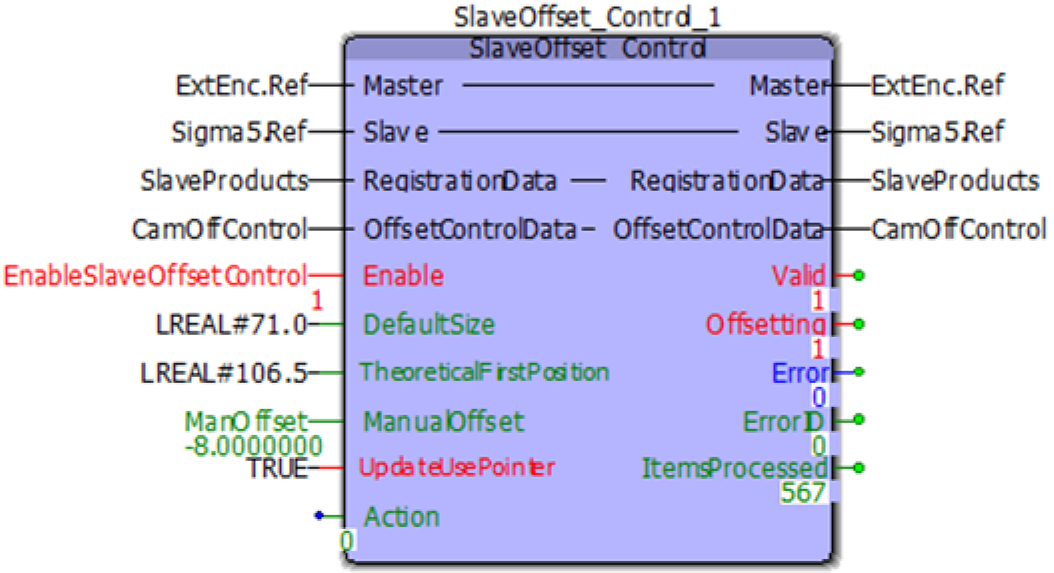
The figure below shows how to configure the SlaveOffsetStruct.




```
       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;
```



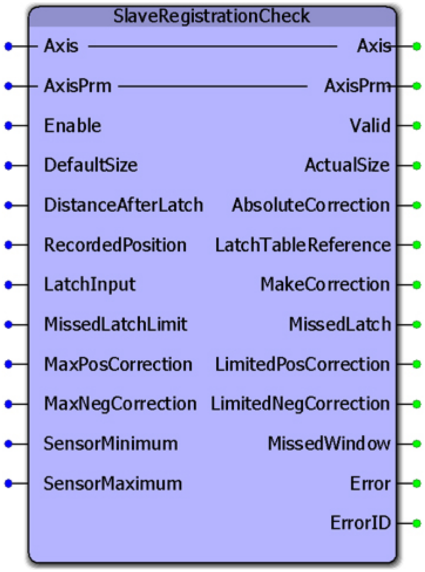
The SlaveOffset_Control block for the application described above can be set up as shown below.



SlaveOffset_Control_1
SlaveOffset_Control

ExtEnc.Ref — Master — Master — ExtEnc.Ref
Sigma5.Ref — Slave — Slave — Sigma5.Ref
SlaveProducts — RegistrationData — RegistrationData — SlaveProducts
CamOffControl — OffsetControlData — OffsetControlData — CamOffControl
EnableSlaveOffsetControl (1) — Enable — Valid (1)
LREAL#71.0 — DefaultSize — Offsetting (1)
LREAL#106.5 — TheoreticalFirstPosition — Error (0)
ManOffset (-8.0000000) — ManualOffset — ErrorID (0)
TRUE — UpdateUsePointer — ItemsProcessed (567)
Action (0)

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
B	DefaultSize	LREAL	Default length of the product in user units.	LREAL#0.0
B	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
B	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#0.0 (function block sets SensorMaxium 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 positons.	
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.	
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 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

ErrorID	Meaning
0	No error.
4377	File reading already in progress.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
4387	Already copying cam data (This error occurs if Execute transitions to TRUE while Busy is already TRUE).
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.
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.
4887	CamTableID does not refer to a valid cam table.
10020	ProductSize cannot be less than or equal to zero.
10021	Maximum allowed consecutive missed registration marks reached.
10086	MaxPosCorrection must be zero or positive, MaxNegCorrection must be or zero or negative.

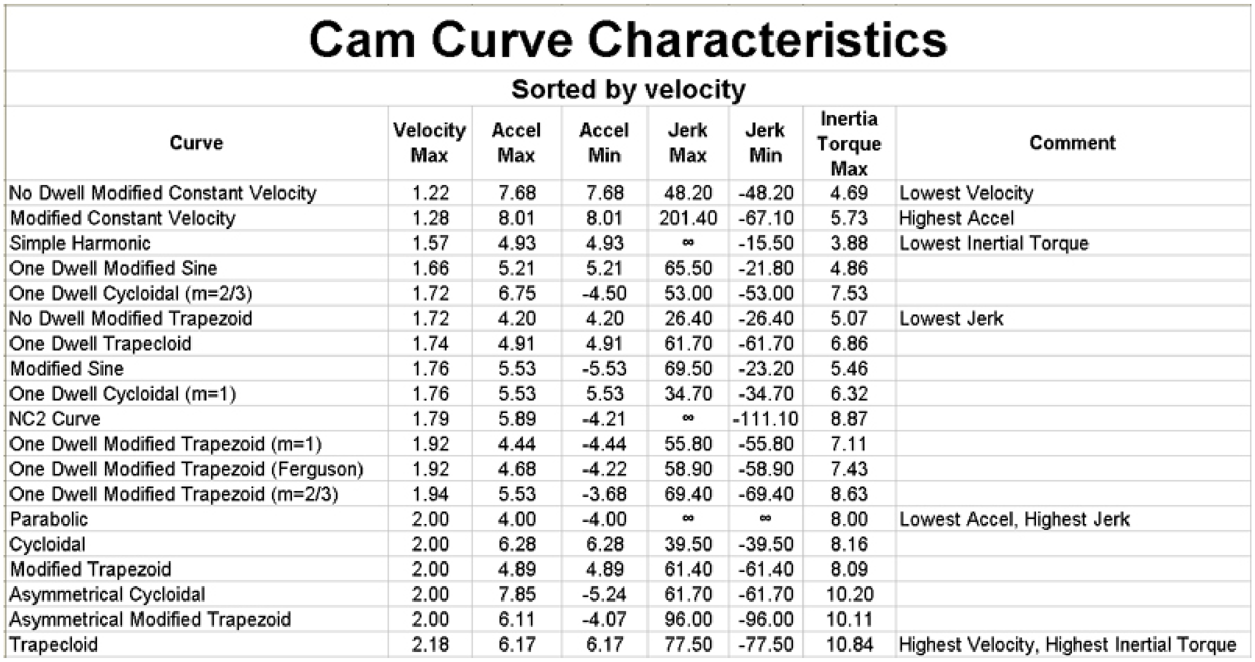
Cam Curve Characteristics

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 Vm, Am, and Jm 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:

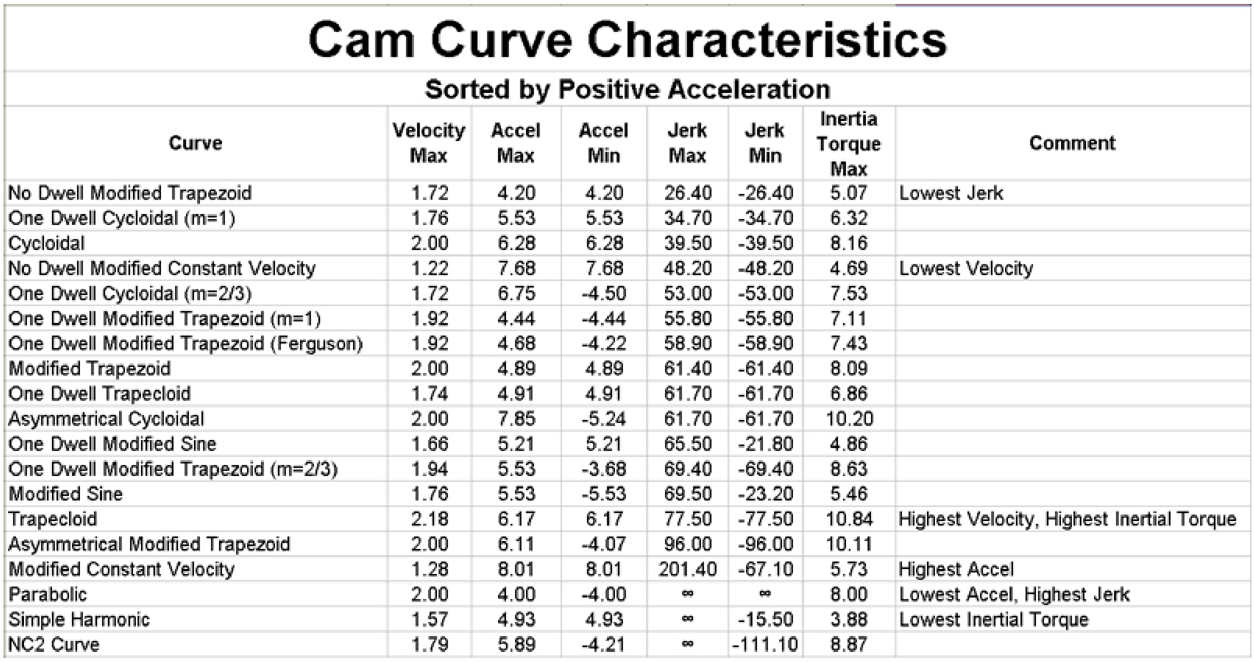
1. Velocity V and Acceleration A are to be continuous
2. Low values of Vm and Qm are needed in low speed and heavy load applications.
3. Low values of Am and Jm are needed in high speed and light load applications.
4. Asymmetrical curve having the longer period of deceleration than acceleration should 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

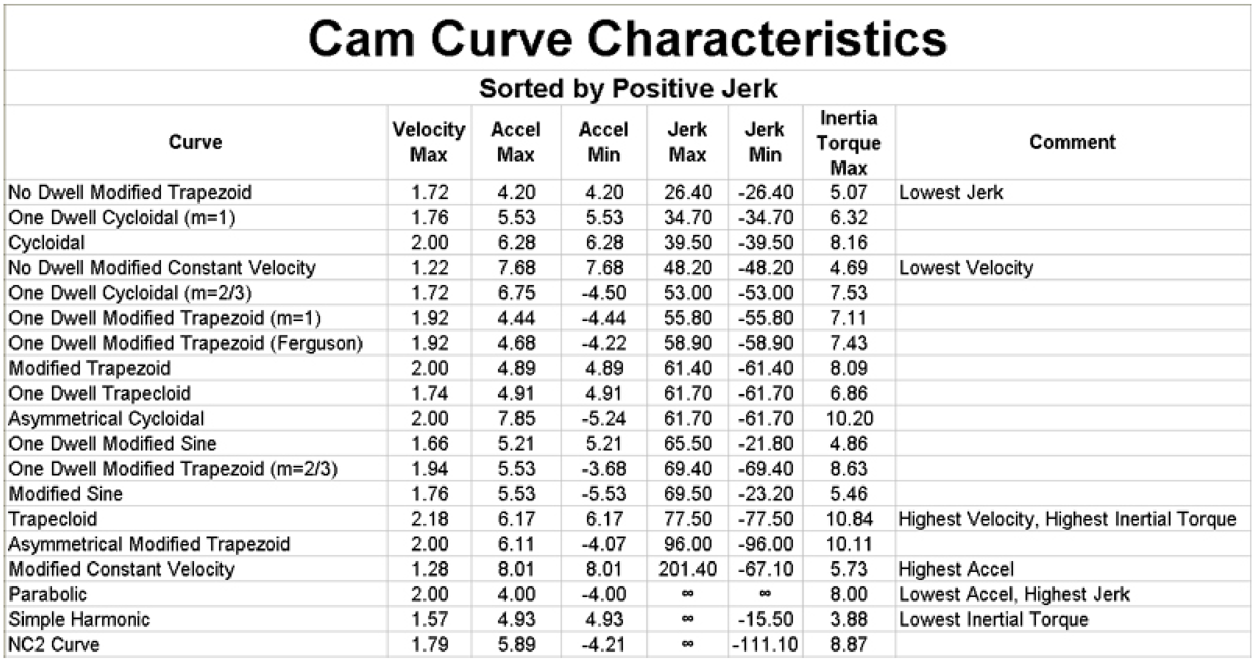
Curve	Velocity Max	Accel Max	Accel Min	Jerk Max	Jerk Min	Inertia Torque Max	Comment
No Dwell Modified Constant Velocity	1.22	7.68	7.68	48.20	-48.20	4.69	Lowest Velocity
Modified Constant Velocity	1.28	8.01	8.01	201.40	-67.10	5.73	Highest Accel
Simple Harmonic	1.57	4.93	4.93	∞	-15.50	3.88	Lowest Inertial 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

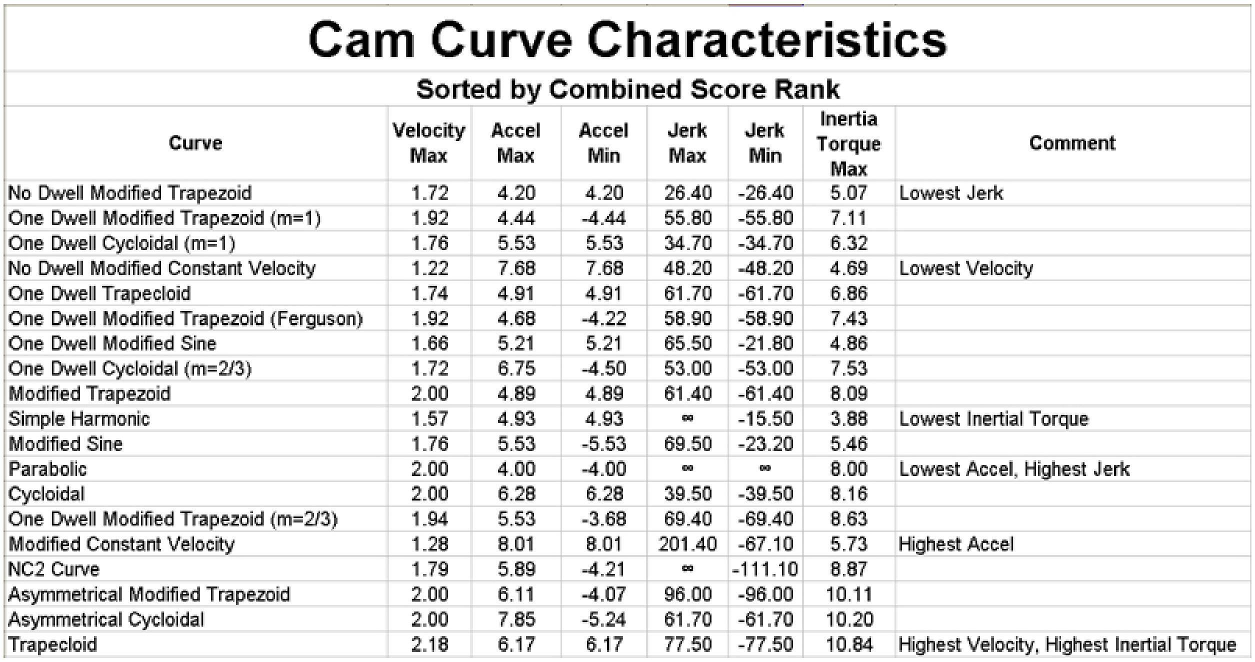
Curve	Velocity Max	Accel Max	Accel Min	Jerk Max	Jerk Min	Inertia Torque Max	Comment
No Dwell Modified Trapezoid	1.72	4.20	4.20	26.40	-26.40	5.07	Lowest Jerk
One Dwell Cycloidal (m=1)	1.76	5.53	5.53	34.70	-34.70	6.32	
Cycloidal	2.00	6.28	6.28	39.50	-39.50	8.16	
No Dwell Modified Constant Velocity	1.22	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 Trapecloid	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	
Trapecloid	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 Trapecloid	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	
Trapecloid	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	



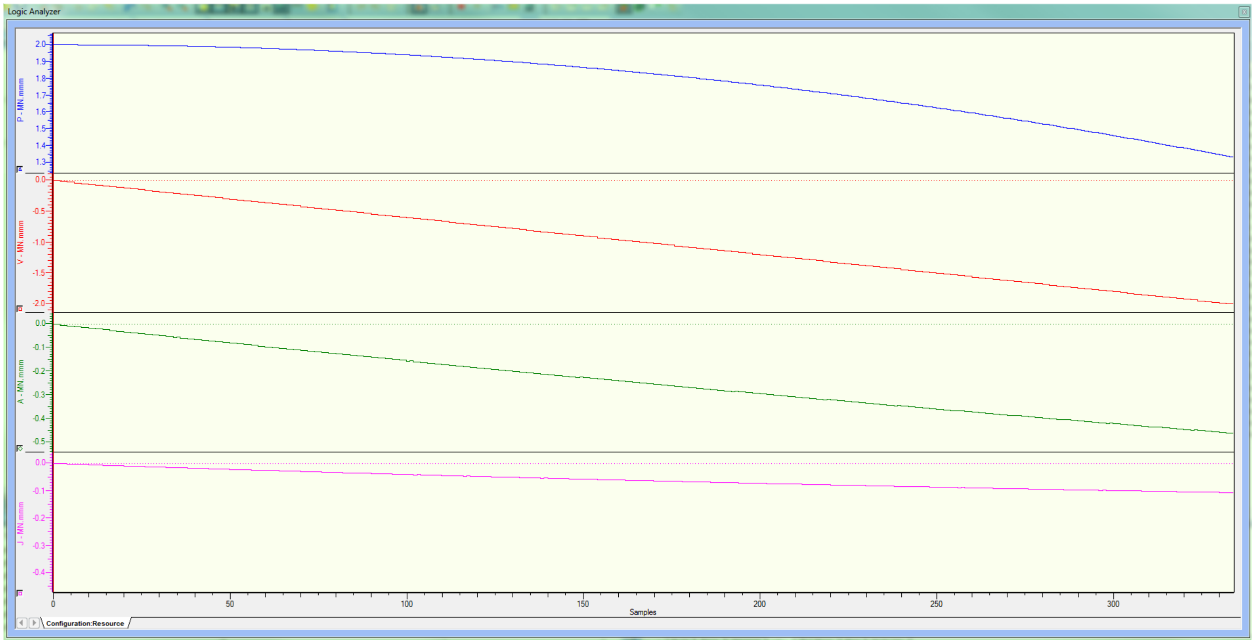
Cam Curve Characteristics

Sorted by Combined Score Rank

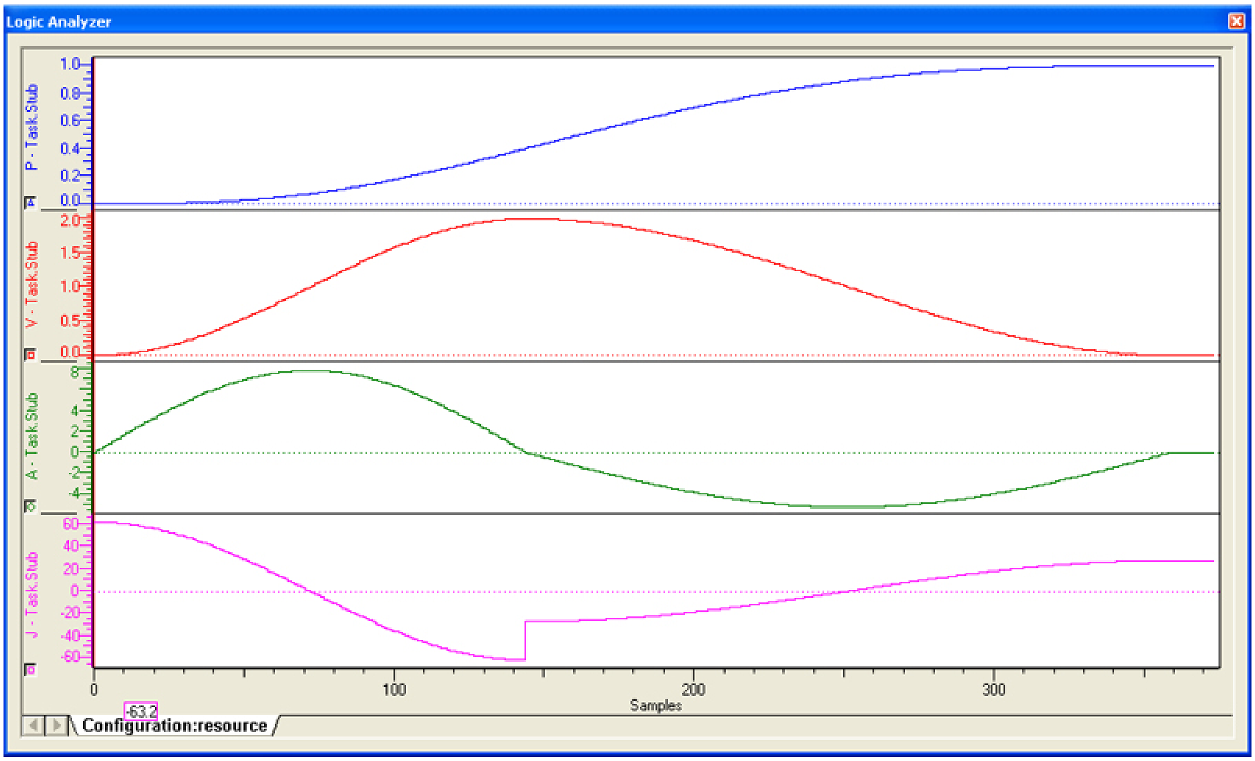
Curve	Velocity Max	Accel Max	Accel Min	Jerk Max	Jerk Min	Inertia Torque Max	Comment
No Dwell Modified Trapezoid	1.72	4.20	4.20	26.40	-26.40	5.07	Lowest Jerk
One Dwell Modified Trapezoid (m=1)	1.92	4.44	-4.44	55.80	-55.80	7.11	
One Dwell Cycloidal (m=1)	1.76	5.53	5.53	34.70	-34.70	6.32	
No Dwell Modified 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

Arc

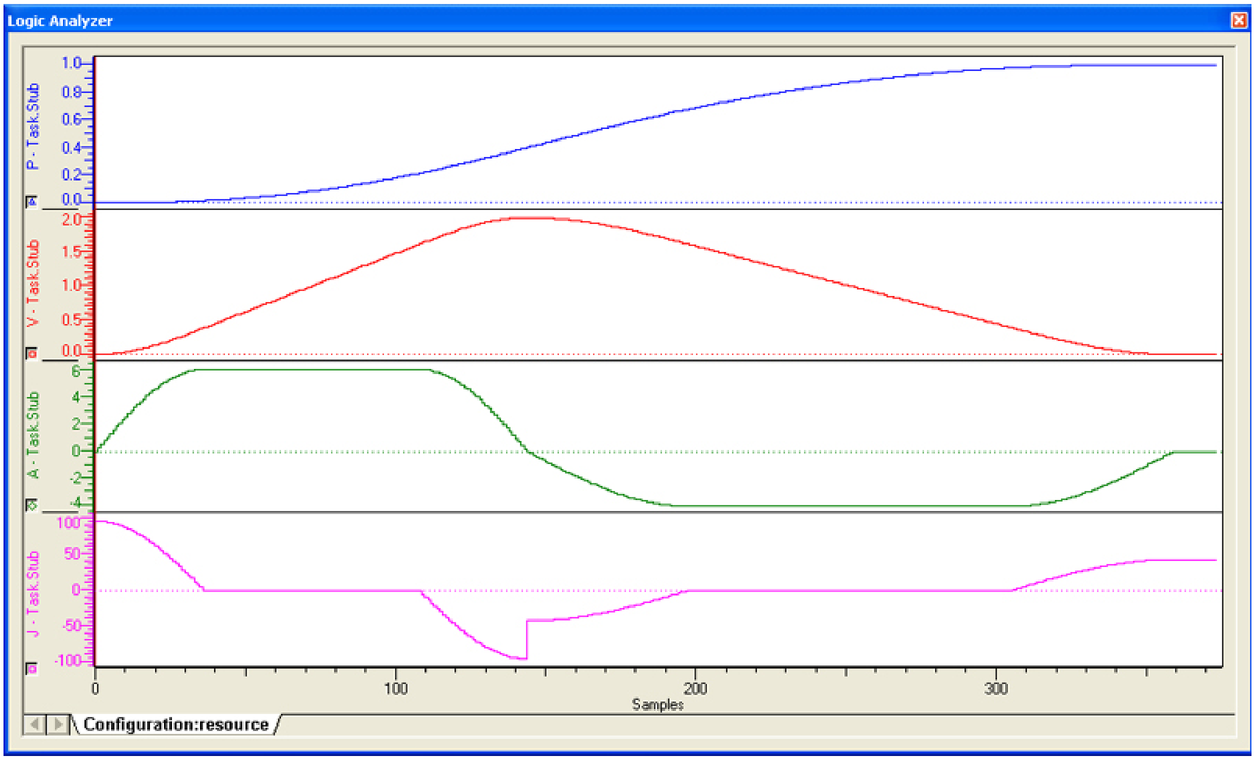
The CamSegmentStruct elements ArcRadius and ArcDirection must be declared for proper usage of this curve type.



Asymmetrical Cycloidal

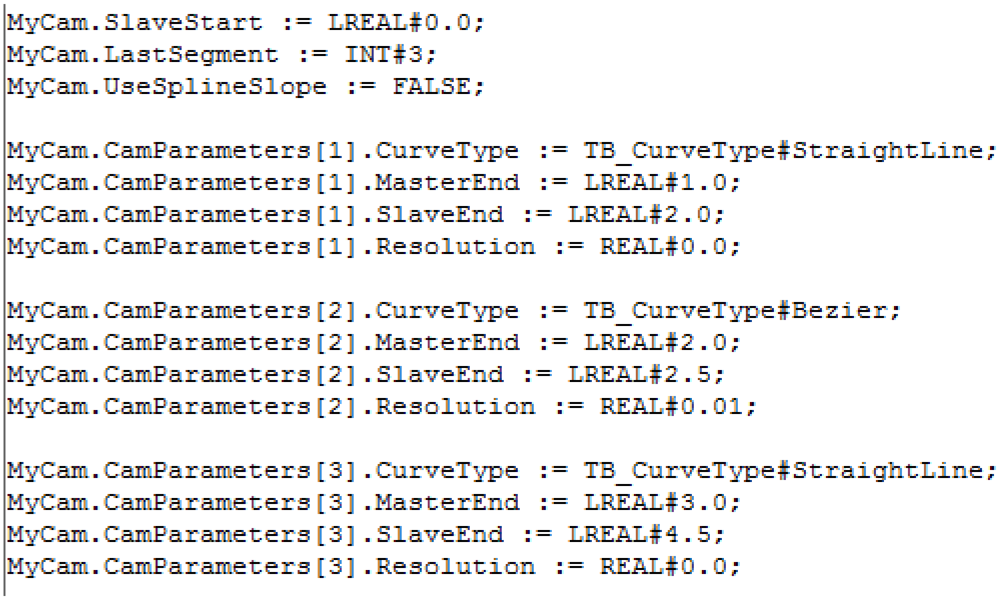


Asymmetrical Modified Trapezoid

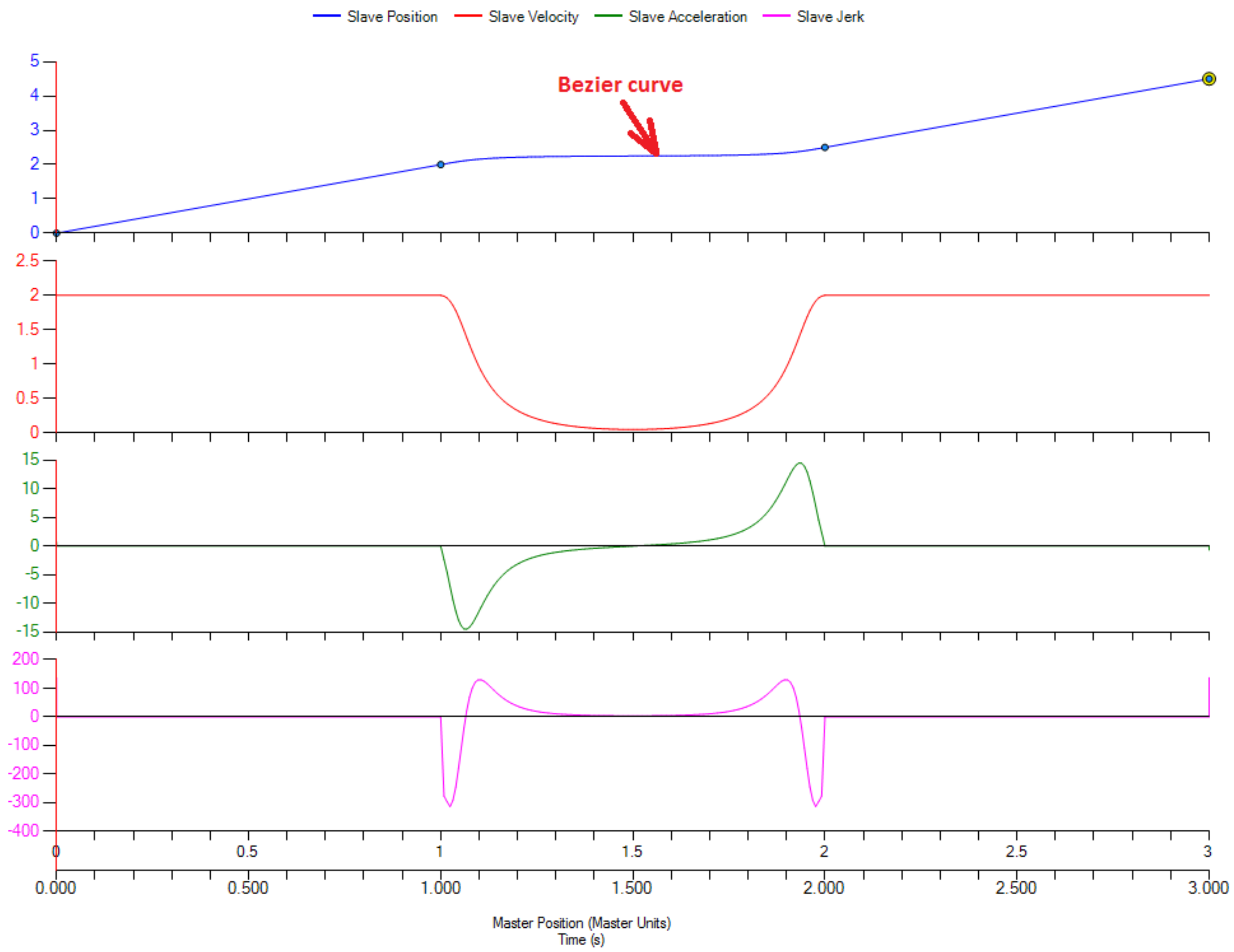


Bezier

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

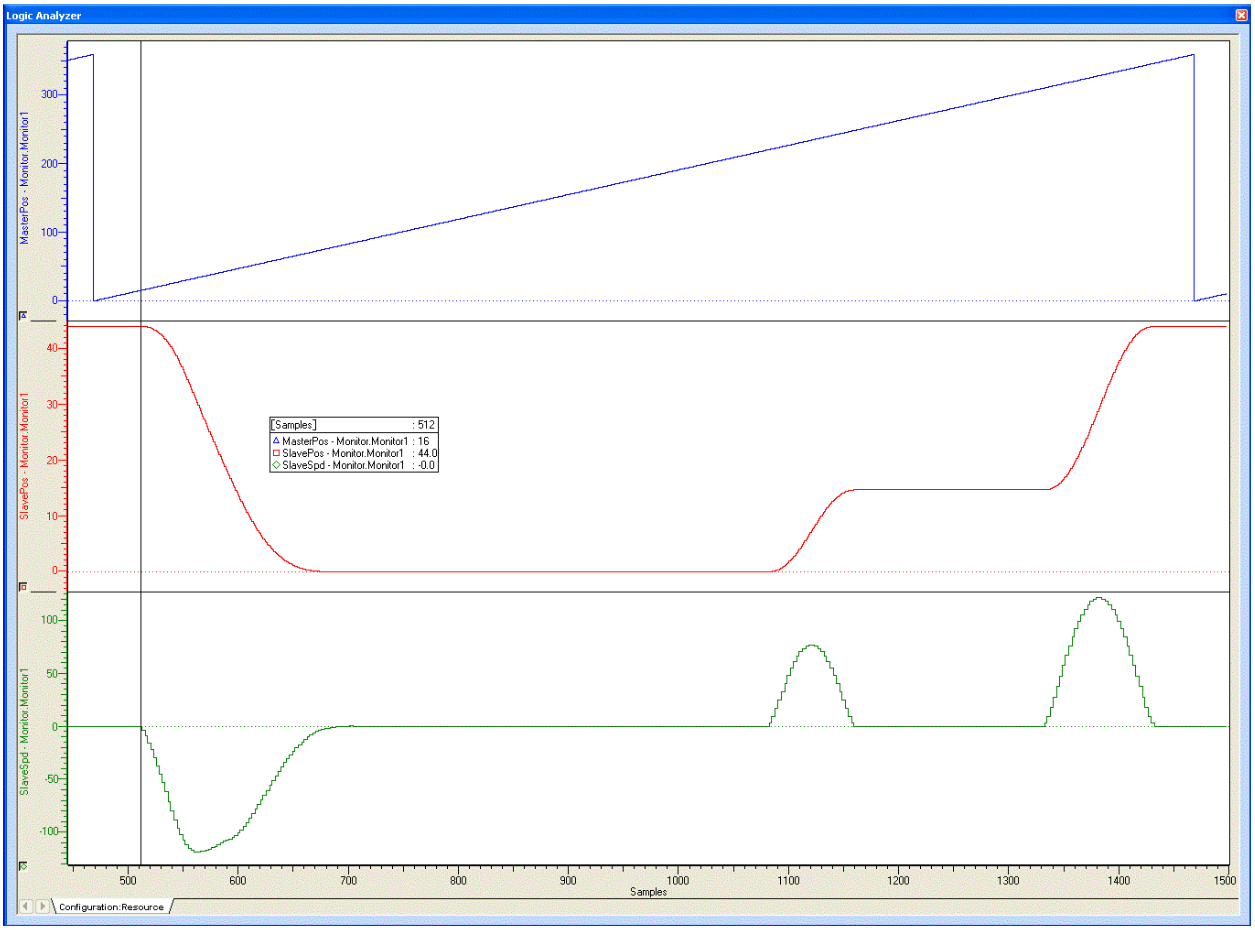


```
MyCam.SlaveStart := LREAL#0.0;
MyCam.LastSegment := INT#3;
MyCam.UseSplineSlope := FALSE;

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;
```



Slave Position
Slave Velocity
Slave Acceleration
Slave Jerk
Bezier curve
5
4
3
2
1
0
2.5
2
1.5
1
0.5
0
15
10
5
0
-5
-10
-15
200
100
0
-100
-200
-300
-400
0
0.5
1
1.5
2
2.5
3
0.000
0.500
1.000
1.500
2.000
2.500
3.000
Master Position (Master Units)
Time (s)

Cubic Spline


Logic Analyzer
MasterPos - Monitor.Monitor1
300
200
100
0
SlavePos - Monitor.Monitor1
40
30
20
10
0
SlaveSpd - Monitor.Monitor1
100
50
0
-50
-100
500
600
700
800
900
1000
1100
1200
1300
1400
1500
Samples
[Samples] : 512
△ MasterPos - Monitor.Monitor1 : 16
□ SlavePos - Monitor.Monitor1 : 44.0
◇ SlaveSpd - Monitor.Monitor1 : -0.0
Configuration:Resource

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.

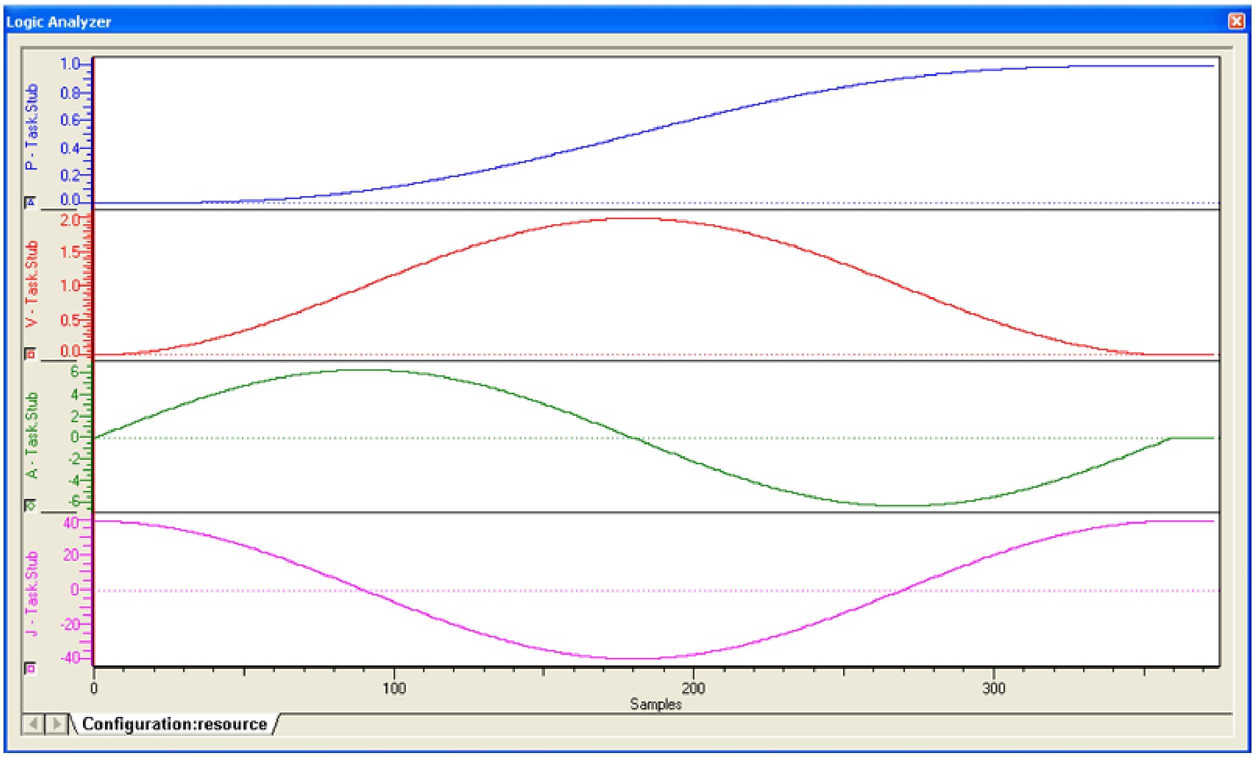
The CamData values are shown below:

```
(*  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;
```

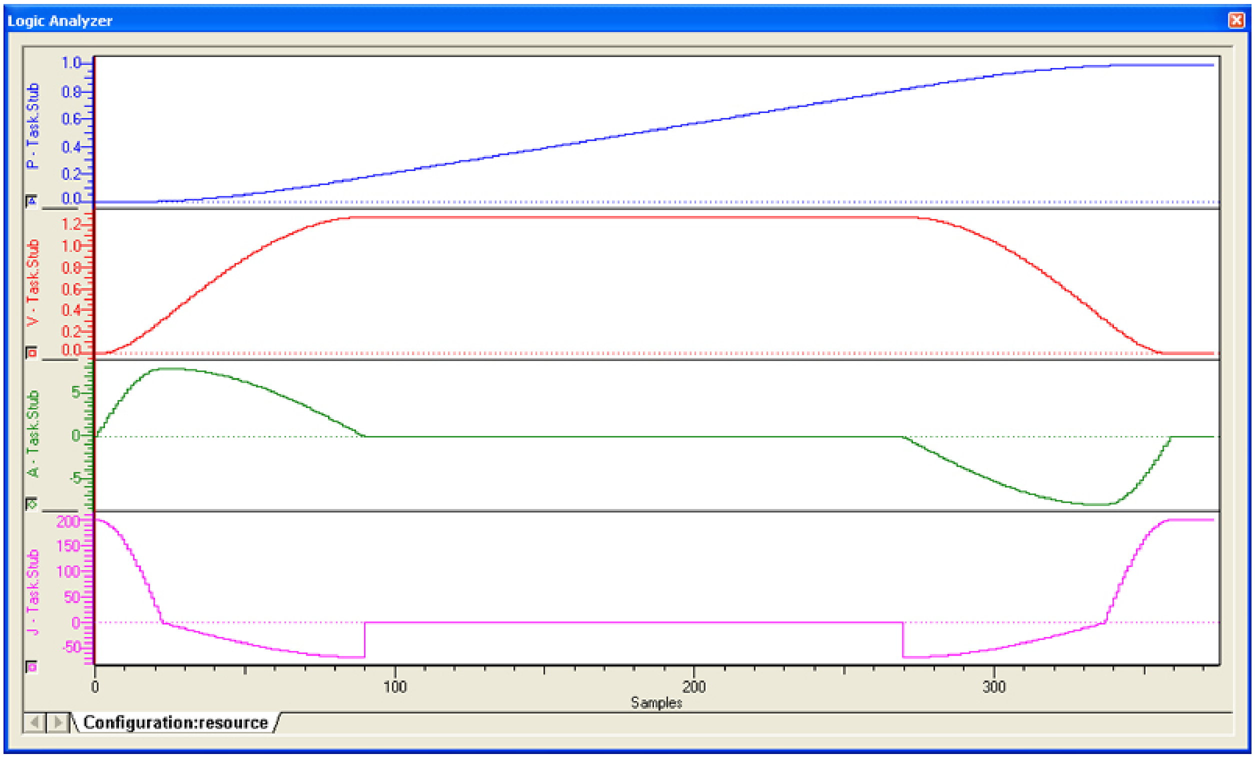
Cycloidal


Logic Analyzer
P - Task.Stub
1.0
0.8
0.6
0.4
0.2
0.0
V - Task.Stub
2.0
1.5
1.0
0.5
0.0
A - Task.Stub
6
4
2
0
-2
-4
-6
J - Task.Stub
40
20
0
-20
-40
0
100
200
300
Samples
Configuration:resource

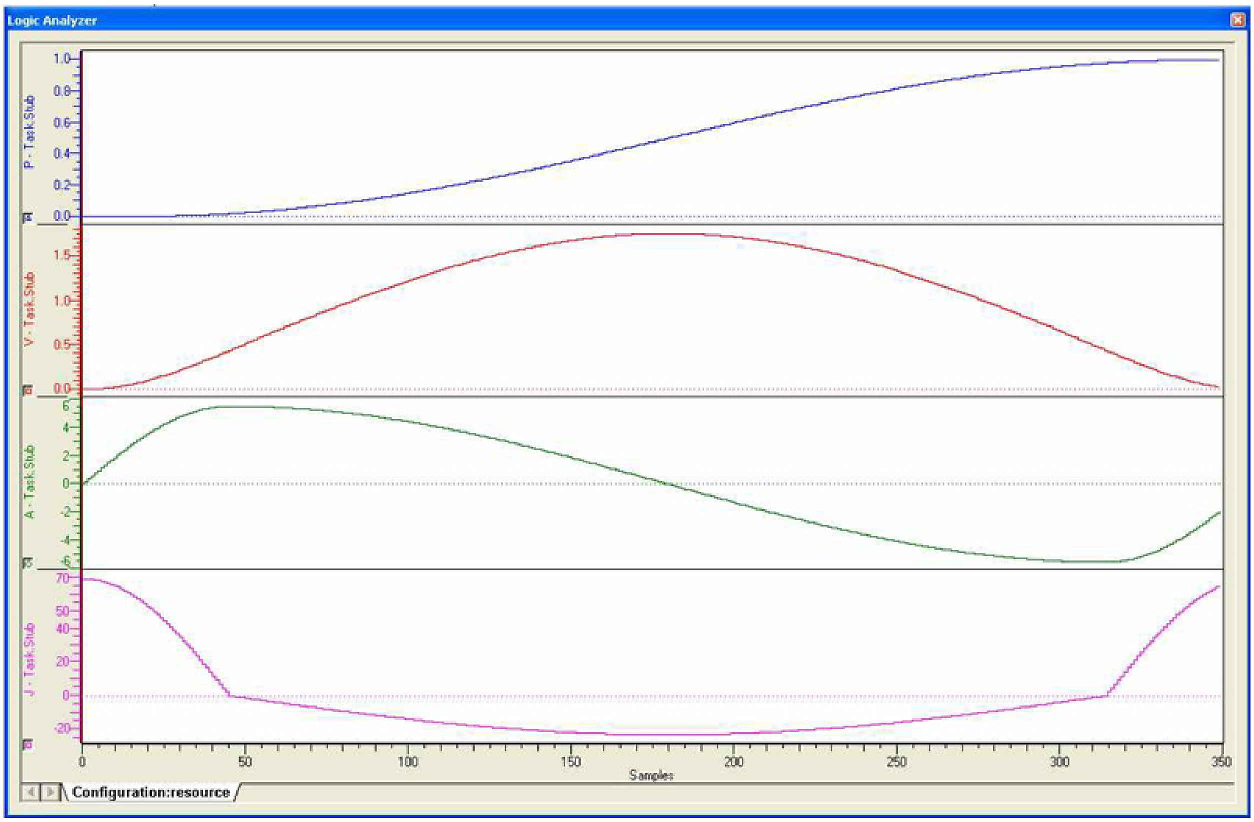
Double Harmonic

This curve type is not supported.

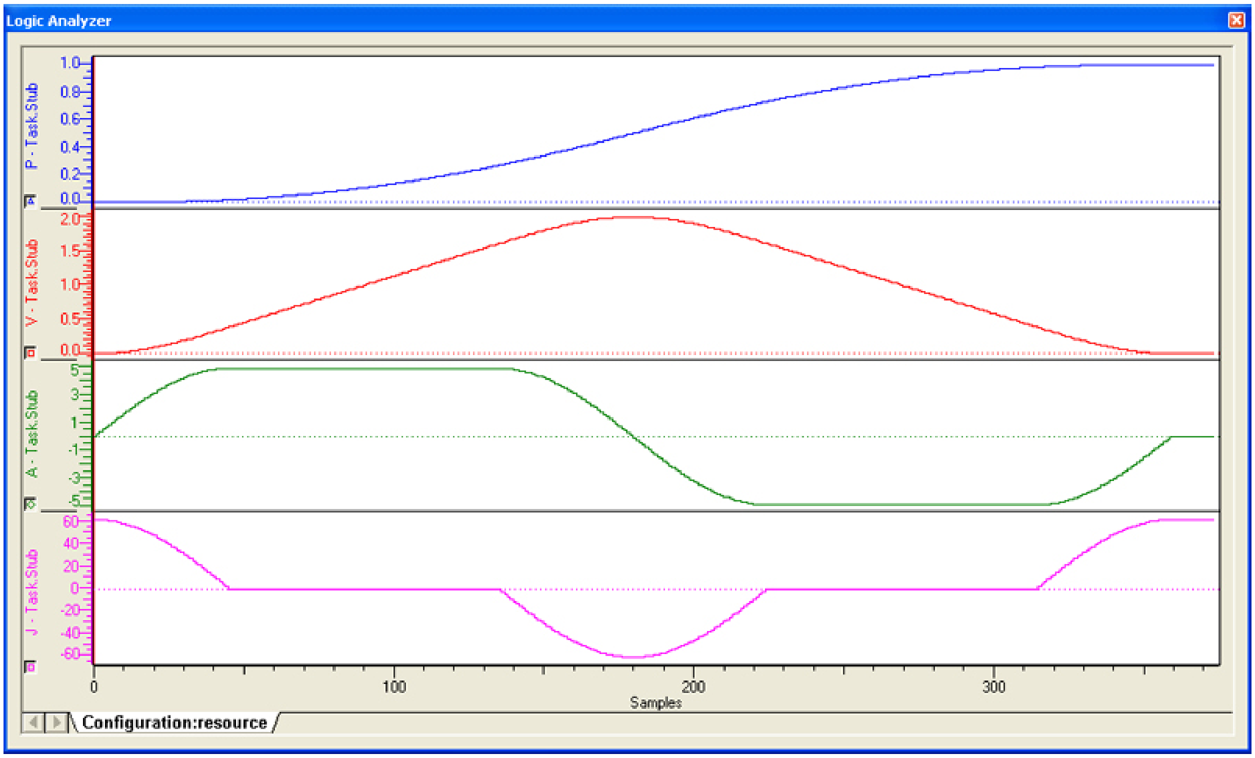
Modified Constant Velocity



Modified Sine

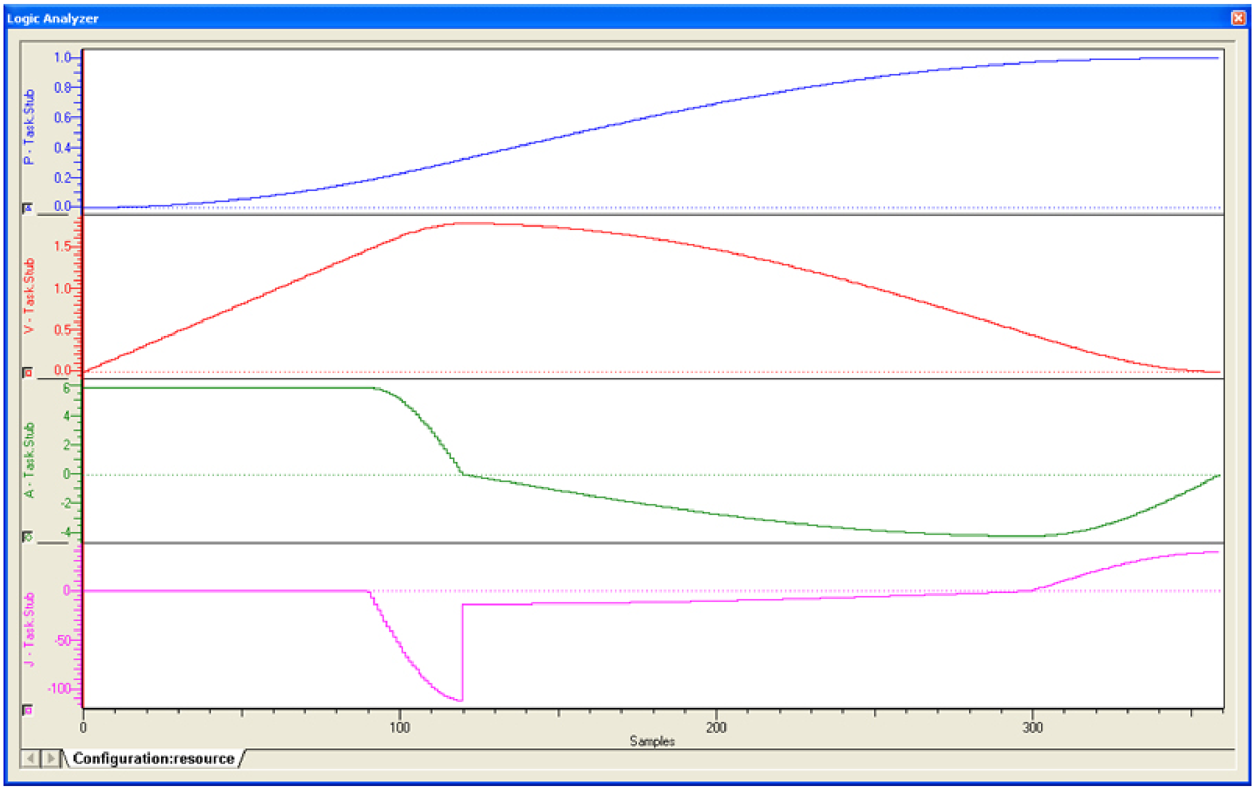


Modified Trapezoid

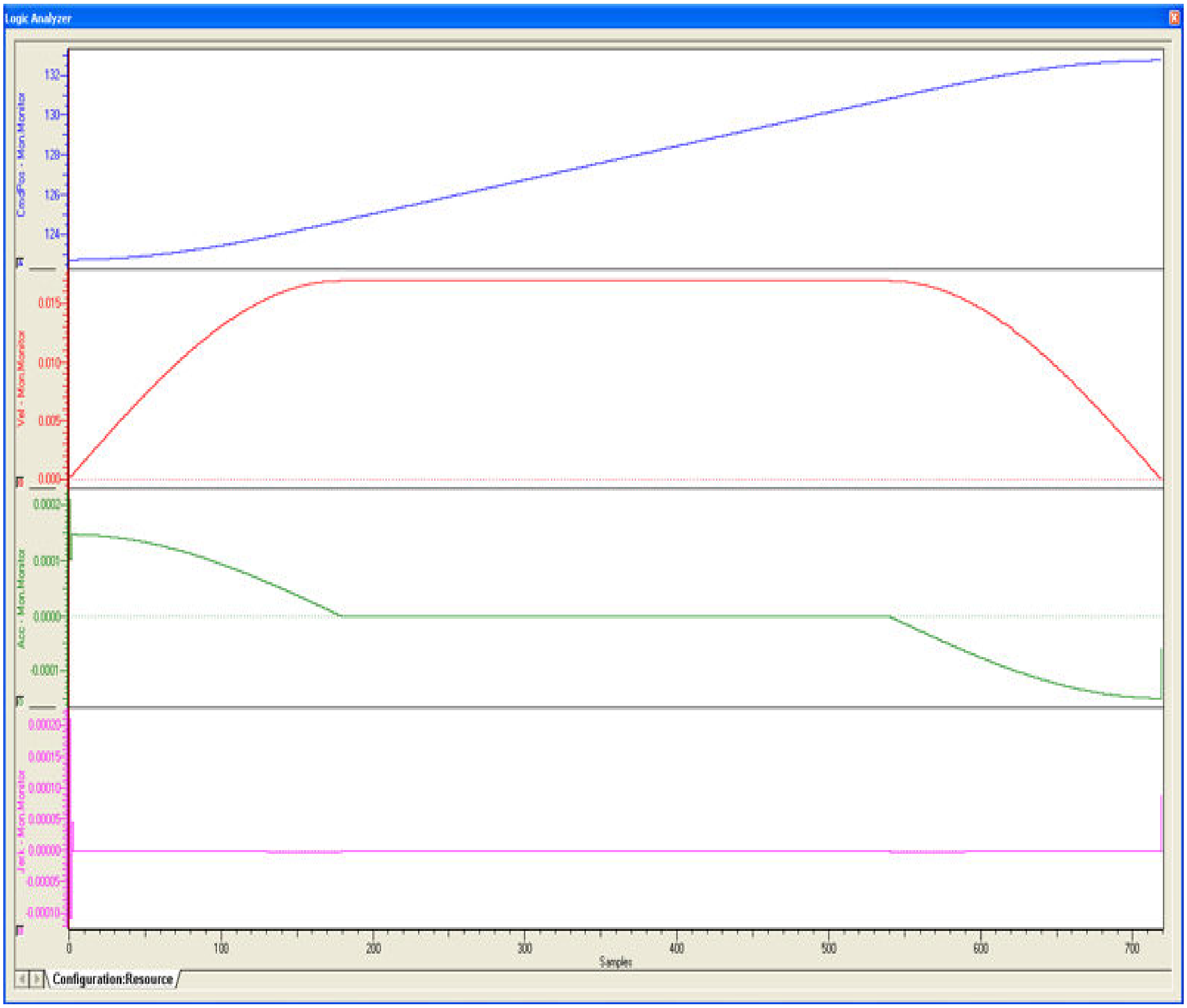


NC2 Curve

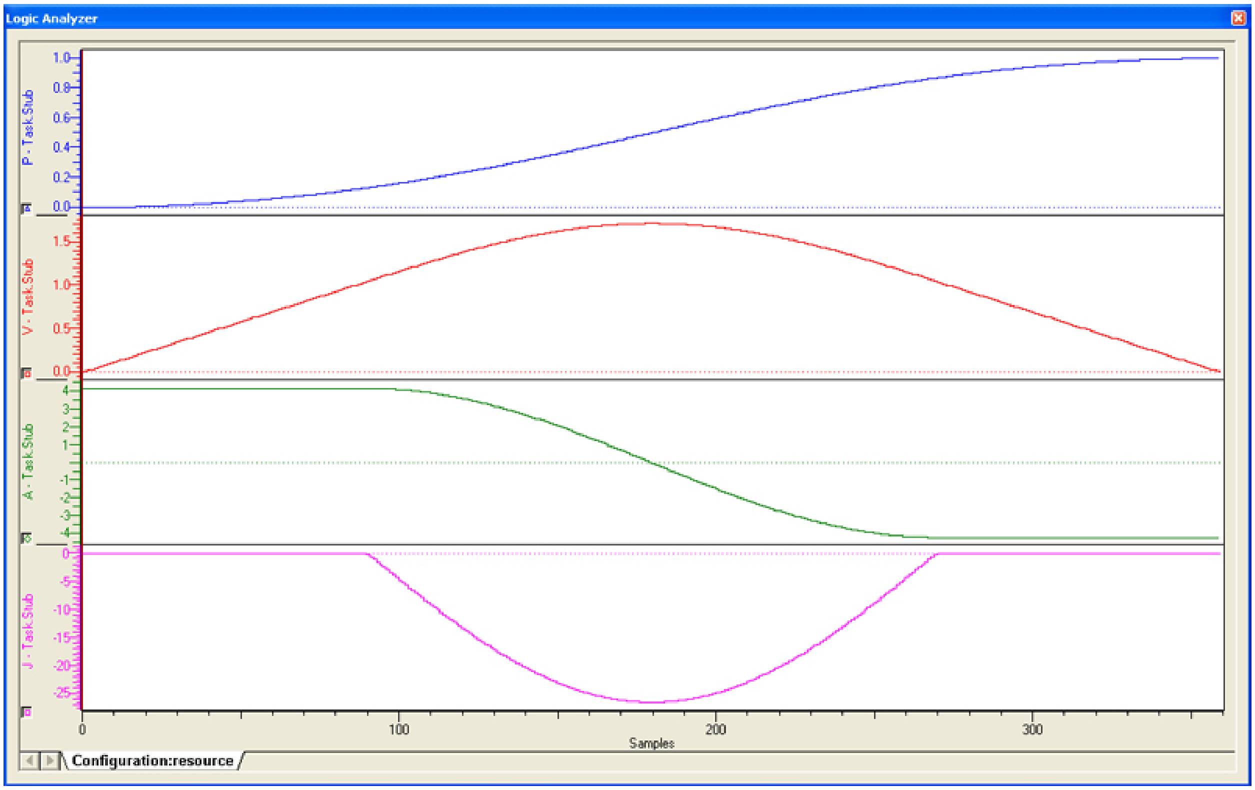
Notes: Deceleration is twice as long as acceleration, which provides the effect of restricting vibration.



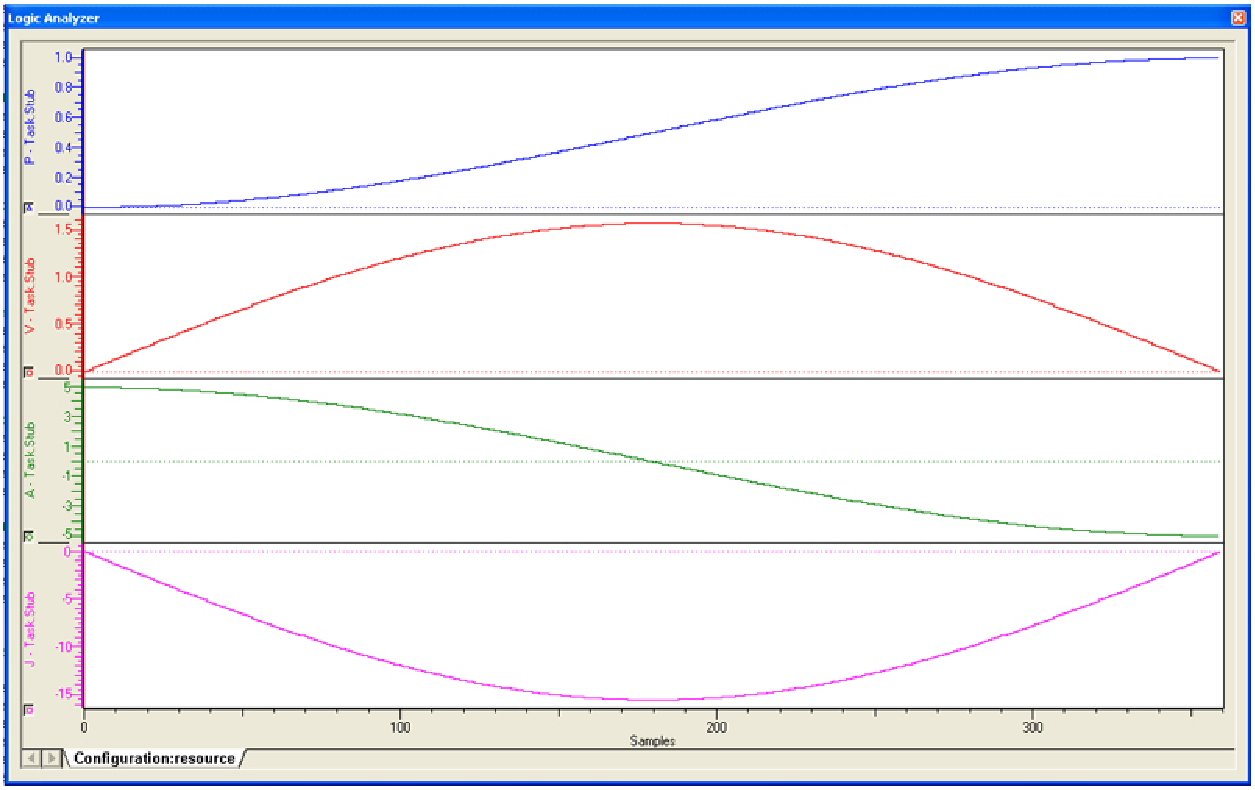
No Dwell Modified Constant Velocity



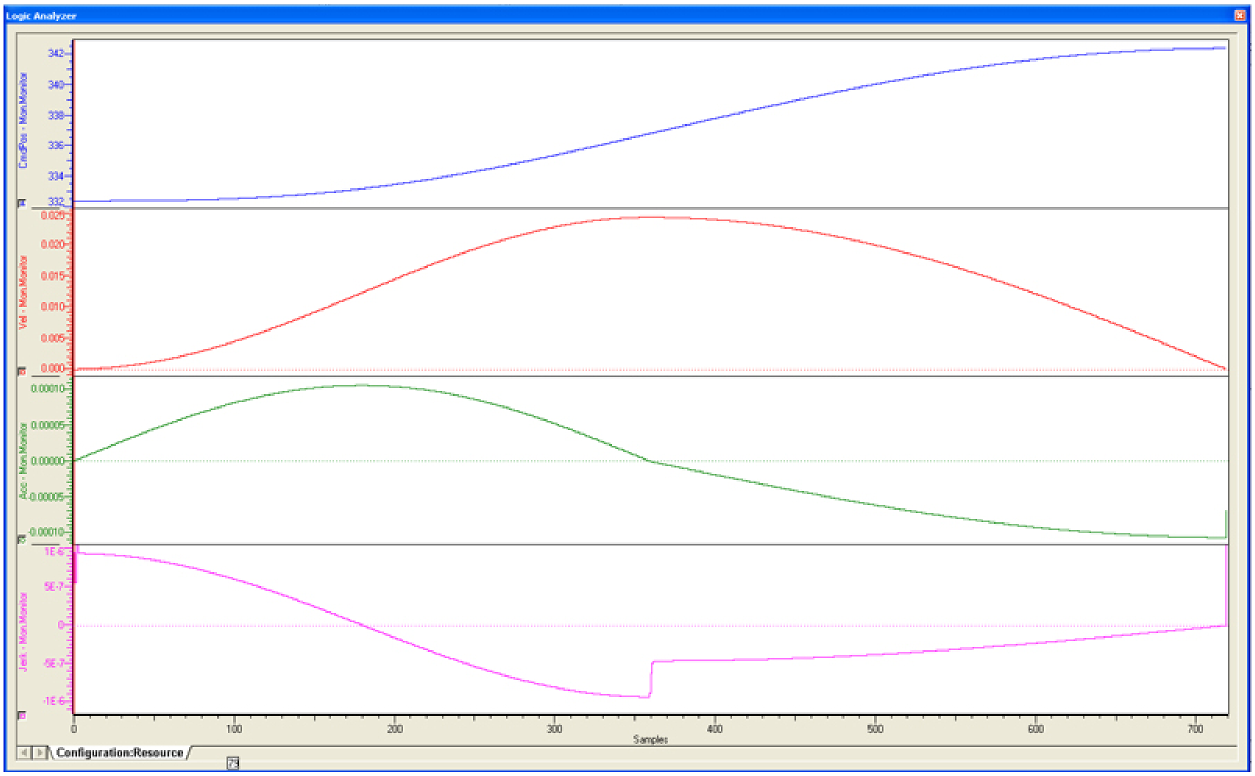
No Dwell Modified Trapezoid



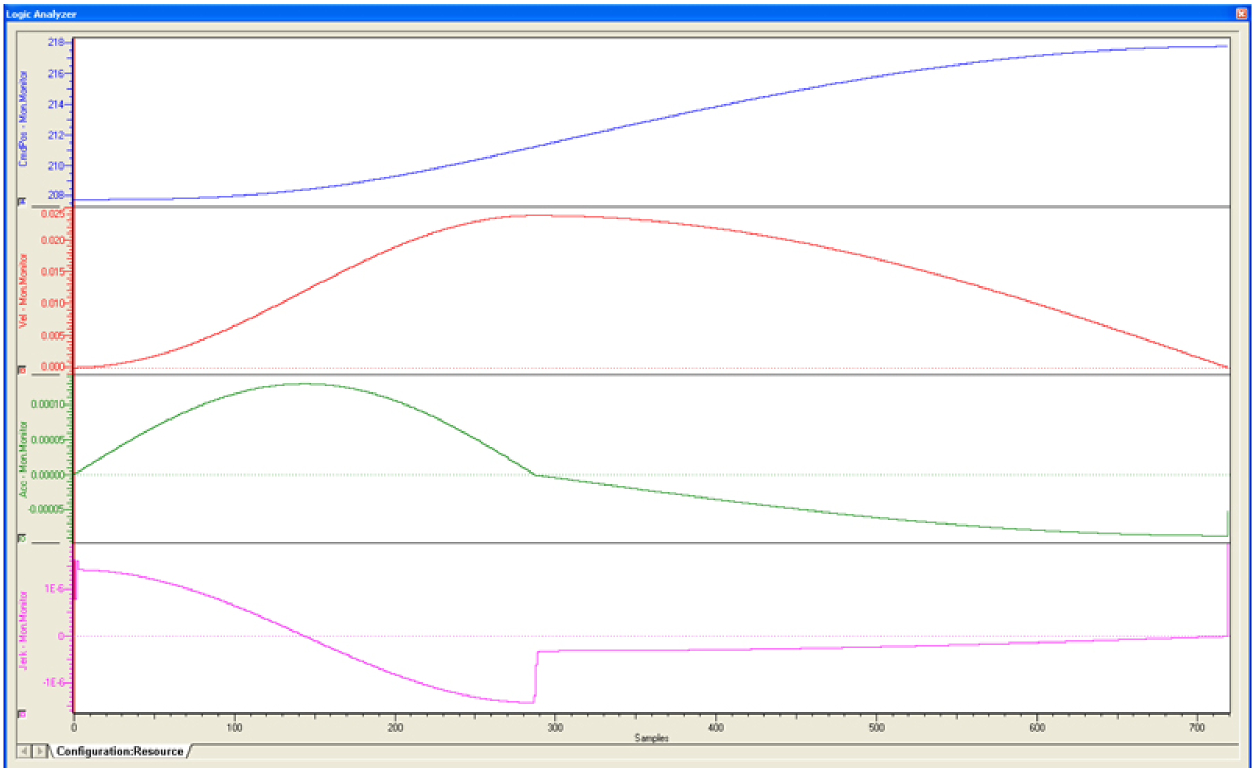
No Dwell Simple Harmonic



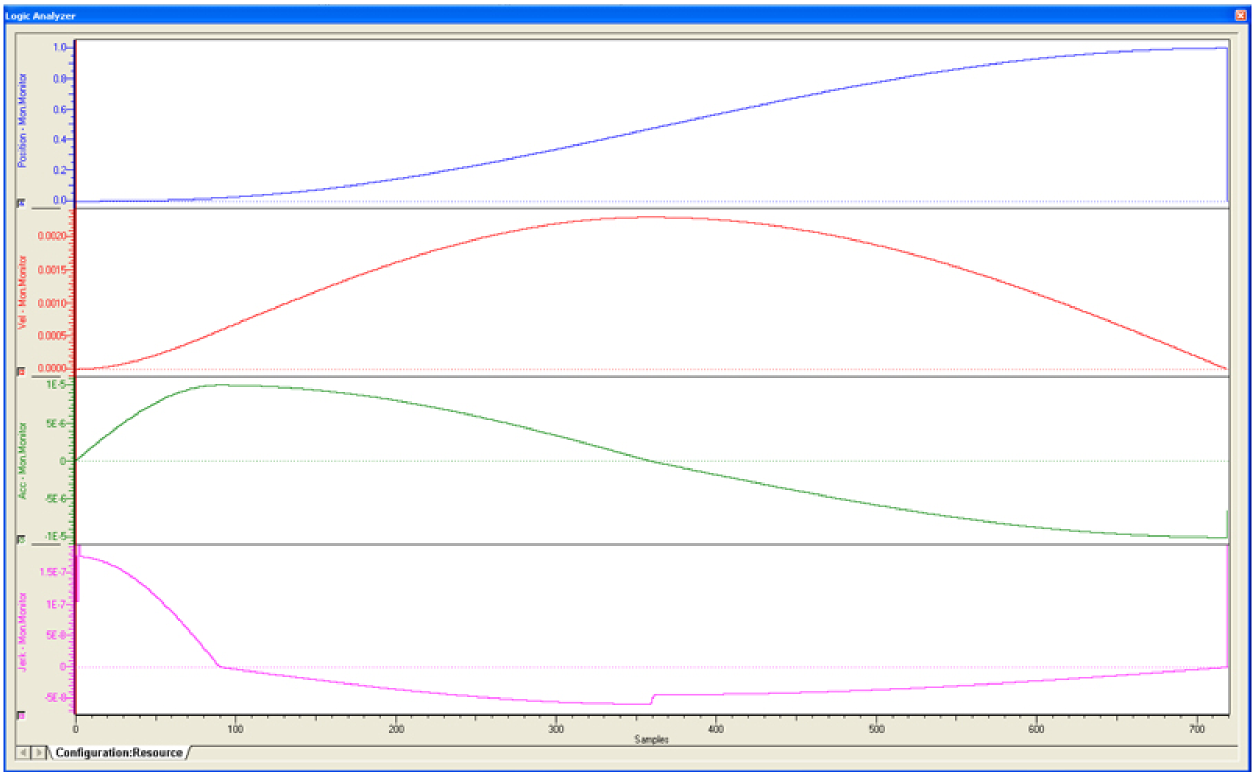
One Dwell Cycloidal_1



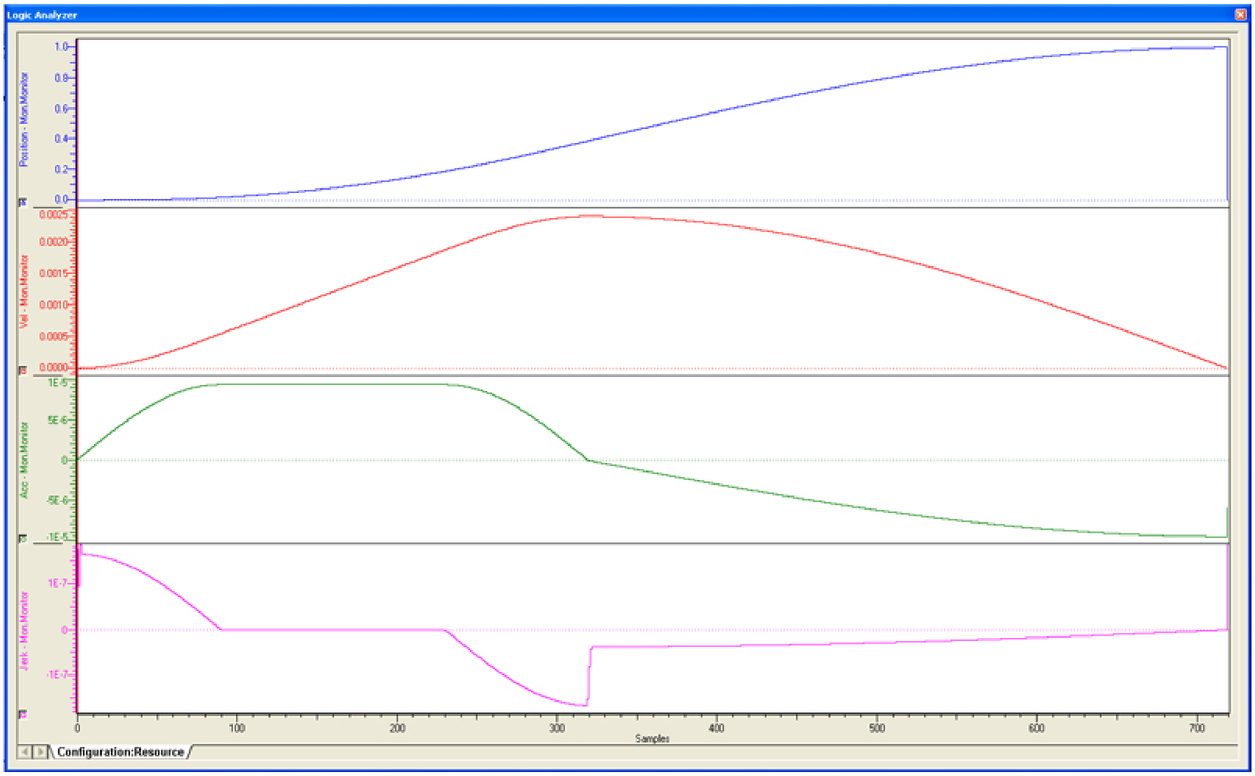
One Dwell Cycloidal_2_3



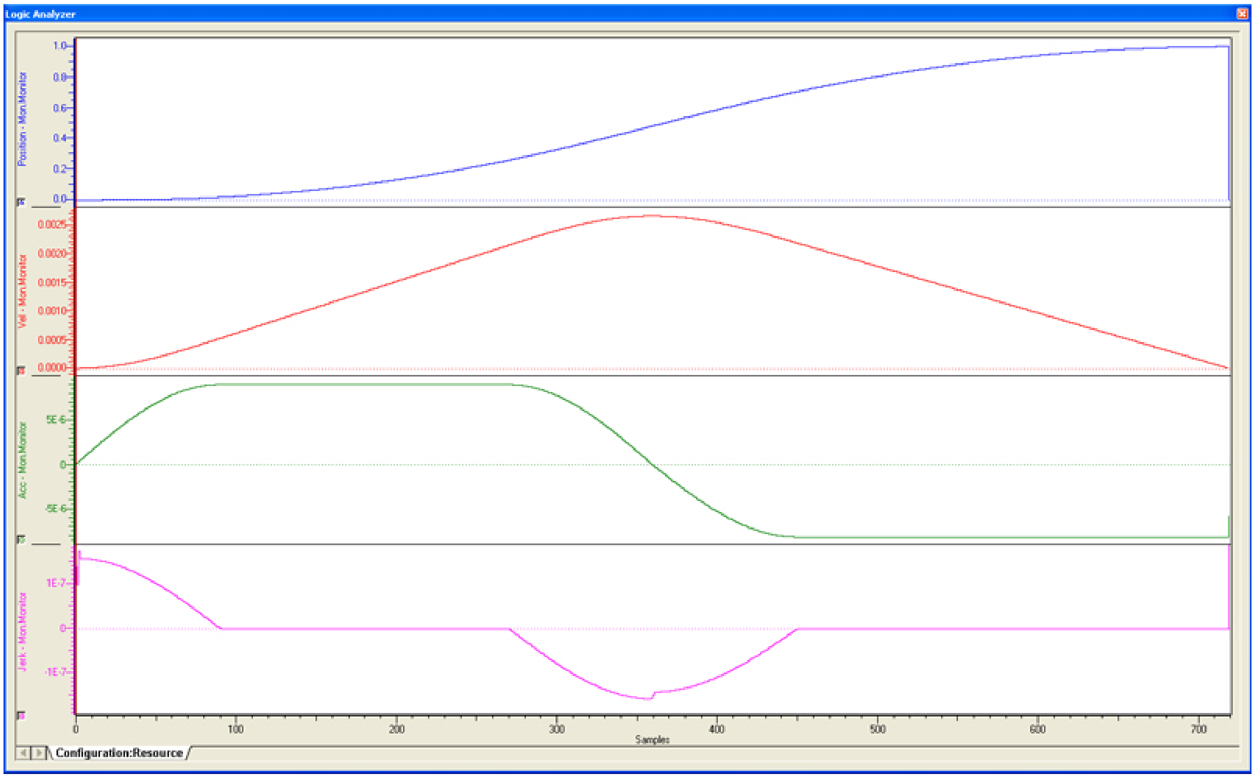
One Dwell Modified Sine



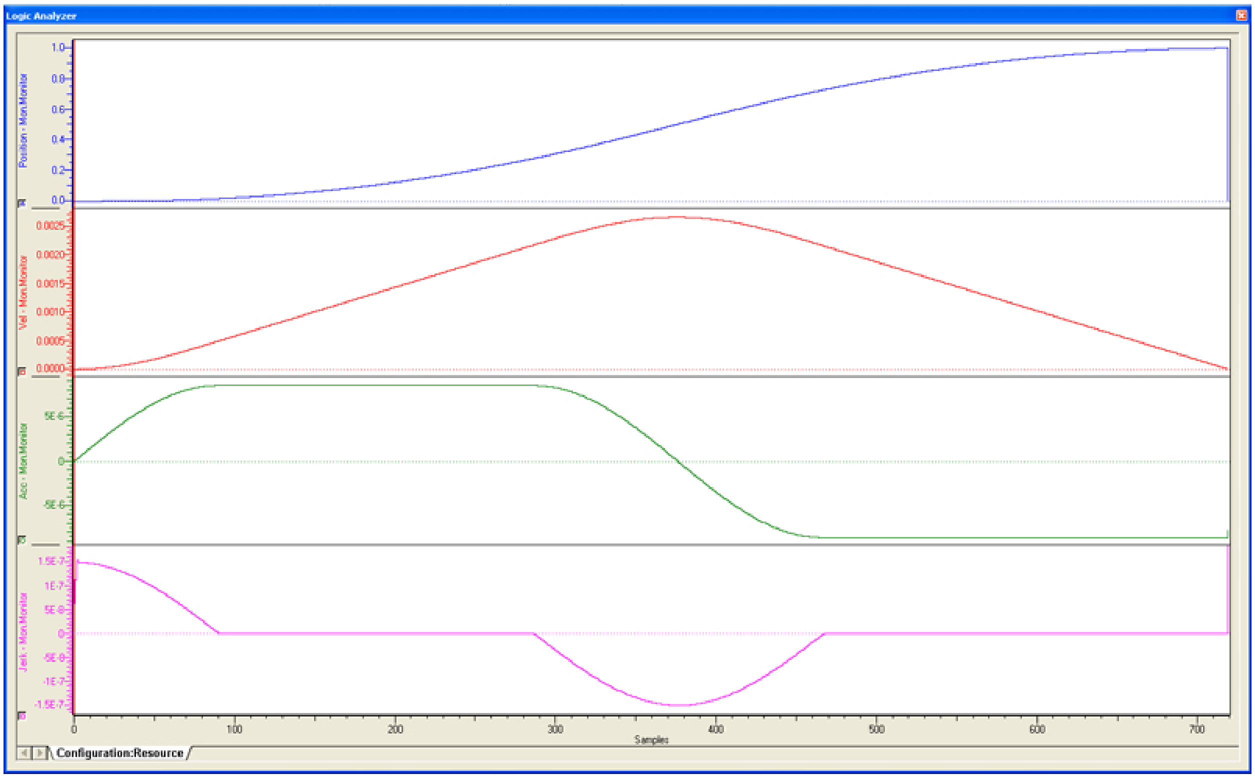
One Dwell Trapecloid



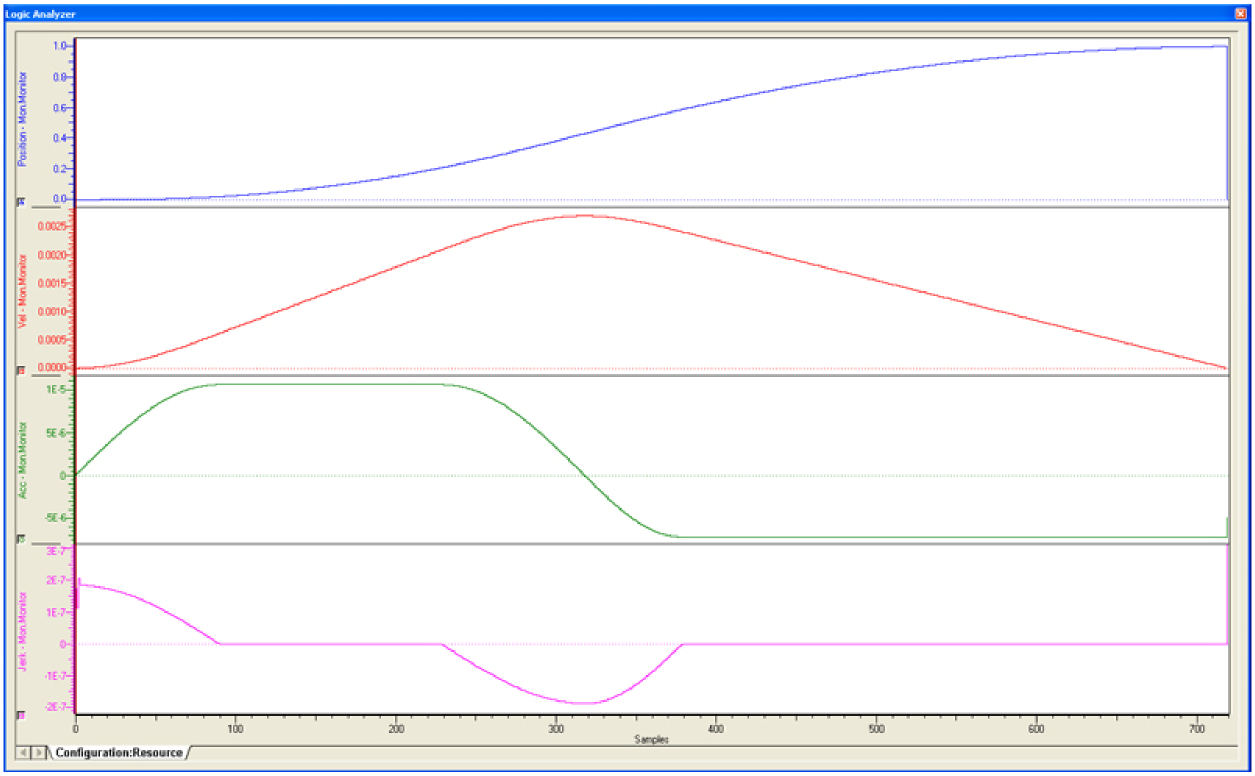
One Dwell Trapezoid



One Dwell Trapezoid_1

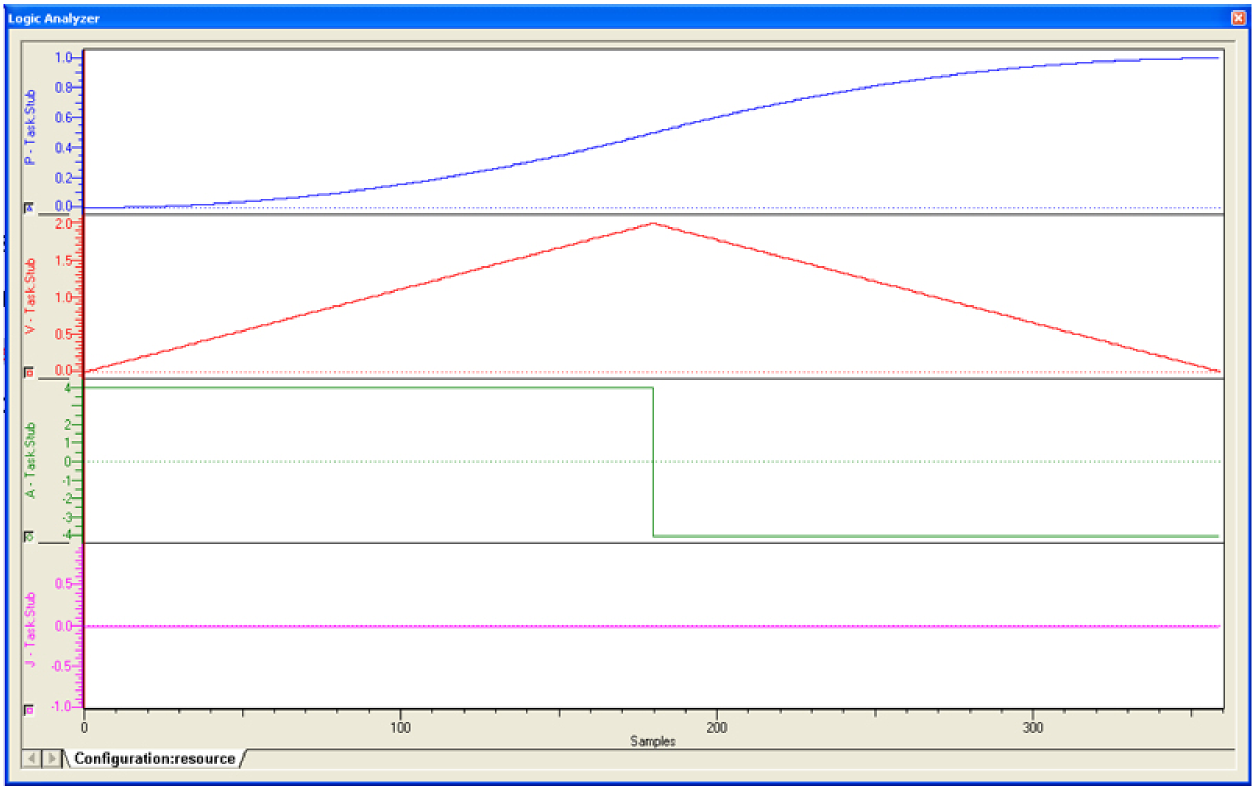


One Dwell Trapezoid_2_3

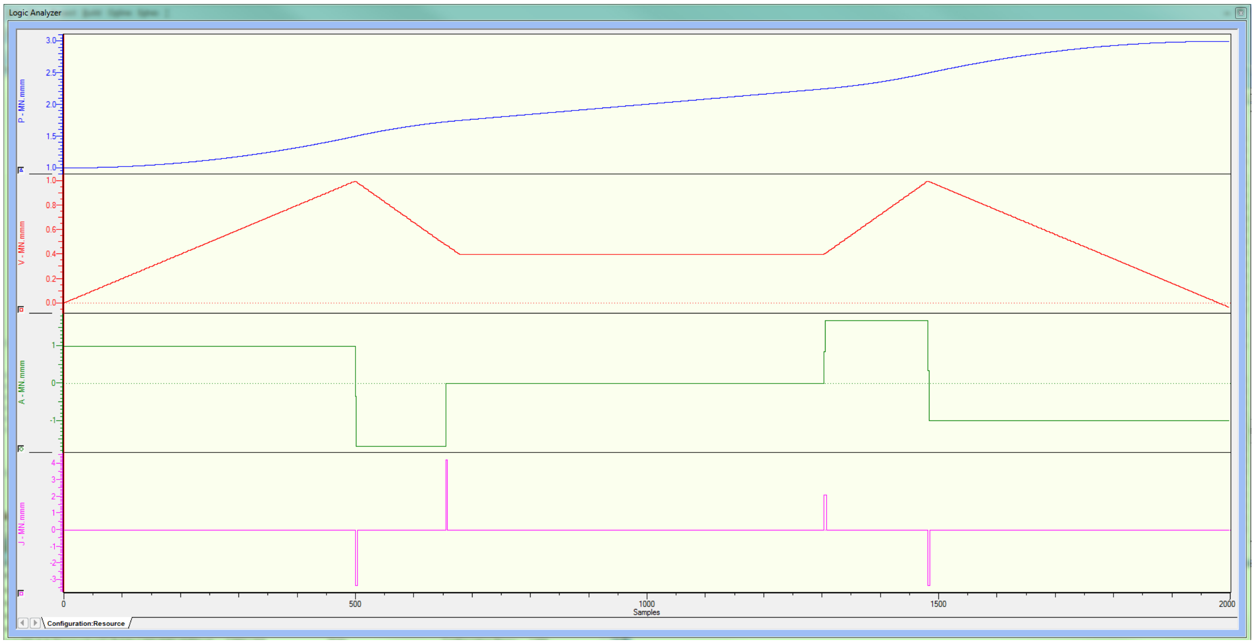


Parabolic

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 Am is the minimum (Am=4) among all curves.   Downside – Can cause vibration. Modified Trapezoid is better.



ParabolicVelocityBlend



Reverse Double Harmonic

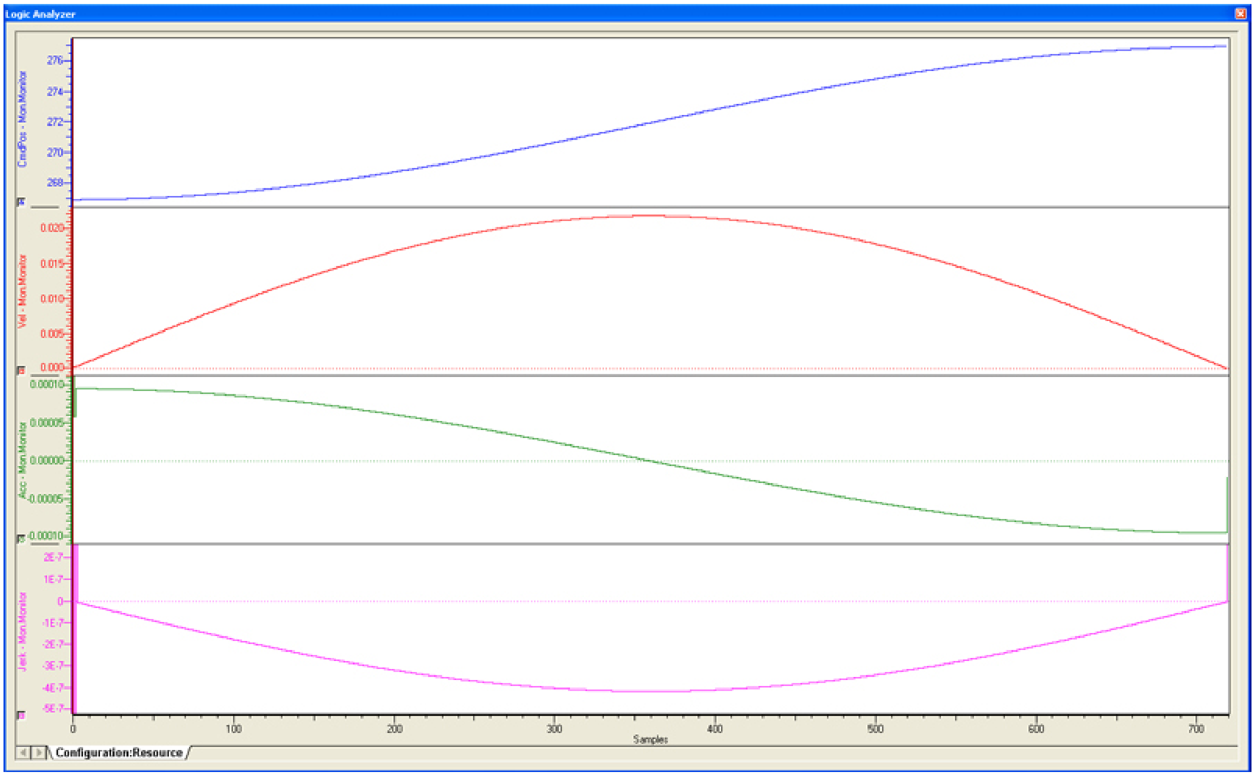
This curve type is not supported.

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

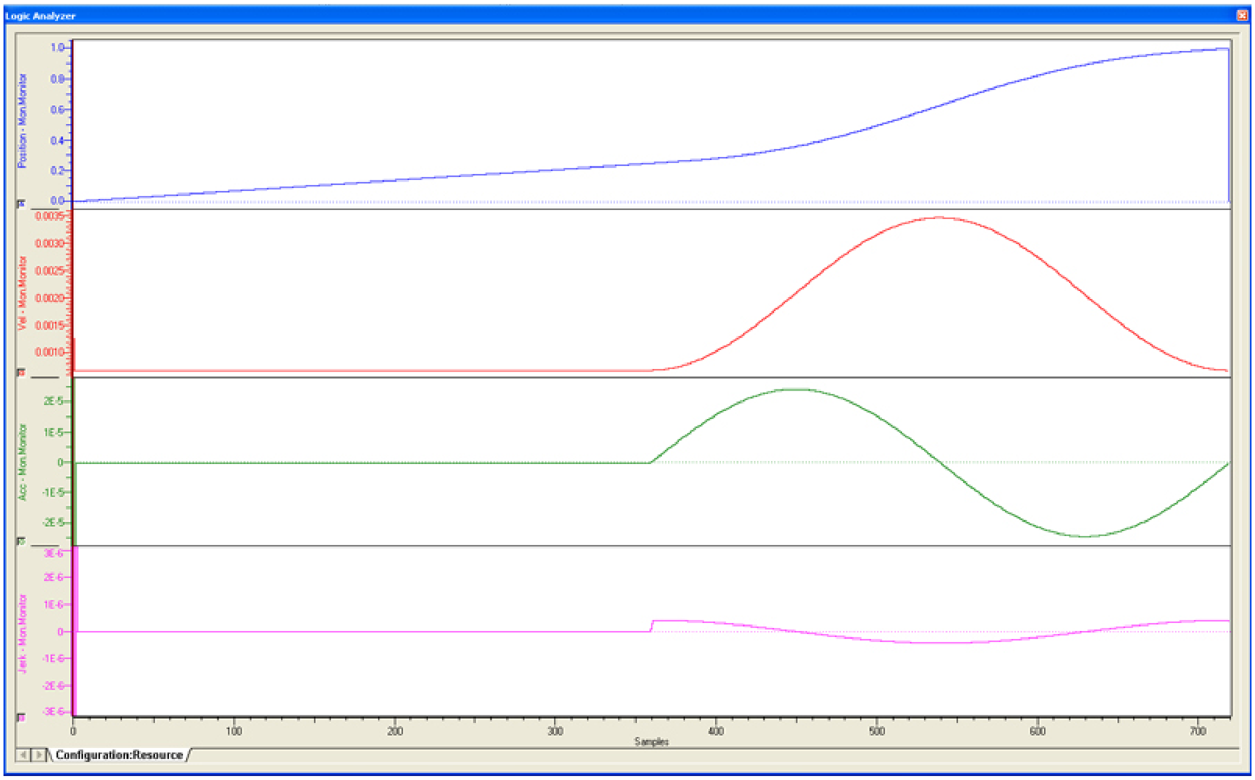
Provides the same profile as Tangent Matching, but designed for use with the CamBlend function block. The difference between this and Tangent Matching is how the matching velocity is determined. For this formula type, two segments are required: a straight line and a tangent blend. Which segment comes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

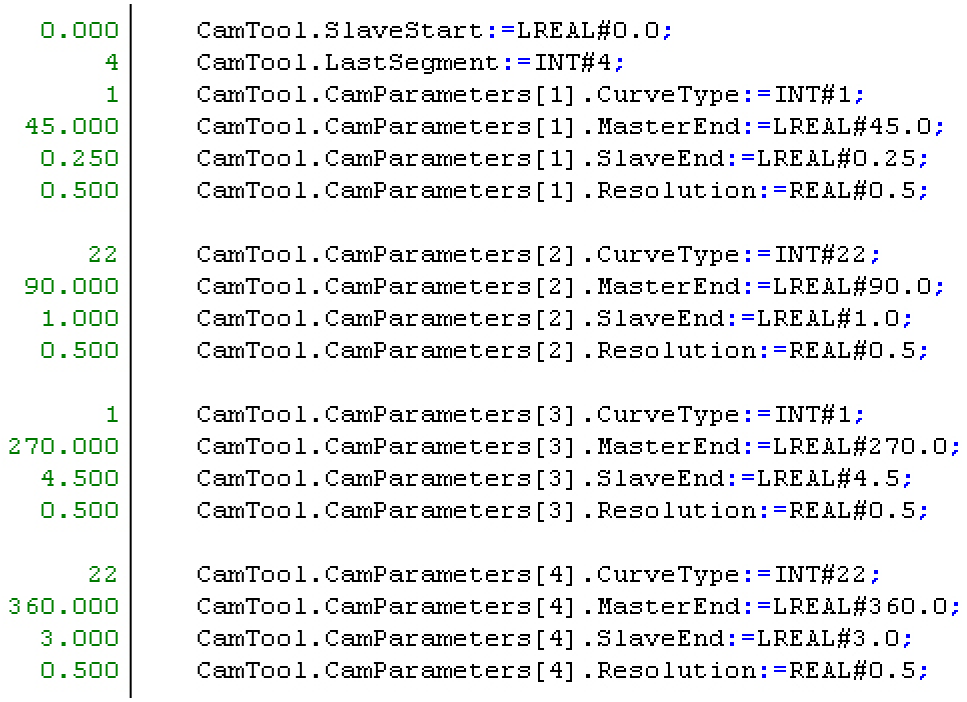
See the CamBlend function block for application examples

Tangent Matching

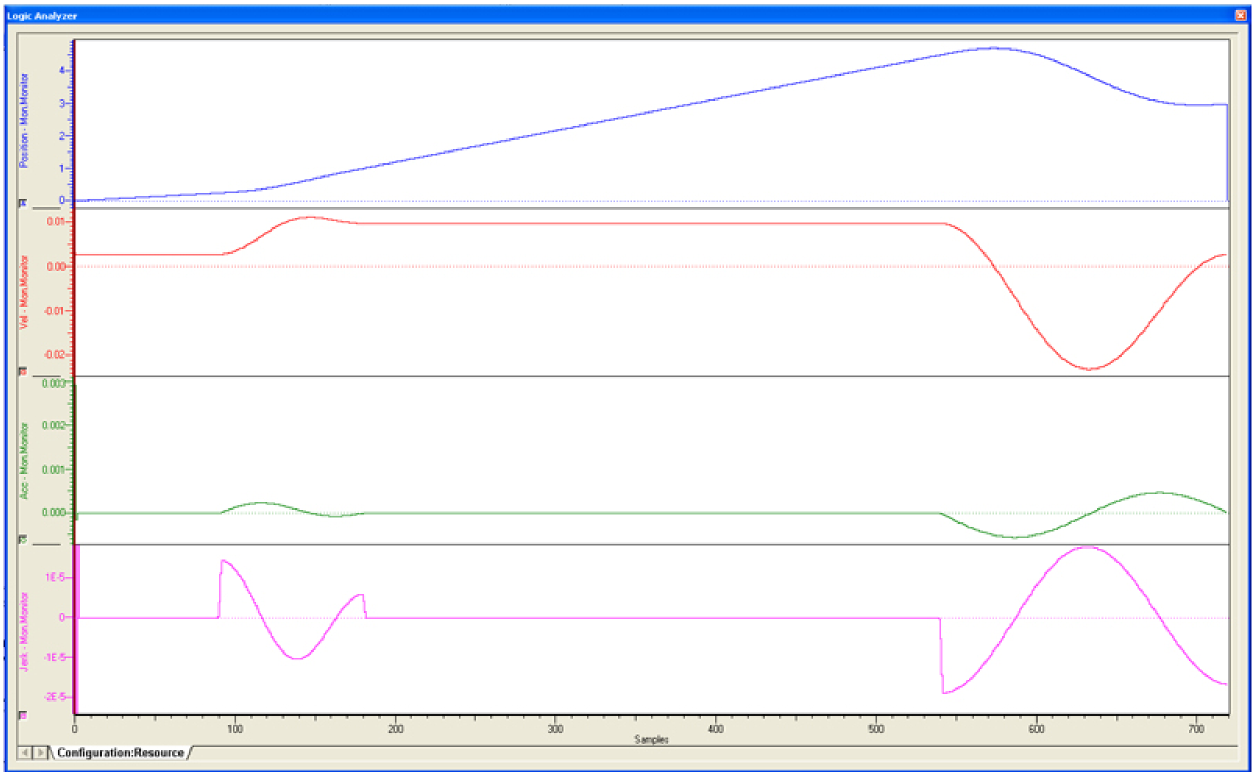
Provides a speed matched profile to minimize jerk between segments. Matches to the previous and next segment. In the case of the Tangent match segment coming first or last, a wraparound match is calculated. A straight line segment is required before and after the tangent match segment.

```
  0.000 | CamTool.SlaveStart:=LREAL#0.0;
      2 | CamTool.LastSegment:=INT#2;
      1 | CamTool.CamParameters[1].CurveType:=INT#1;
180.000 | CamTool.CamParameters[1].MasterEnd:=LREAL#180.0;
  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;
360.000 | CamTool.CamParameters[2].MasterEnd:=LREAL#360.0;
  1.000 | CamTool.CamParameters[2].SlaveEnd:=LREAL#1.0;
  0.500 | CamTool.CamParameters[2].Resolution:=REAL#0.5;
```

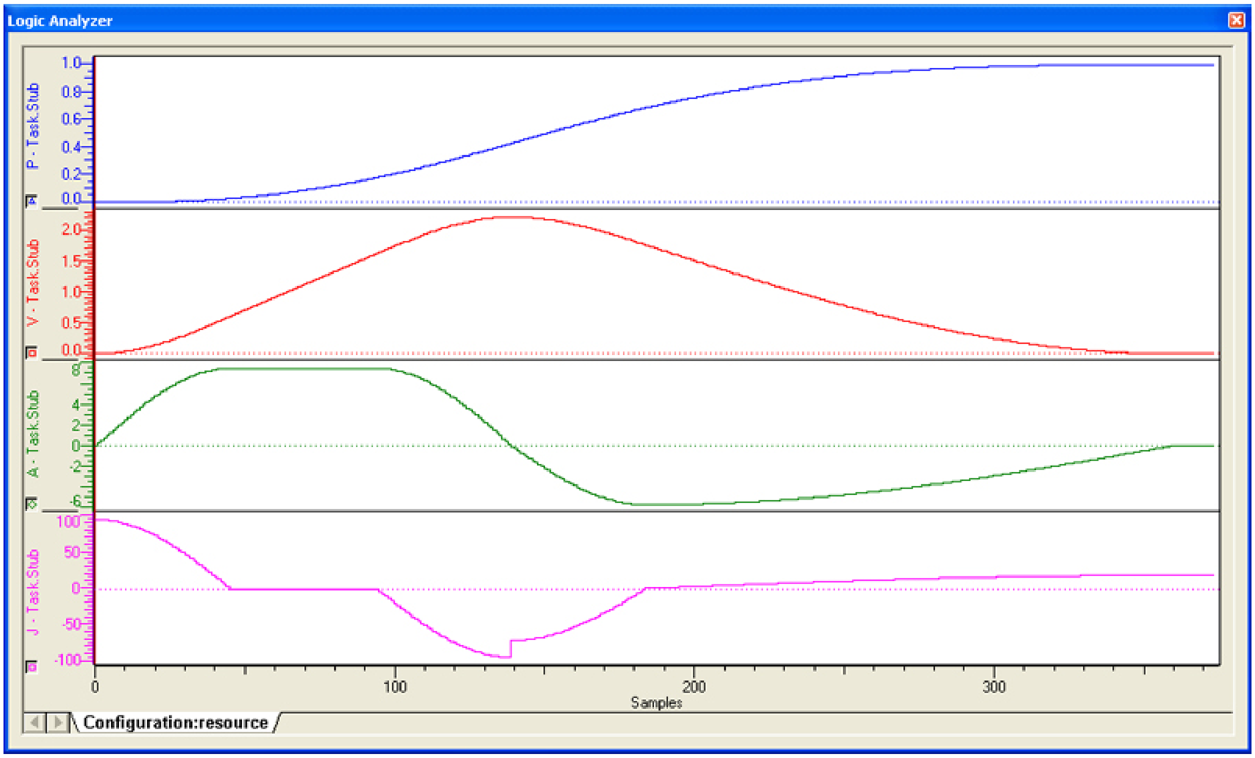




```
  0.000 |   CamTool.SlaveStart:=LREAL#0.0;
      4 |   CamTool.LastSegment:=INT#4;
      1 |   CamTool.CamParameters[1].CurveType:=INT#1;
 45.000 |   CamTool.CamParameters[1].MasterEnd:=LREAL#45.0;
  0.250 |   CamTool.CamParameters[1].SlaveEnd:=LREAL#0.25;
  0.500 |   CamTool.CamParameters[1].Resolution:=REAL#0.5;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



Trapecloid


Logic Analyzer
P - Task.Stub
1.0
0.8
0.6
0.4
0.2
0.0
V - Task.Stub
2.0
1.5
1.0
0.5
0.0
A - Task.Stub
8
4
2
0
-2
-6
J - Task.Stub
100
50
0
-50
-100
0
100
200
300
Samples
Configuration:resource

Getting Started with Communications Toolbox

Requirements for v301

To use the Communications Toolbox, your 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 (v301 or higher)

Communications Revision History

Current Version:

** 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 staritng to copy. Only a big problem (corrupted data) if heavy data streaming pushed the buffering capacity to its limts.

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.

Previous Versions:

(******************************** 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 verison to CHANGE the datatype 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: 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 MPiec 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		MyCommStruct.Serial.something
U	Ethernet	EthernetConfig		MyCommStruct.Ethernet.something

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	
	2	Ethernet	
Method	**For use with the GetParameter function. Specifies how the value is obtained.**		
	0	Parameter	
	1	Character	

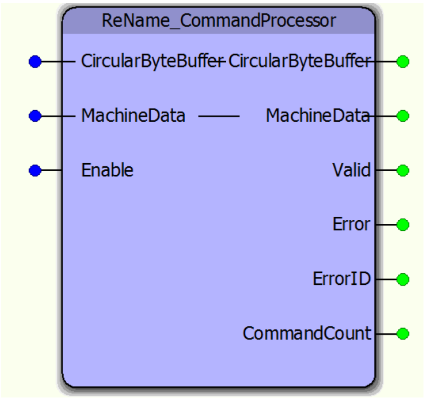
Enumerated Type: Method

Enumerated Type: COM_Type

Communications FBs

Toolbox Help Documentation
Help version created 1/25/2017
YASKAWA

CommandProcessor



This function block must be copied and renamed into your main project and customized for your application. It is deigned to identify variable length commands in the CircularByteBuffer and process them on a case by case basis. Typically machine specific information is populated within a CASE statement for use on the IEC application.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	CircularByteBuffer	CircularBufferStruct		
V	MachineData	YTB_STRING512	A string of characters 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
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

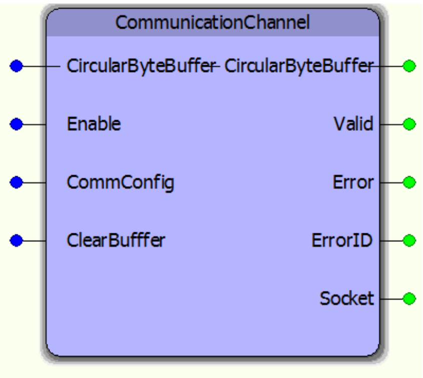
-
-

Error Description

ErrorID	Meaning
0	No error.
8705	The maximum number of concurrently open user sockets/IO device handles has been reached or exceeded.

Example

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 add 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				
B	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
B	CommConfig	CommStruct	Structure containing information to be used in establishing socket or serial communication	
B	ClearBuffer	BOOL	Clears all contents of the circular buffer and resets StorePointer and UsePointer	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
B	Socket	DINT	File handle to be used when writing to device connected to the socket. Only valid when non-zero.	
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.	

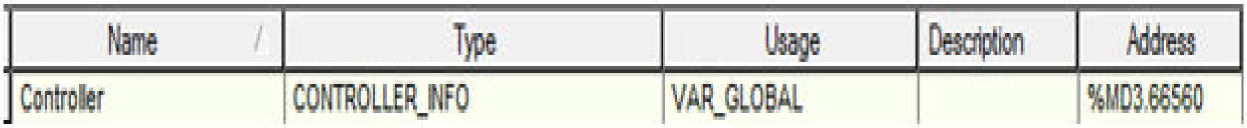
Error Description

ErrorID	Meaning
0	No error.
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.
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.
10022	Product or circular buffer overrun / full.
10023	Buffer size too small / cannot be zero.

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 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 mean the user will not be required to manually type in the controller's IP address for each system deployed.

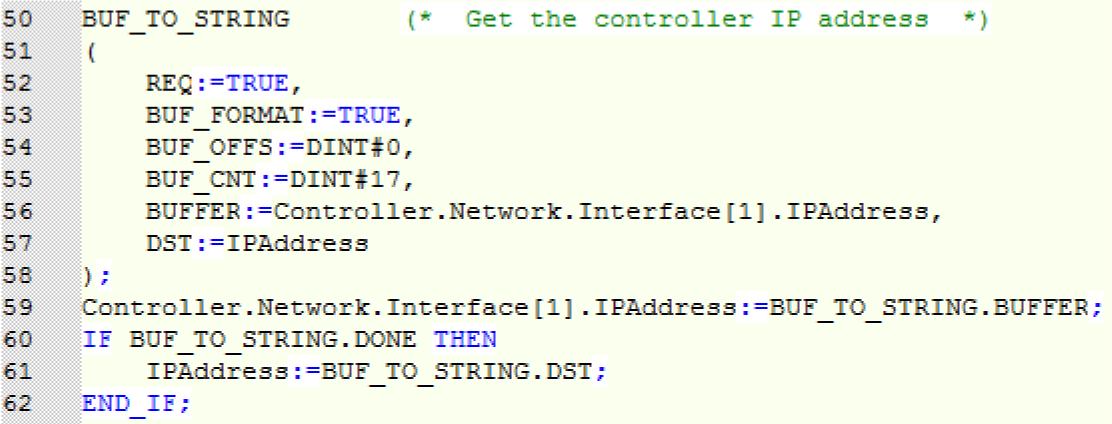
1. Add a variable of type CONTROLLER_INFO to Global Variables as shown below. The Address must be %MD3.66560.



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

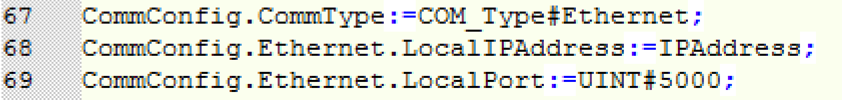
2. Add the following code to the initialize routine to obtain controller's IP address. The variable IPAddress is a STRING. The BUF_TO_STRING function block is located in the PROCONOS firmware library. As shown below, we are using it to extract 15 bytes of the IPAddress. These bytes equate to xxx.xxx.xxx.xxx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

IP Address.



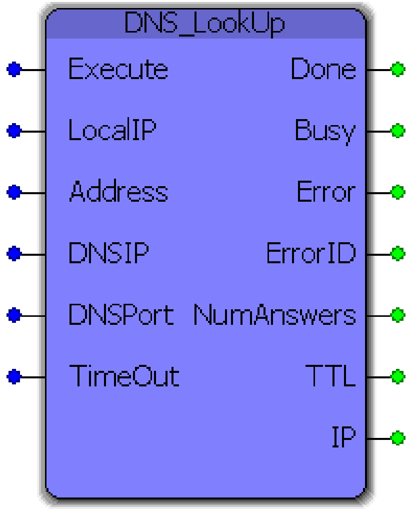
```
BUF_TO_STRING        (*  Get the controller IP address  *)
(
    REQ:=TRUE,
    BUF_FORMAT:=TRUE,
    BUF_OFFS:=DINT#0,
    BUF_CNT:=DINT#17,
    BUFFER:=Controller.Network.Interface[1].IPAddress,
    DST:=IPAddress
);
Controller.Network.Interface[1].IPAddress:=BUF_TO_STRING.BUFFER;
IF BUF_TO_STRING.DONE THEN
    IPAddress:=BUF_TO_STRING.DST;
END_IF;
```

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



```
CommConfig.CommType:=COM_Type#Ethernet;
CommConfig.Ethernet.LocalIPAddress:=IPAddress;
CommConfig.Ethernet.LocalPort:=UINT#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	
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.	
V	LocalIP	YC_ STRING16	The IP address of the controller on the local network.	
V	Address	YC_ STRING128	The domain name to perform the look-up on (not an IPV4 address).	
V	DNSIP	YC_ STRING16	The IP address of the DNS server to perform the lookup through.	
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.	

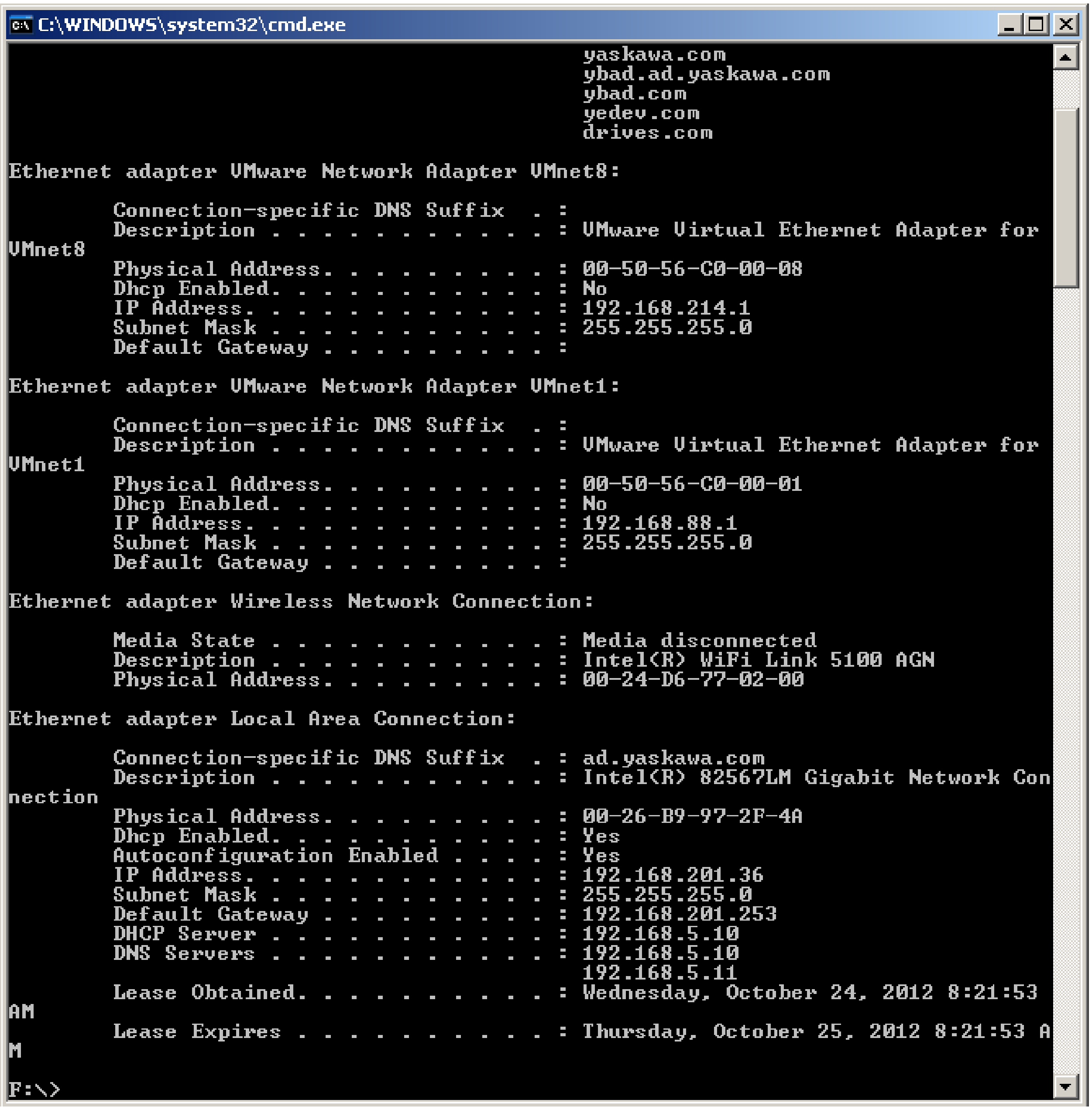
B	Busy	BOOL	Set high upon the rising edge of 'Execute' and reset if Done or Error is true.
B	Error	BOOL	Set high if an error has occurred during the DNS lookup. Cleared upon 'Execute' being reset.
B	ErrorID	UINT	If error is true, this output provides the Error ID. Cleared upon 'Execute' being reset.
E	NumAnswers	INT	The number of answers returned by the DNS server. The answer with the longest TTL is output at 'IP'
E	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. yaskawa.com), not an IPV4 address. Passing an IPV4 address is what is referred to as a "reverse DNS lookup" and is not supported by this block (reason: the Y_DeviceComm library needs an IPV4 address, not a domain name).
- What DNS server(s) your controller has access to depends on the network configuration. If you do not have a local DNS server (see "Setup" below) talk to your IT professional about what DNS server options you have.
- The main purpose of this block is use in other Communications blocks, such as FTP and SMTP.

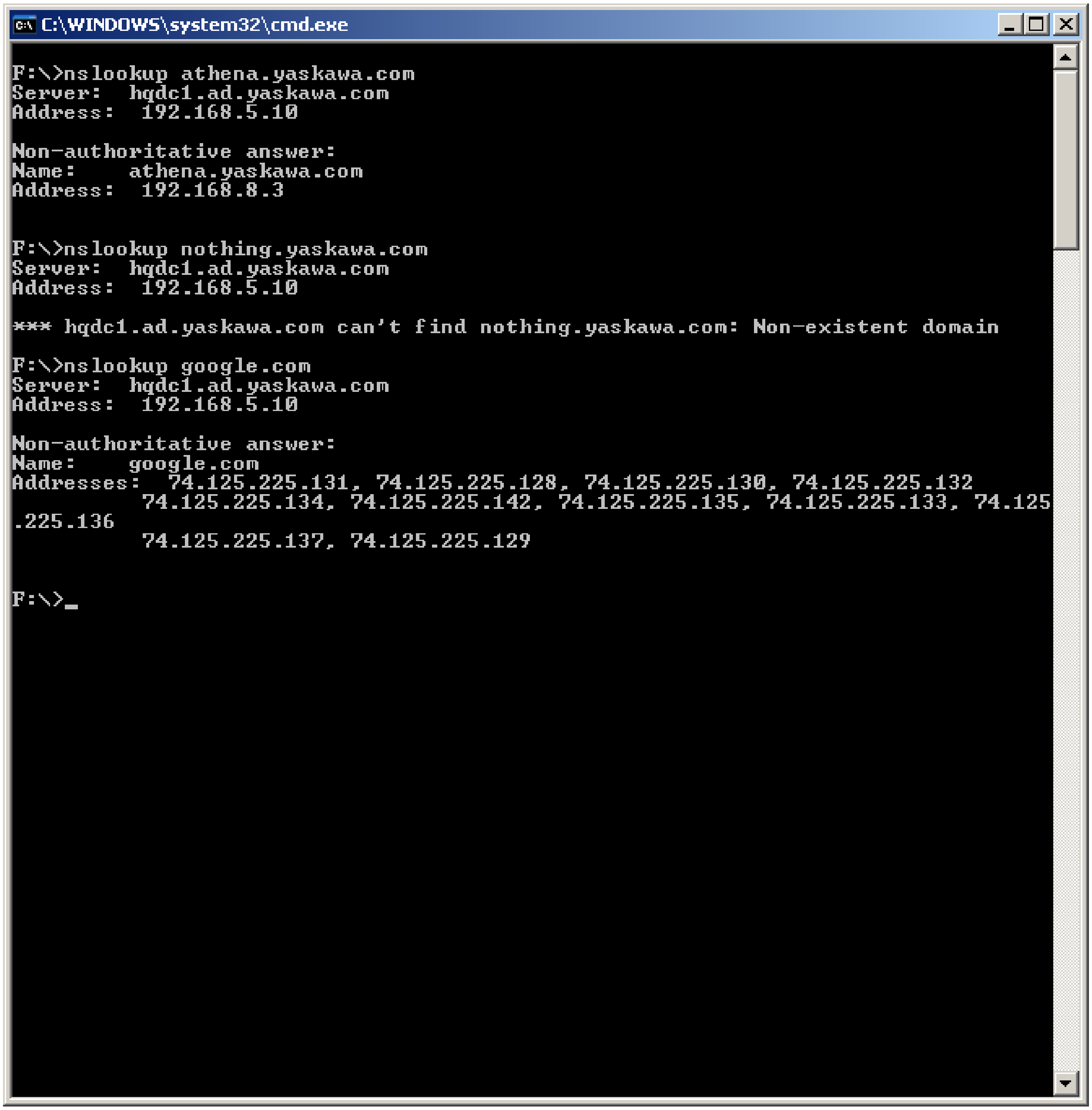
Setup

In order to perform a DNS lookup a connection to a DNS server must first be established. What DNS server you configure this block to use depends on your particular network set up. The easiest way to determine what DNS server to use (or at least to get started) 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 your computer is configured to use.



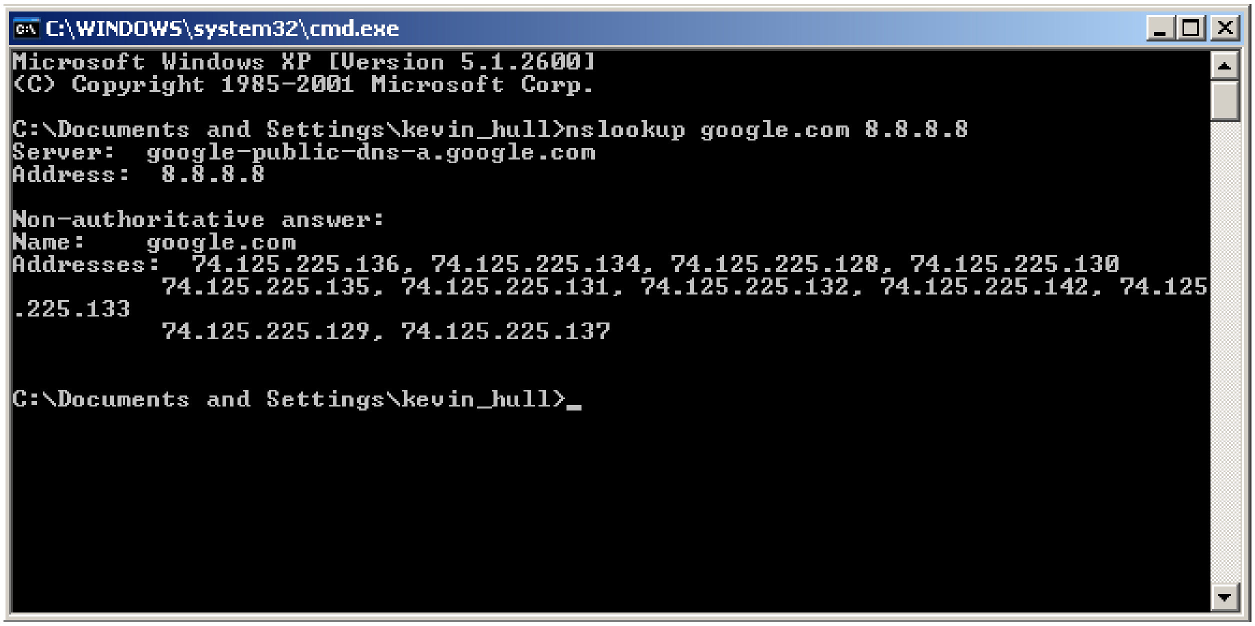
```
                                            yaskawa.com
                                            ybad.ad.yaskawa.com
                                            ybad.com
                                            yedev.com
                                            drives.com

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 A
M

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.



```
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.



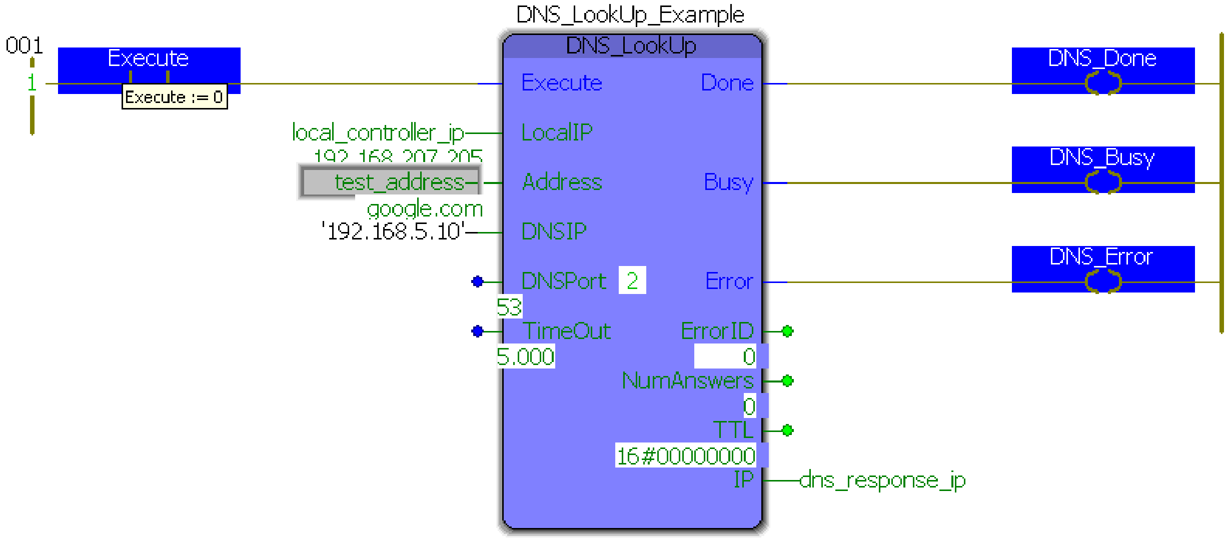
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

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>_
```

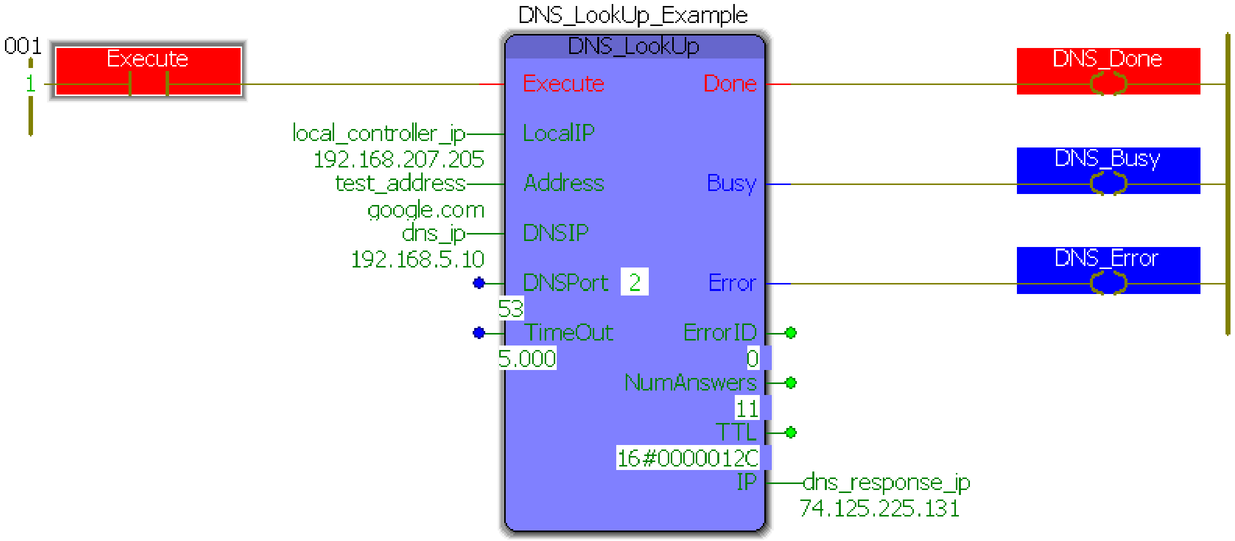
Error Description

ErrorID	Meaning
0	No error.
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.
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.
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.
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 '.'.

Example - External Address

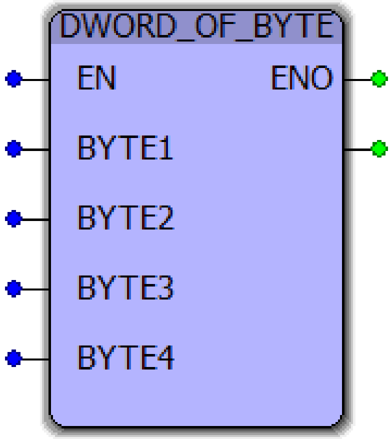
The following example demonstrates the blocks ability to perform a look up for an external address ("google.com") using an internal DNS server. The LocalIP, Address and DNSIP have all be configured and DNSPort and TimeOut have been left to defaults.





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 theresulting DWORD.

Library

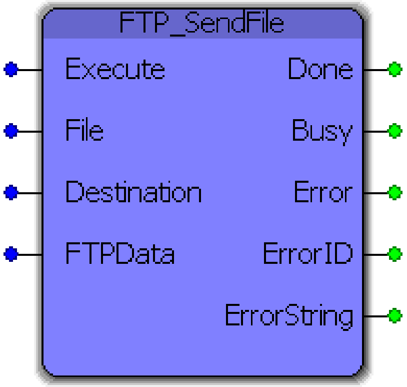
Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	EN	BOOL	Enables the function.	FALSE
V	BYTE1	BYTE	The least significant byte of the DWORD to be assembled.	BYTE#0
V	BYTE2	BYTE		BYTE#0
V	BYTE3	BYTE		BYTE#0
V	BYTE4	BYTE	The most significant byte of the DWORD to be assembled.	BYTE#0
VAR_OUTPUT				
B	ENO	BOOL	High if the function is executing normally.	

Notes

FTP_SendFile



This function block uses the FTP (File Transfer Protocol) to write a file to a specified FTP server.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all inputs are read and the file transfer is performed. To resend the file or send a different file, change the value(s) and re-trigger the execute input.	FALSE
V	File	YC_ STRING128	The full file name and location on the controller, e.g. '/flash/user/data/example.csv'.	STRING#''
V	Destination	YC_ STRING64	The full file name and destination on the FTP server, e.g. 'metrics/example.csv'.	STRING#''
V	FTPData	FTP_Data	The input structure that configures the FTP transfer such as FTP server address, port, etc.	
VAR_OUTPUT				
B	Done	BOOL	Set high upon the completion of a successful file transfer.	
B	Busy	BOOL	Set high upon the start of the file transfer and low upon 'Done' or 'Error' becoming true.	
B	Error	BOOL	Set high when an error occurs during the file transfer. Set low upon Execute being reset.	
B	ErrorID	BOOL	If 'Error' is true, this output provides the Error ID. Cleared upon 'Execute' being reset.	
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 as SSL is not currently supported in the firmware. As a result, all FTP traffic sent and received (e.g. username, password, file data) is sent unencyrpted in plain text and is therefore visible to anyone with access to your internal network. However, this should not be a problem so long as the data you are sending is not of a sensitive matter and your 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

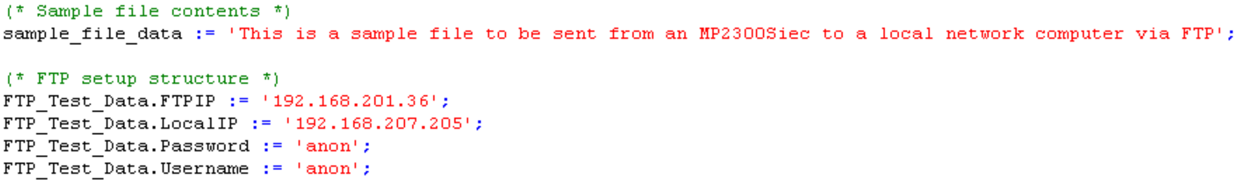
ErrorID	Meaning
0	No error.
8705	The maximum number of concurrently open user sockets/IO device handles has been reached or exceeded.
8706	The socket/IO device handle was invalid. Invalid IP address.
8707	The IP address string was not in a valid format.
8708	The socket/IO device handle could not be created.
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.
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.
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.

Basic Functionality Example - Transferring a File

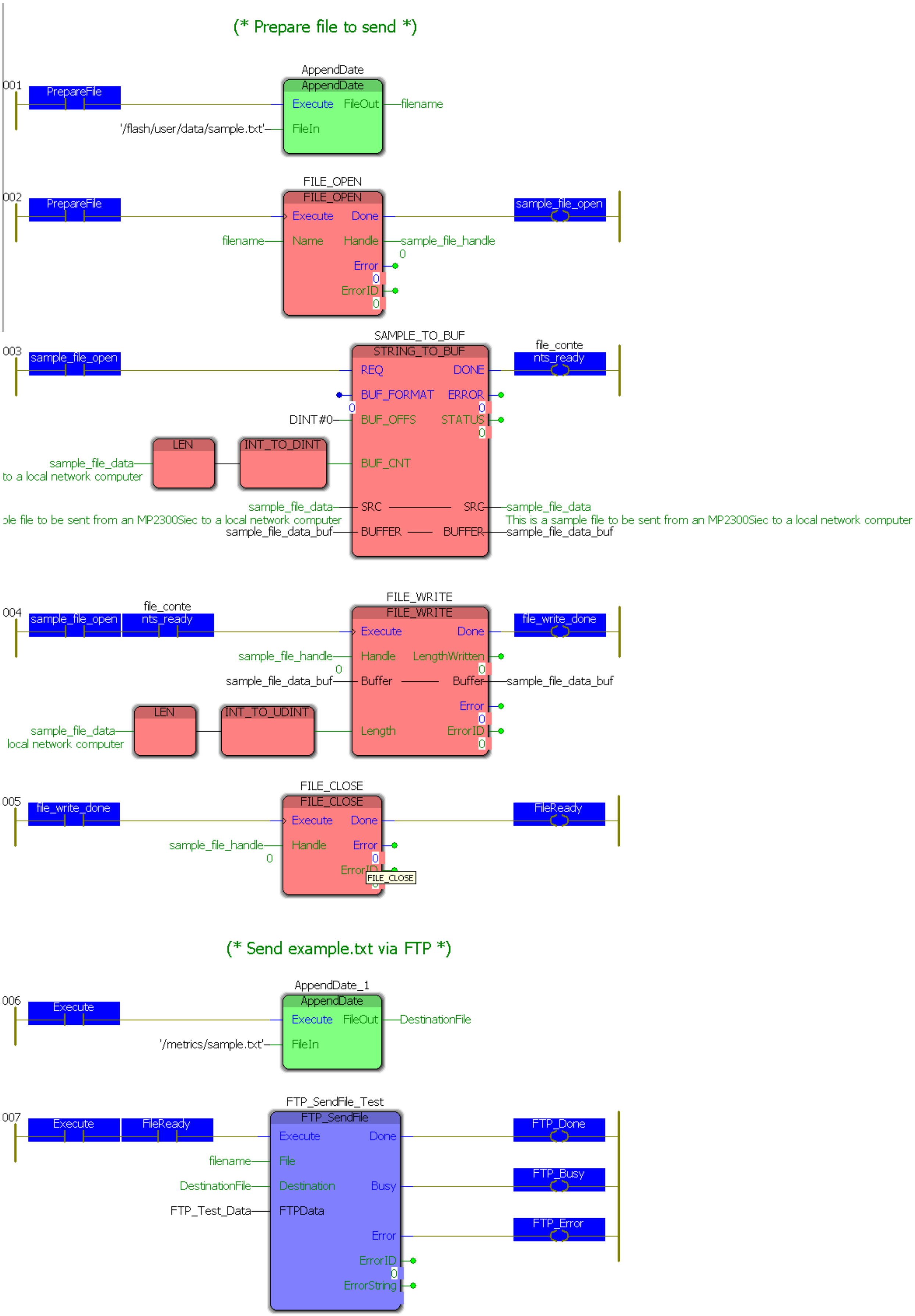
This examples demonstrates how to configure the block using the data structure, create a file to send and execute the FTP_SendFile block.

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.

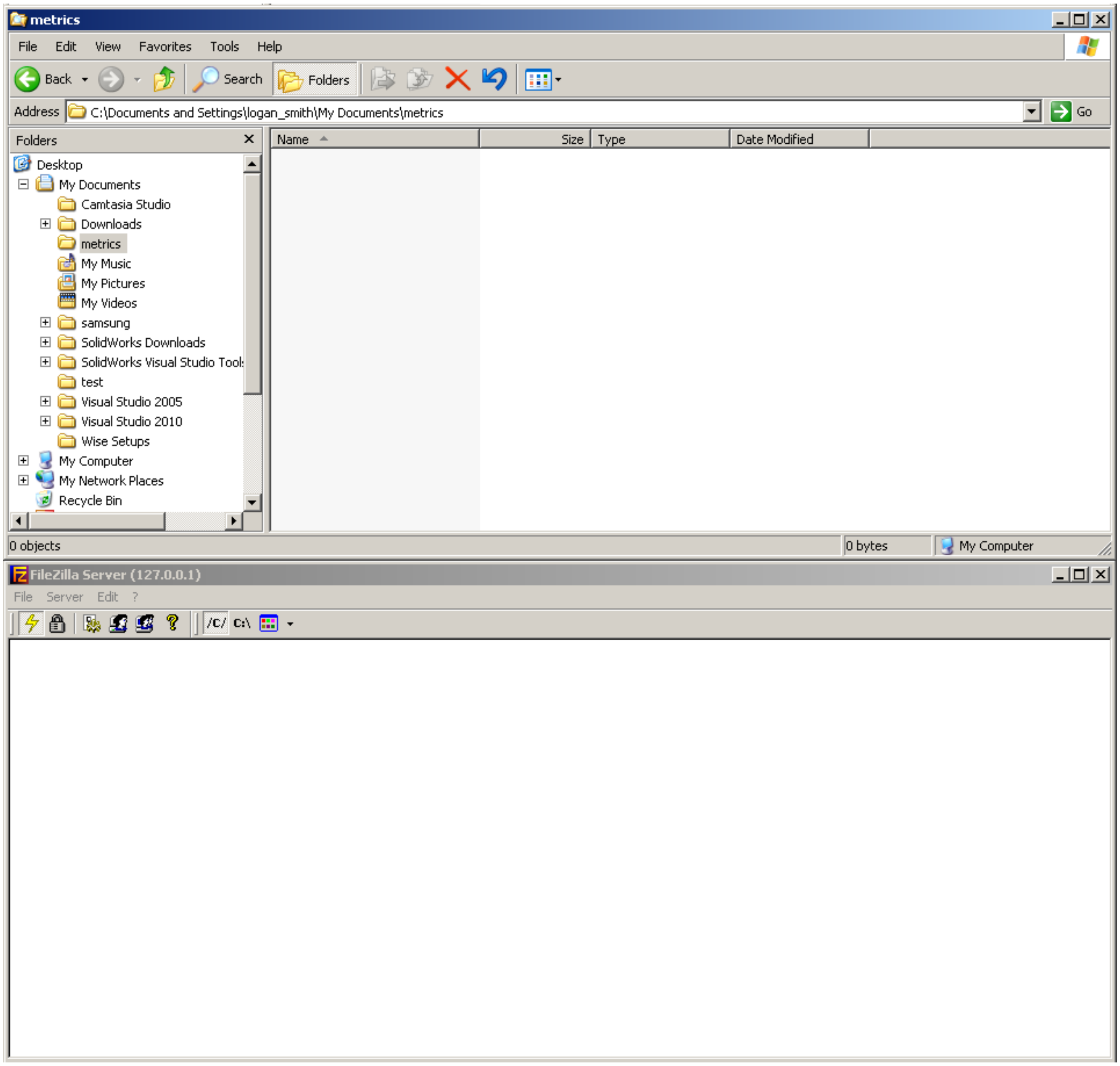


```
(* Sample file contents *)
sample_file_data := 'This is a sample file to be sent from an MP2300Siec to a local network computer via FTP';

(* 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 works by creating a file via the PROCONOS File_Open, String_to_Buf, File_Write and File_Close blocks. 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 block sends the file to the server.


(* Prepare file to send *)
001
PrepareFile
AppendDate
AppendDate
Execute
FileOut
filename
'/flash/user/data/sample.txt'
FileIn
002
PrepareFile
FILE_OPEN
FILE_OPEN
Execute
Done
sample_file_open
filename
Name
Handle
sample_file_handle
0
Error
0
ErrorID
0
003
sample_file_open
SAMPLE_TO_BUF
STRING_TO_BUF
REQ
DONE
file_conte
nts_ready
BUF_FORMAT
ERROR
0
0
DINT#0
BUF_OFFS
STATUS
0
LEN
INT_TO_DINT
sample_file_data
to a local network computer
BUF_CNT
sample_file_data
SRC
SRC
sample_file_data
ple file to be sent from an MP2300Siec to a local network computer
This is a sample file to be sent from an MP2300Siec to a local network computer
sample_file_data_buf
BUFFER
BUFFER
sample_file_data_buf
004
sample_file_open
file_conte
nts_ready
FILE_WRITE
FILE_WRITE
Execute
Done
file_write_done
sample_file_handle
Handle
LengthWritten
0
0
sample_file_data_buf
Buffer
Buffer
sample_file_data_buf
Error
0
LEN
INT_TO_UDINT
sample_file_data
local network computer
Length
ErrorID
0
005
file_write_done
FILE_CLOSE
FILE_CLOSE
Execute
Done
FileReady
sample_file_handle
Handle
Error
0
0
ErrorID
FILE_CLOSE
(* Send example.txt via FTP *)
006
Execute
AppendDate_1
AppendDate
Execute
FileOut
DestinationFile
'/metrics/sample.txt'
FileIn
007
Execute
FileReady
FTP_SendFile_Test
FTP_SendFile
Execute
Done
FTP_Done
filename
File
DestinationFile
Destination
Busy
FTP_Busy
FTP_Test_Data
FTPData
Error
FTP_Error
ErrorID
0
ErrorString

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.

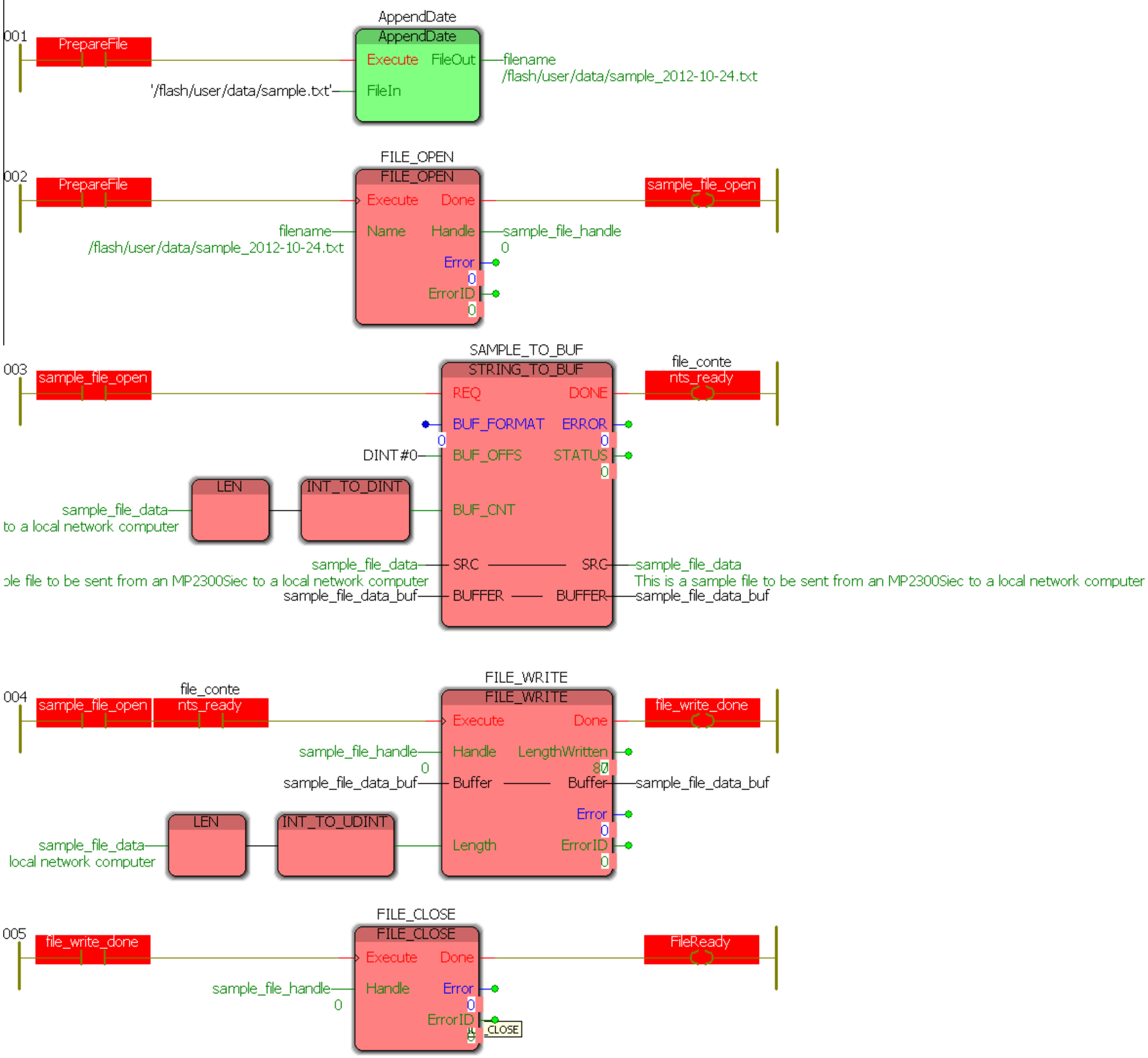


Address: C:\Documents and Settings\logan_smith\My Documents\metrics

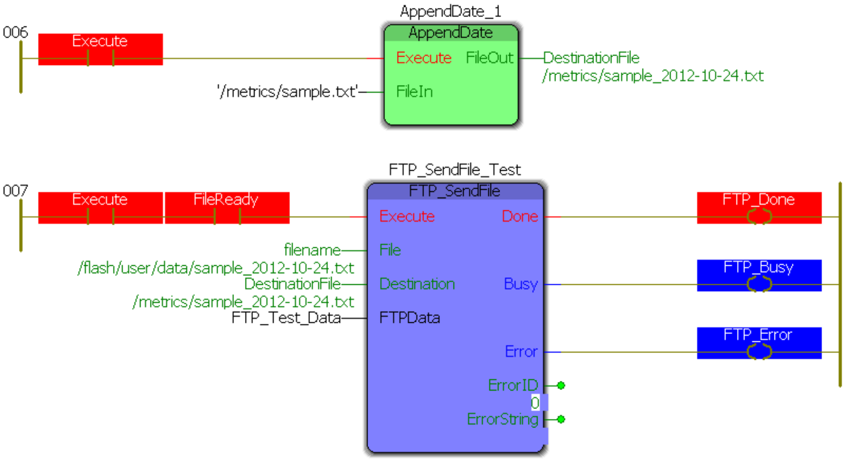
0 objects | 0 bytes

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.

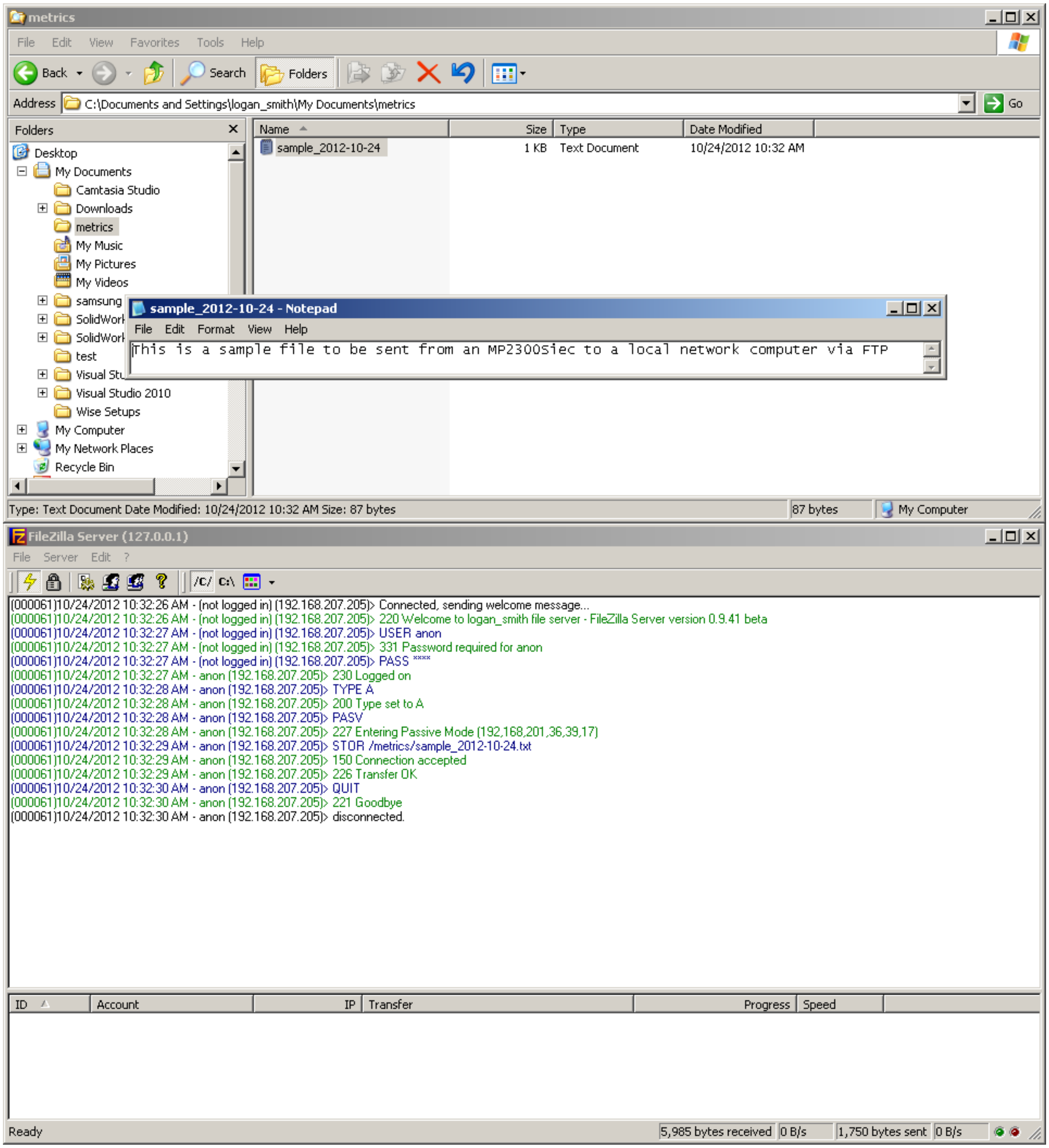
(* Prepare file to send *)



(* Send example.txt via FTP *)



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



C:\Documents and Settings\logan_smith\My Documents\metrics

Name	Size	Type	Date Modified
sample_2012-10-24	1 KB	Text Document	10/24/2012 10:32 AM

```
This is a sample file to be sent from an MP2300Siec to a local network computer via FTP
```

Type: Text Document Date Modified: 10/24/2012 10:32 AM Size: 87 bytes

```
(000061)10/24/2012 10:32:26 AM - (not logged in) (192.168.207.205)> Connected, sending welcome message...
(000061)10/24/2012 10:32:26 AM - (not logged in) (192.168.207.205)> 220 Welcome to logan_smith file server - FileZilla Server version 0.9.41 beta
(000061)10/24/2012 10:32:27 AM - (not logged in) (192.168.207.205)> USER anon
(000061)10/24/2012 10:32:27 AM - (not logged in) (192.168.207.205)> 331 Password required for anon
(000061)10/24/2012 10:32:27 AM - (not logged in) (192.168.207.205)> PASS ****
(000061)10/24/2012 10:32:27 AM - anon (192.168.207.205)> 230 Logged on
(000061)10/24/2012 10:32:28 AM - anon (192.168.207.205)> TYPE A
(000061)10/24/2012 10:32:28 AM - anon (192.168.207.205)> 200 Type set to A
(000061)10/24/2012 10:32:28 AM - anon (192.168.207.205)> PASV
(000061)10/24/2012 10:32:28 AM - anon (192.168.207.205)> 227 Entering Passive Mode (192,168,201,36,39,17)
(000061)10/24/2012 10:32:29 AM - anon (192.168.207.205)> STOR /metrics/sample_2012-10-24.txt
(000061)10/24/2012 10:32:29 AM - anon (192.168.207.205)> 150 Connection accepted
(000061)10/24/2012 10:32:29 AM - anon (192.168.207.205)> 226 Transfer OK
(000061)10/24/2012 10:32:30 AM - anon (192.168.207.205)> QUIT
(000061)10/24/2012 10:32:30 AM - anon (192.168.207.205)> 221 Goodbye
(000061)10/24/2012 10:32:30 AM - anon (192.168.207.205)> disconnected.
```

5,985 bytes received | 0 B/s | 1,750 bytes sent | 0 B/s

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.

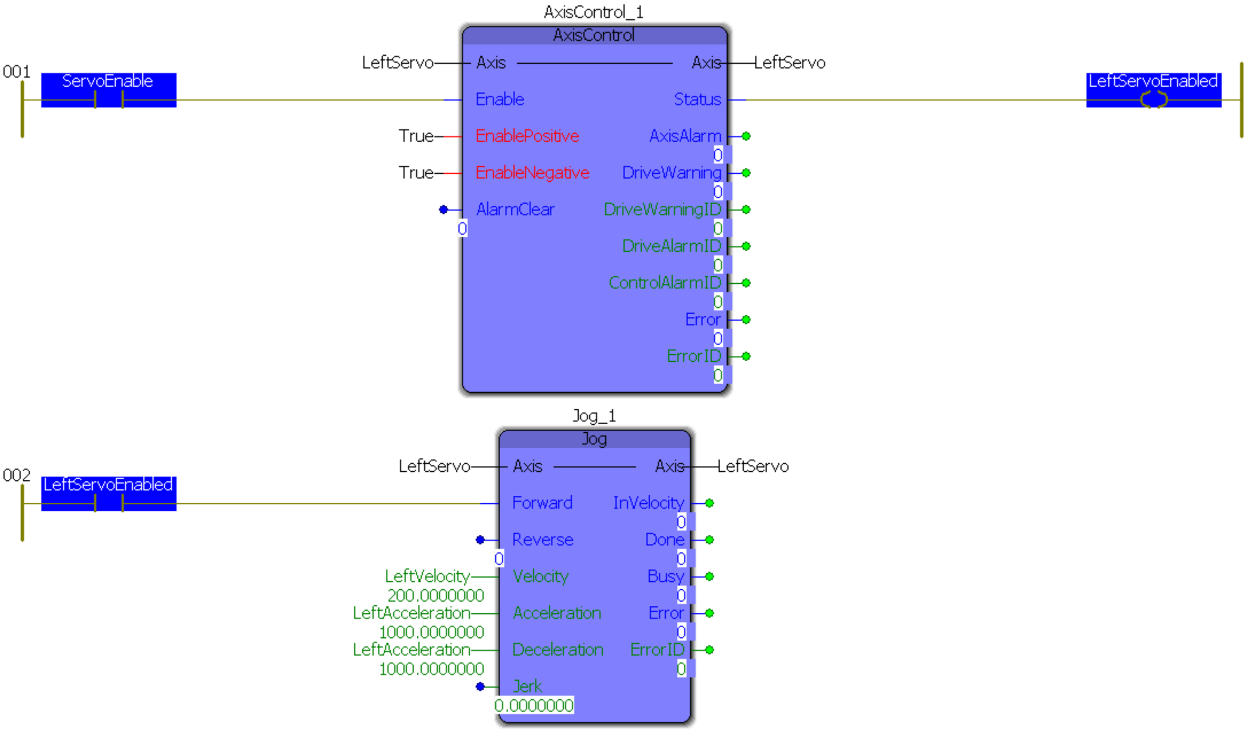
Advanced Functionality Example - Transferring a Metrics File at a Specified Rate

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.

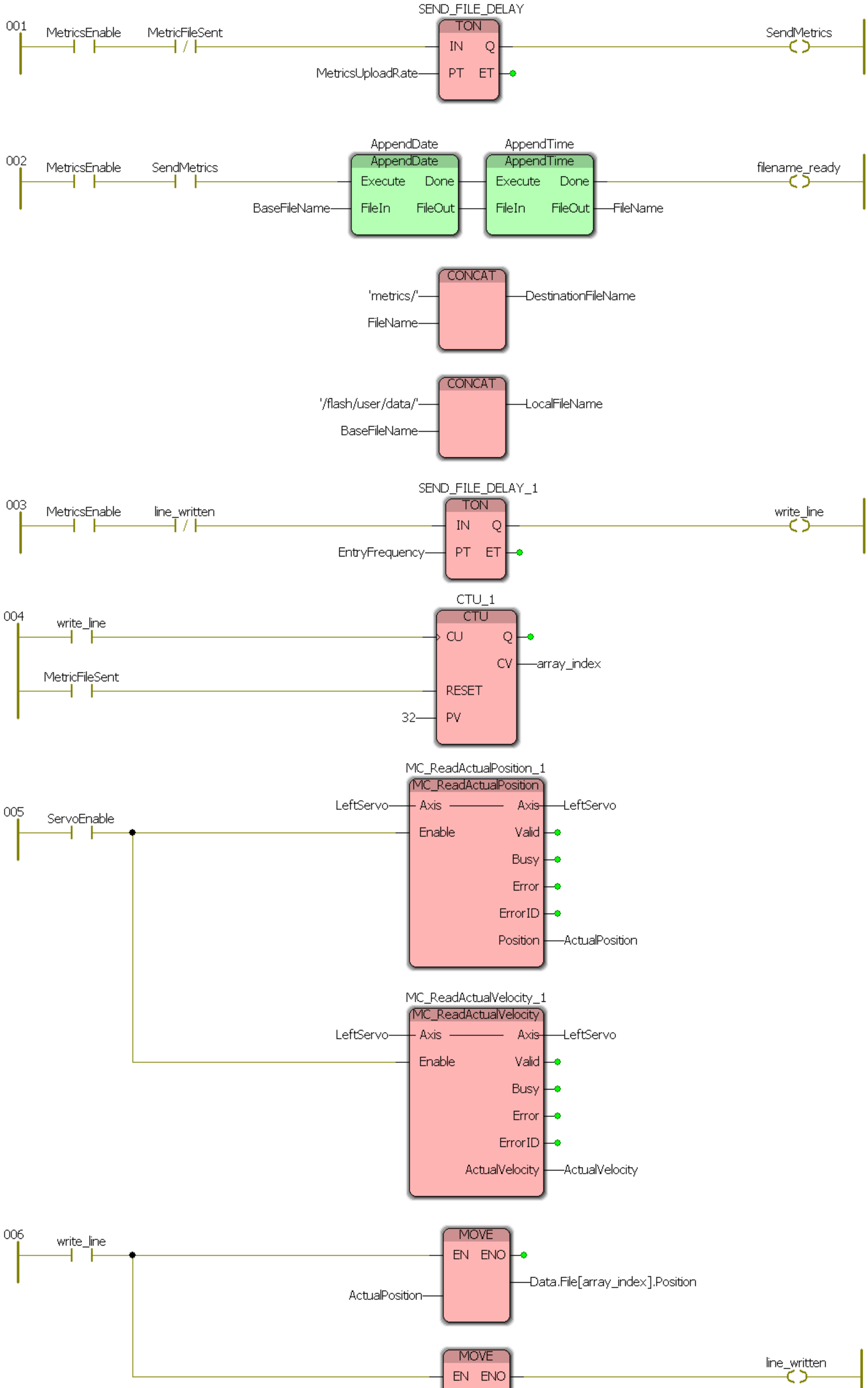
The same data configuration structure was used but there is no preset message for the file as it will be created dynamically.

```
(* 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.

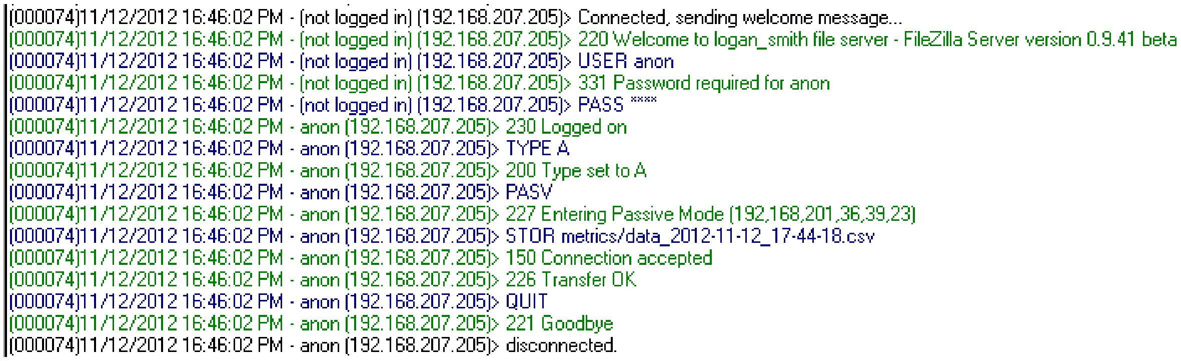


Controlling this example is very simple. The servo is turned on by "ServoEnable" which then in turn starts the jog at a constant velocity. The rest of the example is controlled in the main program:

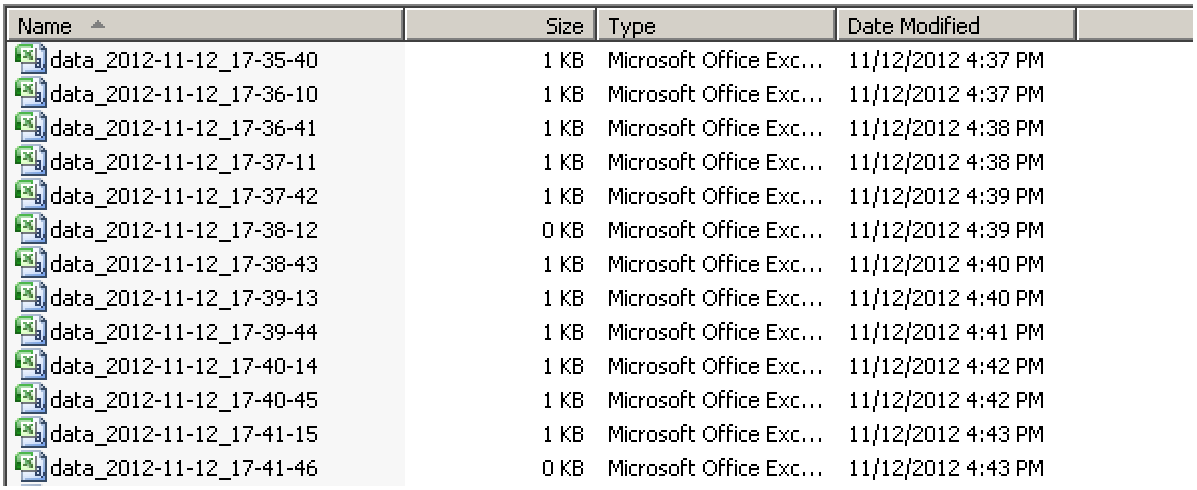

001
MetricsEnable
MetricFileSent
SEND_FILE_DELAY
TON
IN
Q
PT
ET
MetricsUploadRate
SendMetrics
002
MetricsEnable
SendMetrics
AppendDate
AppendDate
Execute
Done
FileIn
FileOut
BaseFileName
AppendTime
AppendTime
Execute
Done
FileIn
FileOut
FileName
filename_ready
CONCAT
'metrics/'
FileName
DestinationFileName
CONCAT
'/flash/user/data/'
BaseFileName
LocalFileName
003
MetricsEnable
line_written
SEND_FILE_DELAY_1
TON
IN
Q
PT
ET
EntryFrequency
write_line
004
write_line
MetricFileSent
CTU_1
CTU
CU
Q
CV
RESET
PV
32
array_index
005
ServoEnable
MC_ReadActualPosition_1
MC_ReadActualPosition
LeftServo
Axis
Axis
LeftServo
Enable
Valid
Busy
Error
ErrorID
Position
ActualPosition
MC_ReadActualVelocity_1
MC_ReadActualVelocity
LeftServo
Axis
Axis
LeftServo
Enable
Valid
Busy
Error
ErrorID
ActualVelocity
ActualVelocity
006
write_line
MOVE
EN
ENO
ActualPosition
Data.File[array_index].Position
MOVE
EN
ENO
line_written

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:

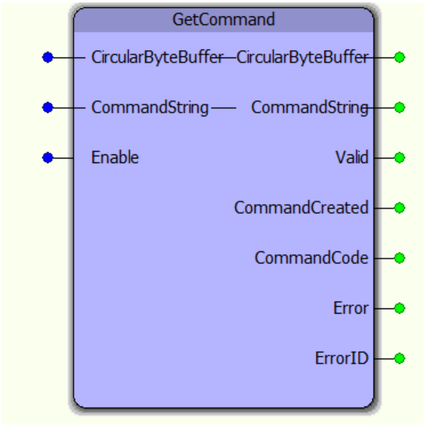


```
(000074)11/12/2012 16:46:02 PM - (not logged in) (192.168.207.205)> Connected, sending welcome message...
(000074)11/12/2012 16:46:02 PM - (not logged in) (192.168.207.205)> 220 Welcome to logan_smith file server - FileZilla Server version 0.9.41 beta
(000074)11/12/2012 16:46:02 PM - (not logged 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

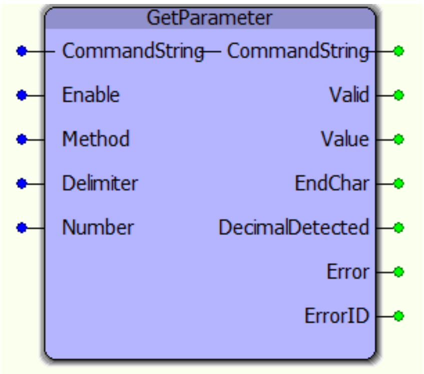
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	CircularByteBuffer	CircularBufferStruct	Structure containing a data buffer and other operational information required to manage the CircularByteBuffer.	
V	CommandString	CTB_CommandStruct	Input string containing at least two bytes of command characters 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	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.	

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

ErrorID	Meaning
0	No error.
10165	CommandString length is too long or command delimiter not found.

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	Delimiter	YTB_STRING1	String value of the character separating parameters within the CommandString.	BYTE#44 (",")
B	Number	INT	Depending on Method input, either the number of the parameter value to be found.	INT#0
B	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 paramter.	Method#Parameter
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.	
B	EndChar	INT	Last character index in byte buffer searched. When using Method#	
B	DecimalDetected	BOOL	Indicates if the value output contains a decimal or not. (useful to determine conversion to a Real or Integer 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.

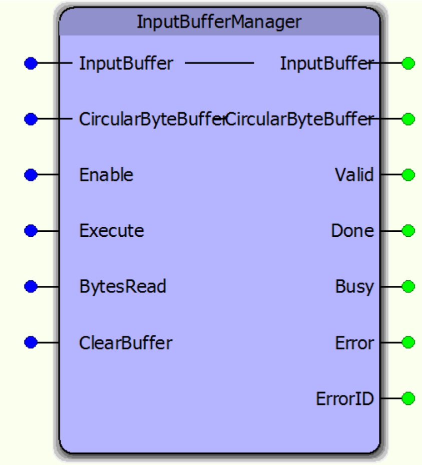
Notes

- 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 provided in ReName_CommandProcessor customization section.

Error Description

ErrorID	Meaning
0	No error.
10160	CommandString length is invalid.
10162	Parameter being searched for is out of range.
10163	Mode input not valid.
10164	Invalid character position input.

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	Clears all contents of the circular buffer and resets StorePointer and UsePointer	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.

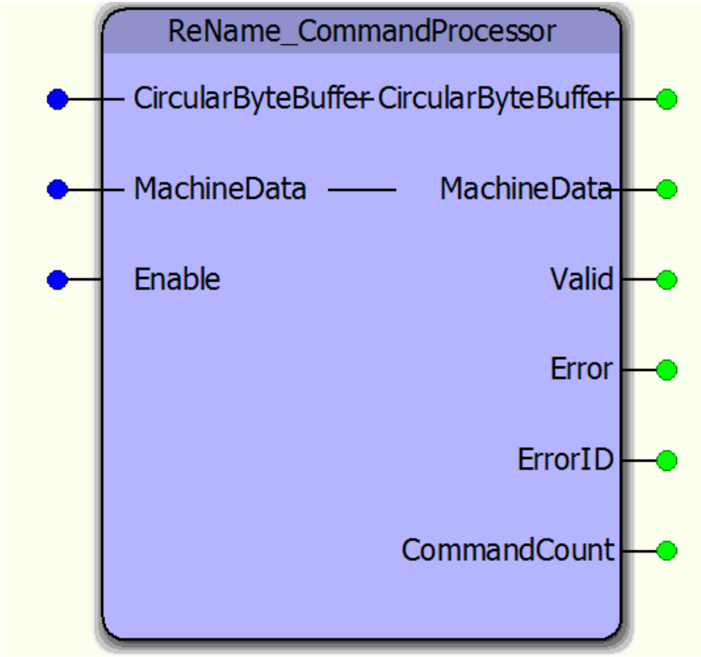
Notes

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

ErrorID	Meaning
0	No error.
10022	Product or circular buffer overrun / full.
10023	Buffer size too small / cannot be zero.

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 which will be used to operate the machine. To use this function, you must copy and paste into your main project, and rename it, and customize it.

Library

Comm Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	MachineData	MyMachineStruct	A user customizable structure containing machine data used in processing commands.	
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
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.	

V	CommandCount	UDINT	Number of commands that have been processed since this function block 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.
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 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.

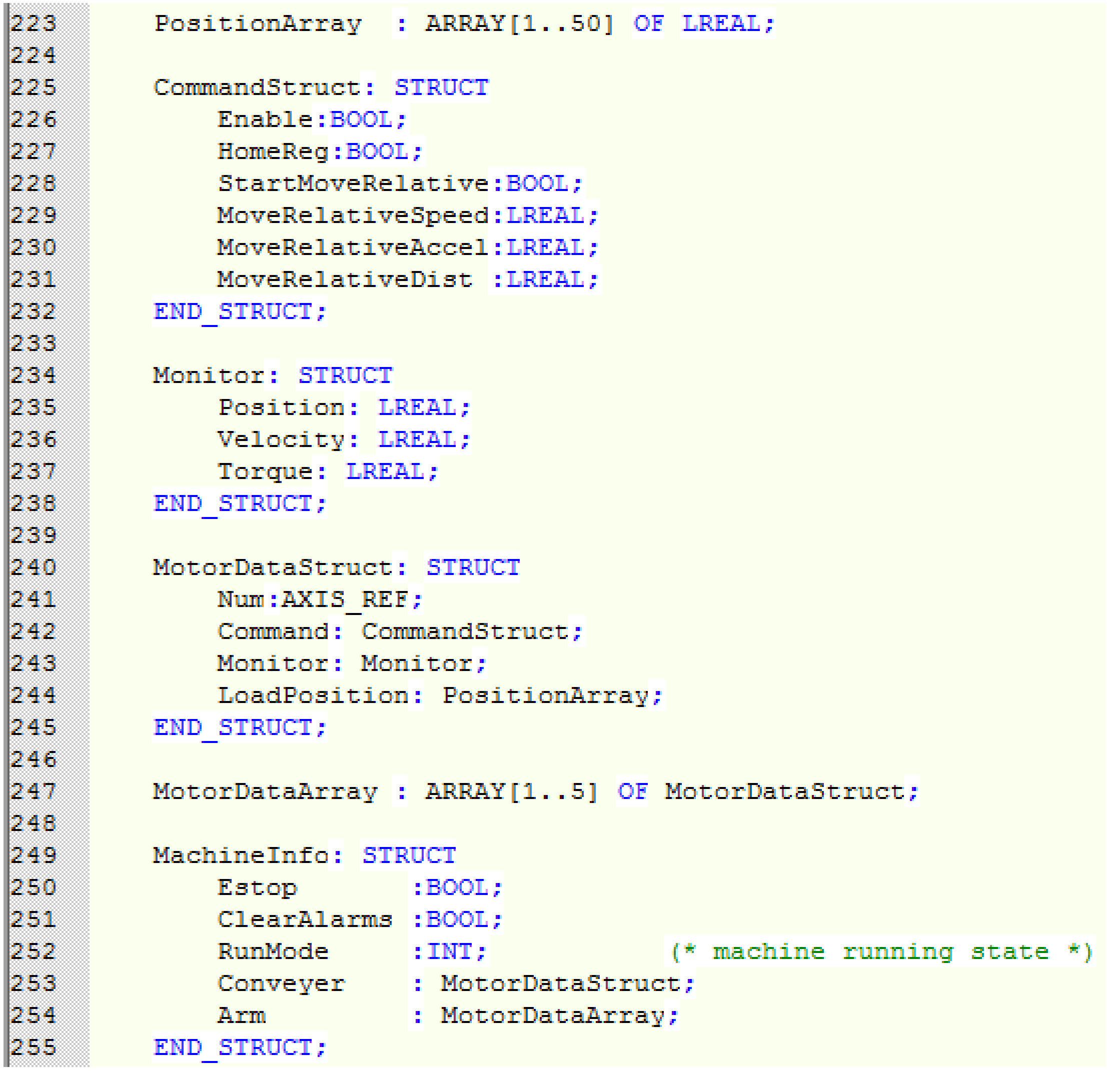
Error Description

ErrorID	Meaning
0	No error.
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.

Customization Steps

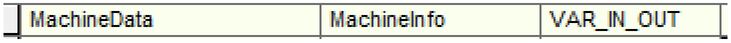
1. Copy this Function block from the Comm Toolbox, paste it into your project, and rename with a different (but similar) name.
2. Data type MyMachineStruct (VAR_IN_OUT ‘MachineData’) is only an example structure. A custom structure must be designed to uniquely match the needs of

the application. An example is shown below.



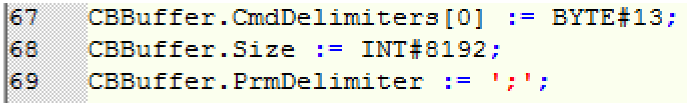
```
    PositionArray  : ARRAY[1..50] OF LREAL;

    CommandStruct: STRUCT
        Enable:BOOL;
        HomeReg:BOOL;
        StartMoveRelative:BOOL;
        MoveRelativeSpeed:LREAL;
        MoveRelativeAccel:LREAL;
        MoveRelativeDist :LREAL;
    END_STRUCT;

    Monitor: STRUCT
        Position: LREAL;
        Velocity: LREAL;
        Torque: LREAL;
    END_STRUCT;

    MotorDataStruct: STRUCT
        Num:AXIS_REF;
        Command: CommandStruct;
        Monitor: Monitor;
        LoadPosition: PositionArray;
    END_STRUCT;

    MotorDataArray : ARRAY[1..5] OF MotorDataStruct;

    MachineInfo: STRUCT
        Estop          :BOOL;
        ClearAlarms    :BOOL;
        RunMode        :INT;             (* machine running state *)
        Conveyer       : MotorDataStruct;
        Arm            : MotorDataArray;
    END_STRUCT;
```

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



MachineData	MachineInfo	VAR_IN_OUT

4. Initialize the configuration elements in CircularByteBuffer.

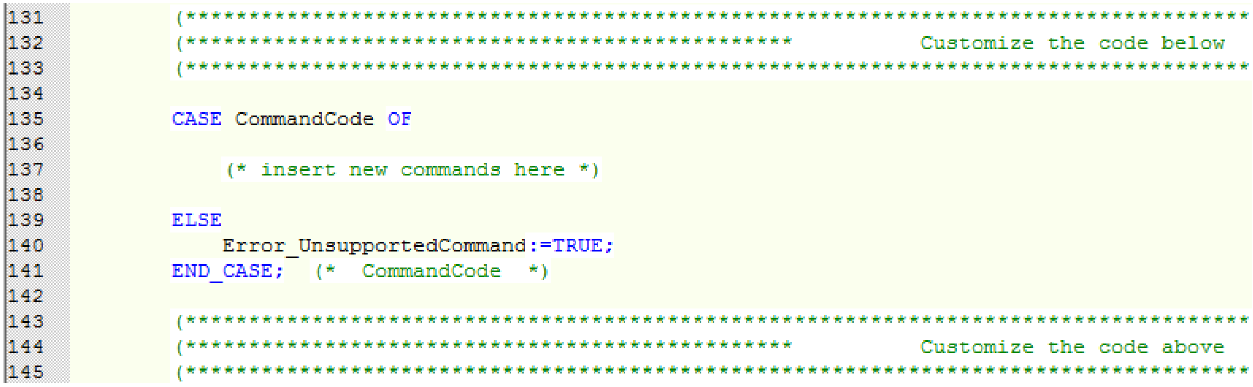


```
CBBuffer.CmdDelimiters[0] := BYTE#13;
CBBuffer.Size := INT#8192;
CBBuffer.PrmDelimiter := ';';
```

1. a. 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.
2. b. 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).

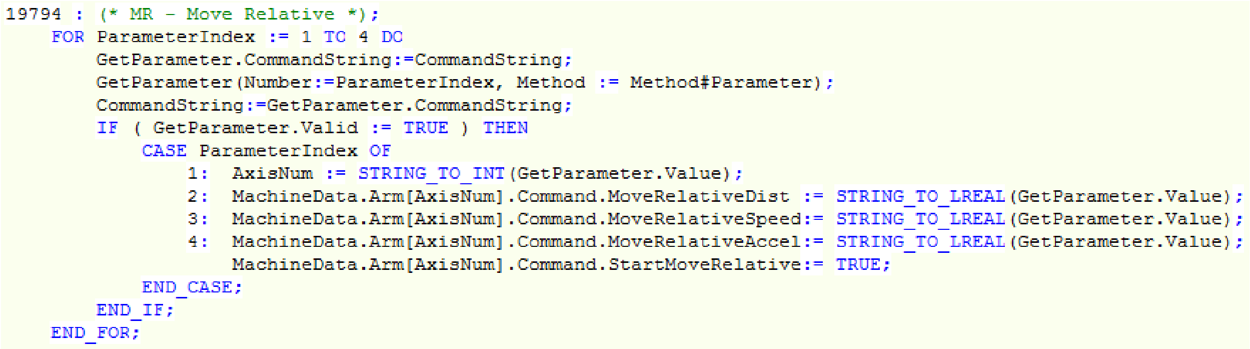
    3. c. Size must represent the defined size of the DataType definition for the CircularBufferStruct’s “Data? Element. If Size not specified, it will default to zero and the InputBufferManager function block will cause an error. Normally, this value is 8192 as the structure definition is in the Comm Toolbox itself. If this must be increased for any reason, modify the Comm Toolbox DataType definition and set the Size input accordingly.

5. Locate the comments “Customize the code below? and “Customize the code above?
6. Remove example commands to avoid potential errors in operation.



```
(*******************************************************************************
(*********************************************             Customize the code below
(*******************************************************************************

CASE CommandCode OF

    (* insert new commands here *)

ELSE
    Error_UnsupportedCommand:=TRUE;
END_CASE;  (*  CommandCode  *)

(*******************************************************************************
(*********************************************             Customize the code above
(*******************************************************************************
```

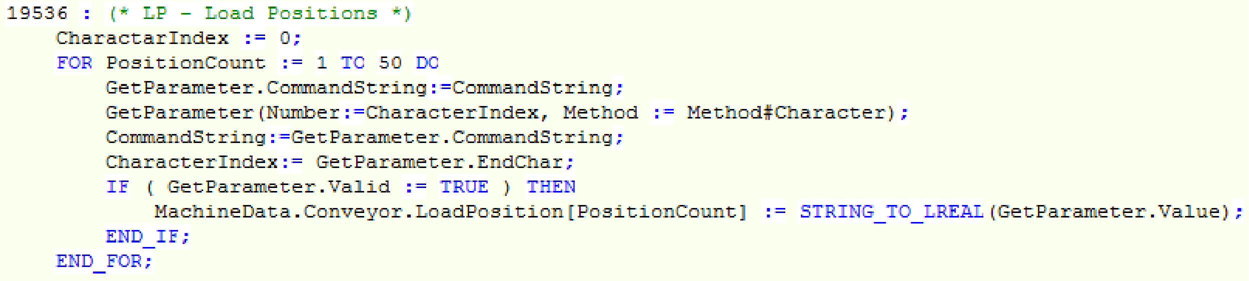
7. Add your commands. Two examples are shown below:
    1. Move Relative command
        1. MR,<axisnumber>,<distance>,<speed>,<accel/decel>
        2. Calculate the CommandCode which corresponds to the ASCII characters ‘MR’. The equation is: CHAR_TO_INT(‘M’) * 256 + CHAR_TO_INT(‘R’) = 19794.
        3. Add the CommandCode to the case statement.
        4. 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;
```

    2. Load Positions command
        1. LP,<Position1>,<Position2>,…,<Position50>
        2. Calculate the CommandCode which corresponds to the ASCII characters ‘LP’. The equation is: CHAR_TO_INT(‘L’) * 256 + CHAR_TO_INT(‘P’) = 19536
        3. 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?

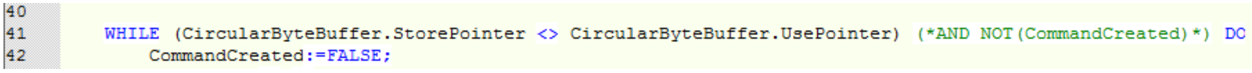


```
19536 : (* LP - Load Positions *)
    CharactarIndex := 0;
    FOR PositionCount := 1 TO 50 DO
        GetParameter.CommandString:=CommandString;
        GetParameter(Number:=CharacterIndex, Method := Method#Character);
        CommandString:=GetParameter.CommandString;
        CharacterIndex:= GetParameter.EndChar;
        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



```

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

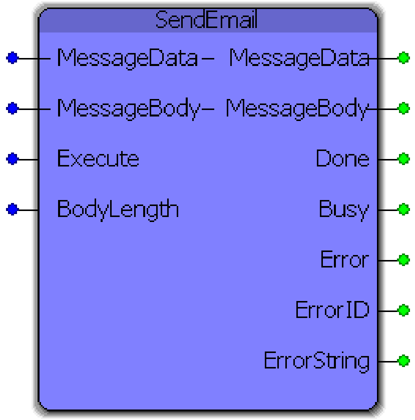
ReName_CommunicationsMgr

ReName_CommunicationsMgr is a reference POU showing the recommended setup of the command stream features.

Customization Required:

1. Find the ReNameCommandProcessor Function Block and change the DataType of MachineData VAR_IN_OUT in accordance with a custom structure that you will create for your application.
2. The only other area that may require customization is located under the comment “Prepare to create the Response Output for the Command Channel”. Once a connection has been established, the Y_WriteDevice function block can be used to send a buffer of data (monitor information or command responses for example) back to the device issuing commands.

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	YC_ 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 inputs are read and the e-mail(s) is sent. To resend the e-mail or send a different file, change the value(s) 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	YC_ 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

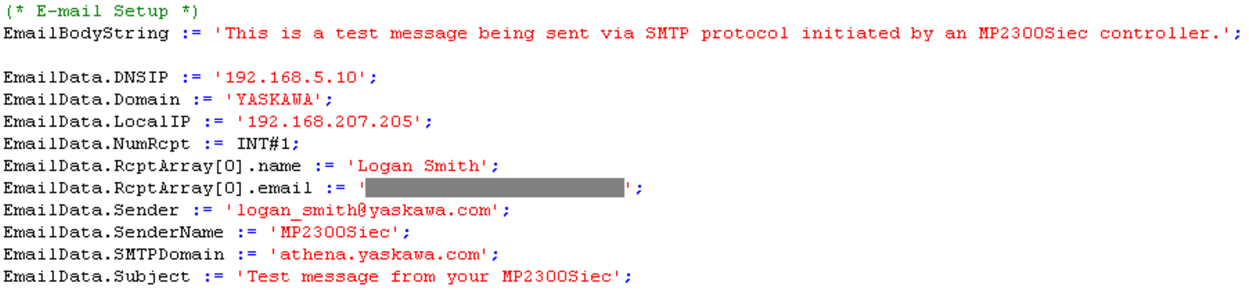
ErrorID	Meaning
0	No error.
8705	The maximum number of concurrently open user sockets/IO device handles has been reached or exceeded.
8706	The socket/IO device handle was invalid. Invalid IP address.
8707	The IP address string was not in a valid format.
8708	The socket/IO device handle could not be created.
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.
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.
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.
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.

Example

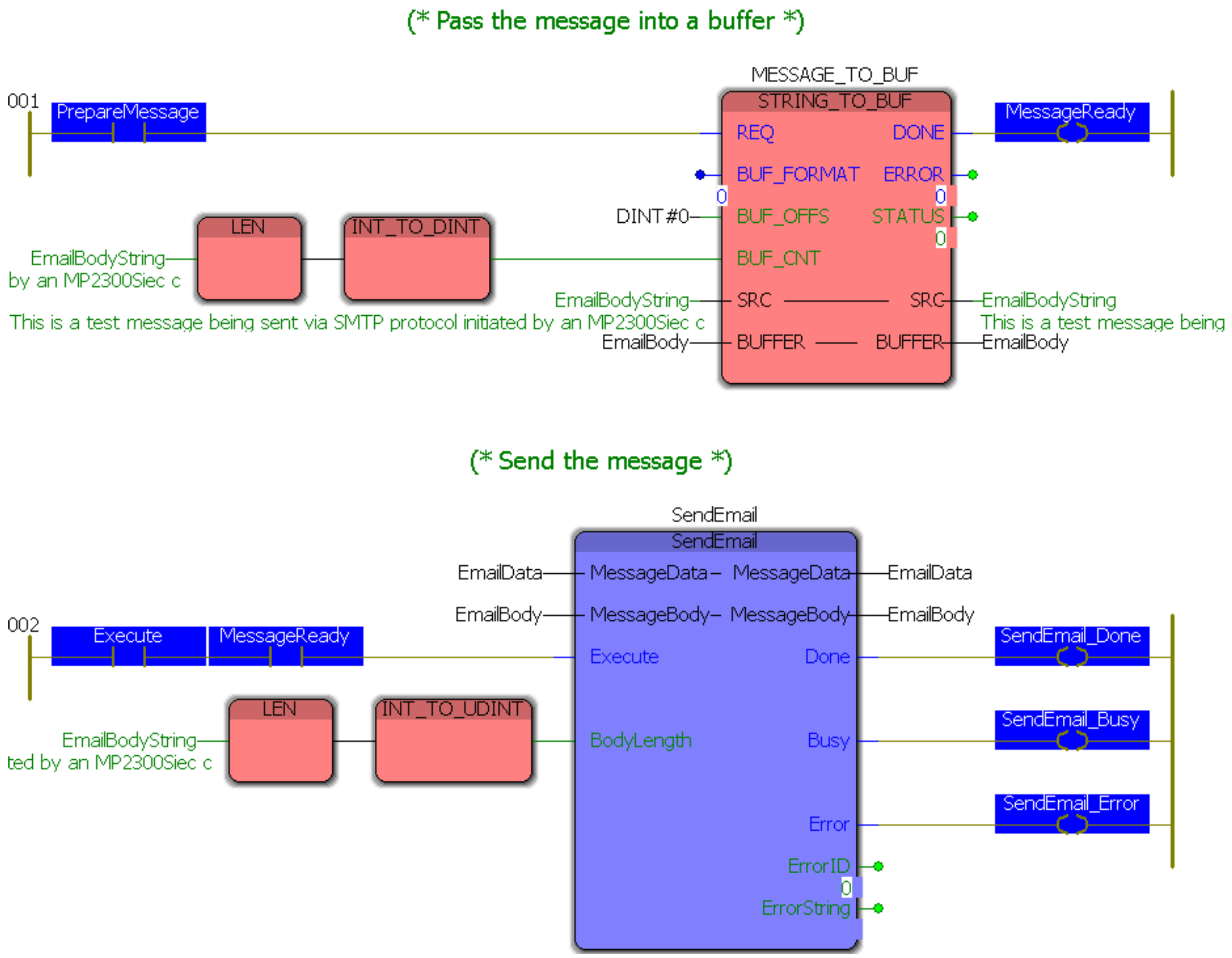
As this is a complicated function, additional examples are provided in separate help files listed under "Additional Examples" and prefixed with "SMTP_". The example shown here sets up the block, creates a message body and sends an e-mail to external Gmail account.

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

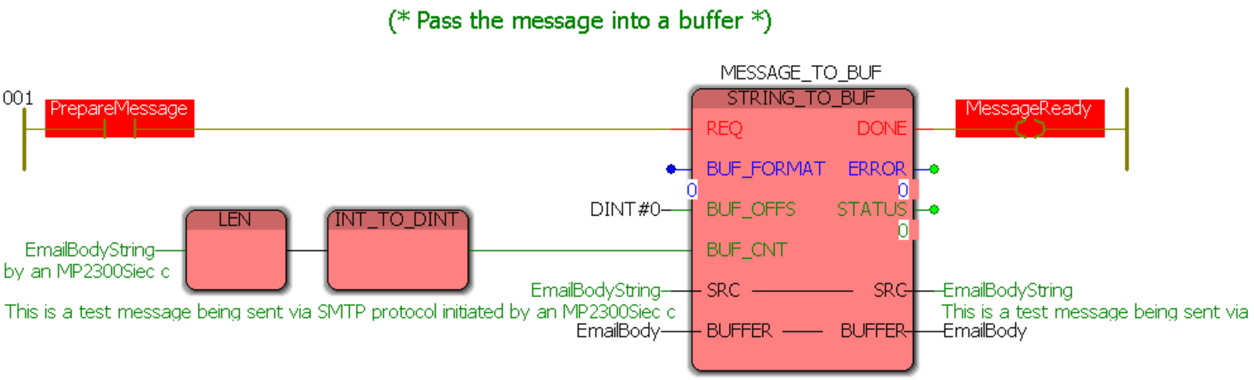


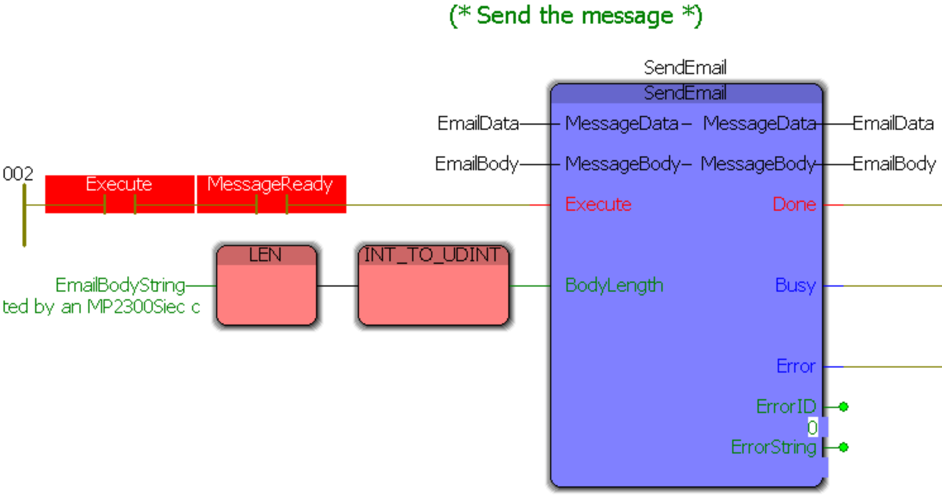
```
(* E-mail Setup *)
EmailBodyString := 'This is a test message being sent via SMTP protocol initiated by an MP2300Siec controller.';

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 := '[redacted]';
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.

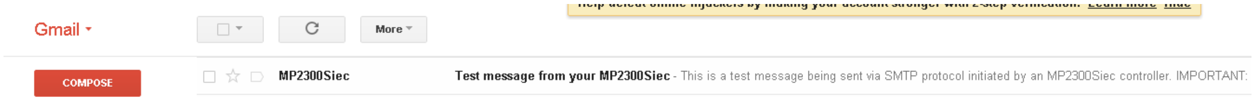


After toggling PrepareMessage, here is the result.



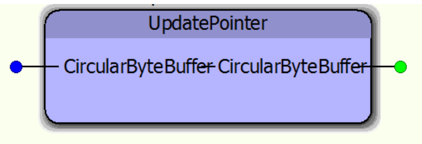


And to demonstrate the end result, here is the e-mail in the inbox of the Gmail account used. The sender and subject are both listed correctly and a portion of the send message can be seen.



MP2300Siec Test message from your MP2300Siec - This is a test message being sent via SMTP protocol initiated by an MP2300Siec controller. IMPORTANT:

UpdatePointer



The UpdatePointer function block is a supporting function block referenced by the GetCommand function block. It updates the UsePointer of 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.

File Read Write Toolbox

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Getting Started with File Read / Write Toolbox

Requirements for v301

To use the File Read / Write Toolbox, your 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 (v301 or higher)

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

The four main functions in this library are:

- Write_Binary_File
- Write_CSV_File
- Read_Binary_File
- Write_CSV_File

To use any of 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_DataTypes

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

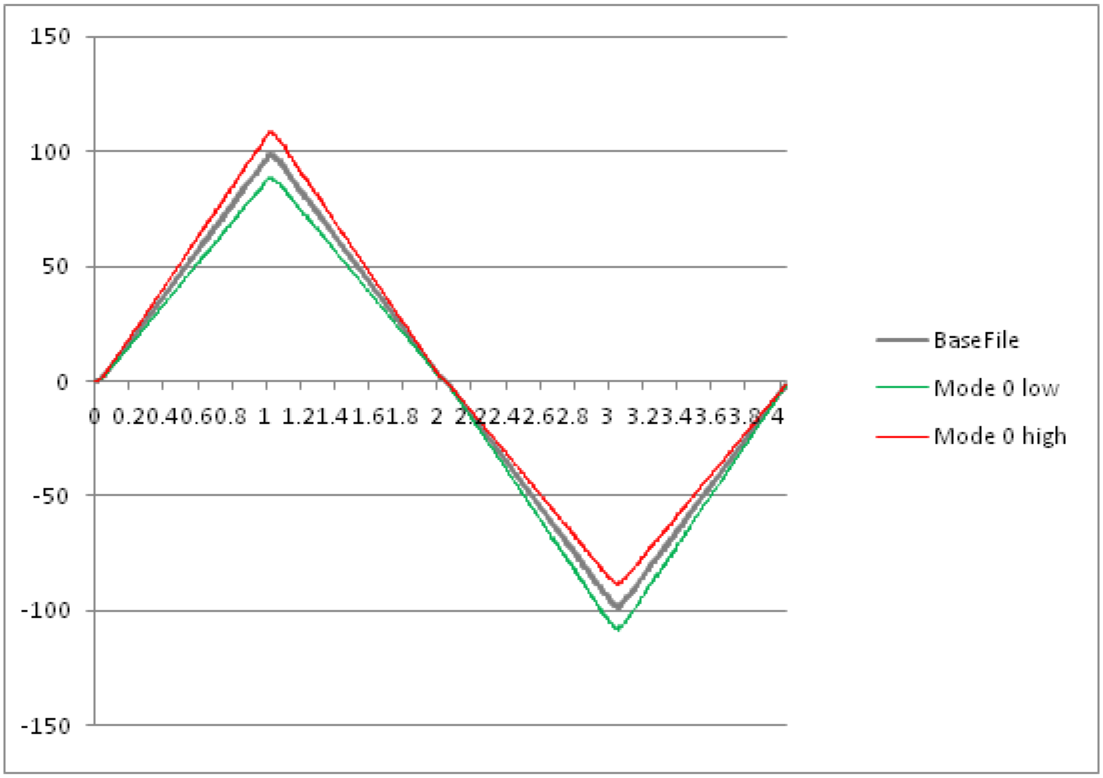
Enumerated Types in the File R/W Toolbox

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 zero, the second to 1, etc.

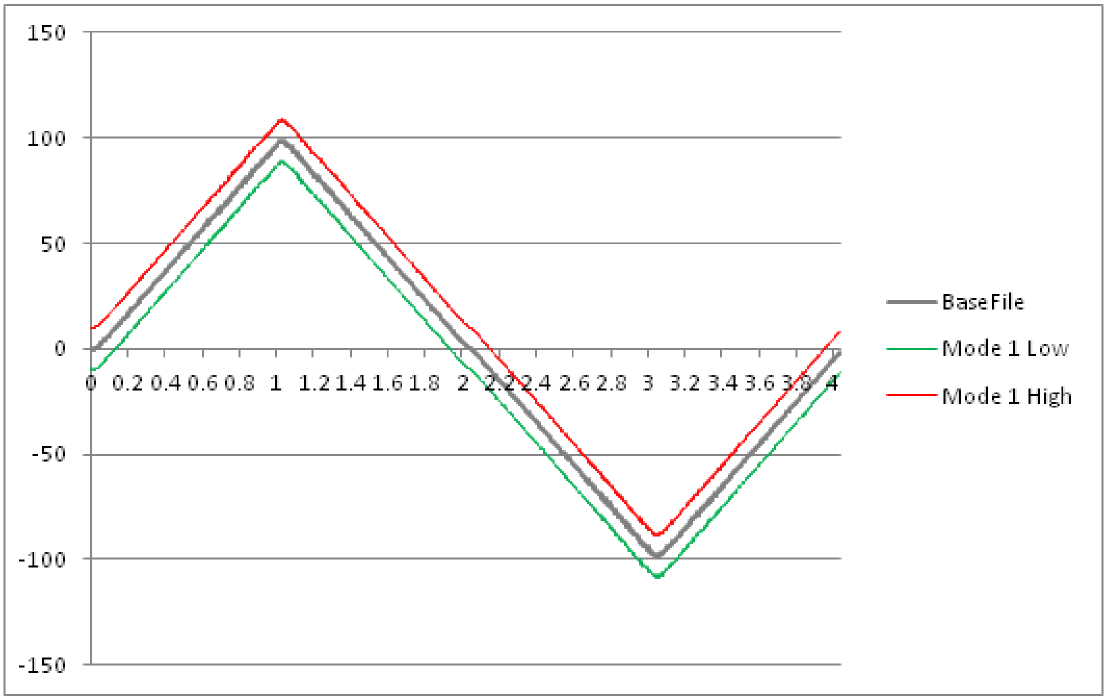
Enumerated Types Declaration

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.

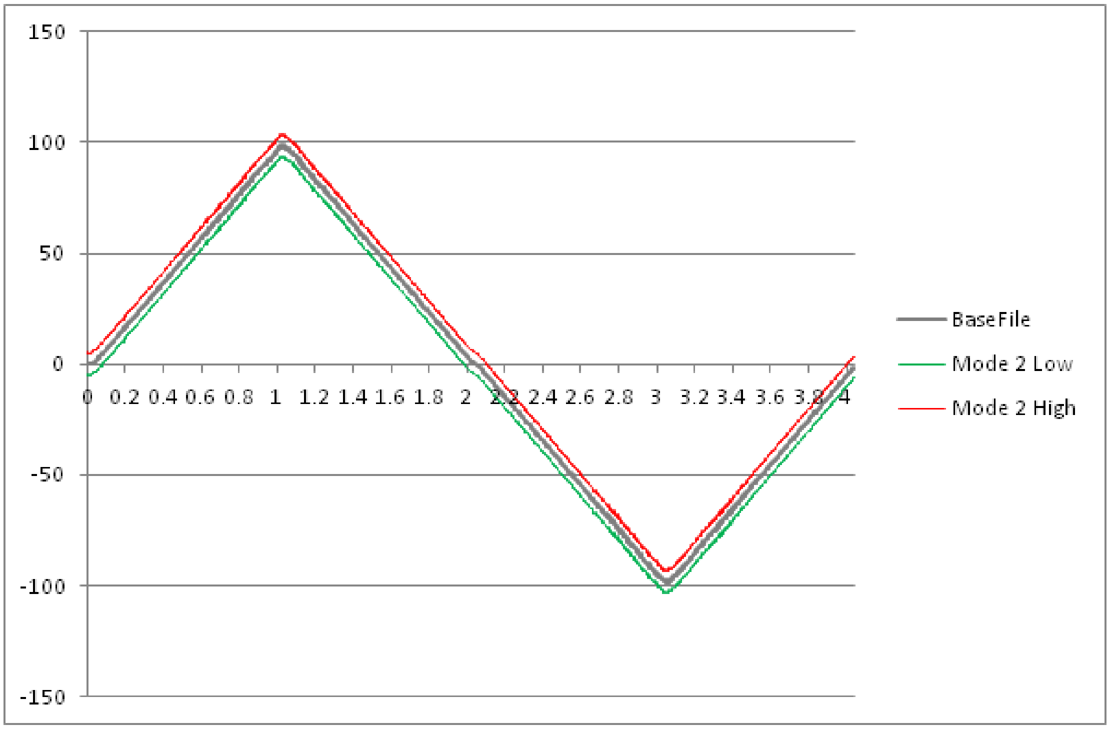
Percentage



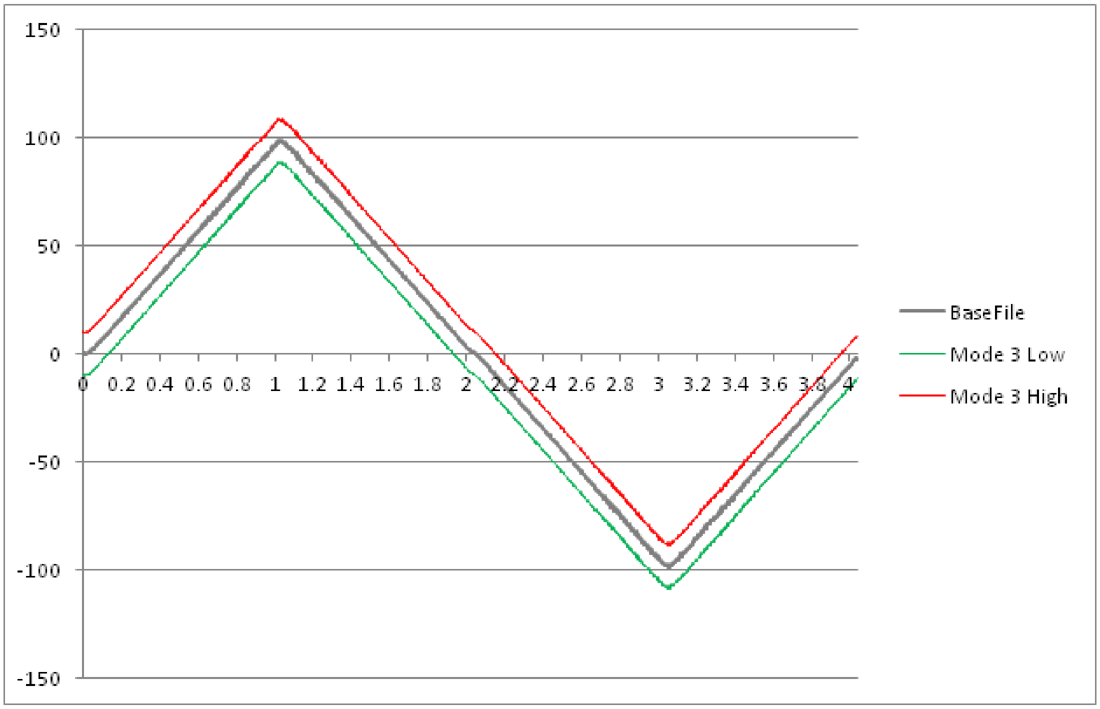
PercentOfMax



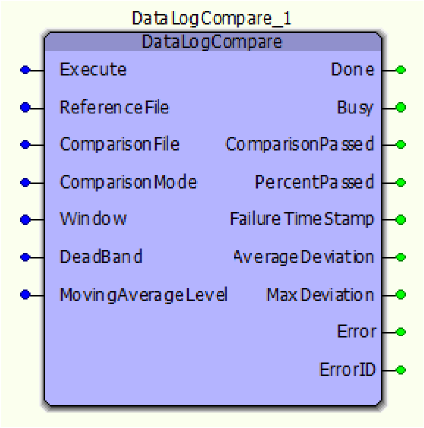
PercentOfAverage


150
100
50
0
-50
-100
-150
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4
3.6
3.8
4
BaseFile
Mode 2 Low
Mode 2 High

RawWindow


150
100
50
0
-50
-100
-150
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4
3.6
3.8
4
BaseFile
Mode 3 Low
Mode 3 High

DataLogCompare



This function block will read a ComparisonFile and determine how much of the data is within a specified window of the ReferenceFile.

Library

File RW 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
V	ReferenceFile	STRING	File used as reference to compare against. Example STRING#'flash/user/data/mylog.csv'	STRING#''
V	ComparsionFile	STRING	File to be characterized for determining pass / fail. Example STRING#'flash/user/data/mylog.csv'	STRING#''
V	ComparsionMode	ComparisonMode	Different modes of comparison, changes the calculation used to determine if ComparisonValue within range of ReferenceFile. (example graphs below)	ComparisonMode#Percentage
V	Window	LREAL	Allowable deviation from ReferenceFile log the ComparisonFile log to be within and still pass comparison.	LREAL#0.0

V	DeadBand	LREAL	If the base value is within the deadband – no comparison will be made.	LREAL#0.0
V	MovingAverageLevel	INT	Applies a moving average filter to both the ReferenceFile and ComparisonFile before comparing the values – useful when comparing logs of noisy data such as Torque. Set the size of the moving average filter from 0 – 1000.	INT#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)	
V	ComparisonPassed	BOOL	ComparisonFile is within allowable window of the ReferenceFile 100% of the time.	
V	PercentPassed	LREAL	Percentage of ComparisonFile data points within allowable range of ReferenceFile.	
V	FailureTimeStamp	LREAL	If the comparison failed, this is the first time at which the ComparsionFile deviated from the ReferenceFile	
V	AverageDeviation	LREAL	The average deviation of the ComparisonFile from the ReferenceFile. (In modes Precent, PercentOfMax, and PercentOfAverage this value is a percentage, in RawWindow mode this is a raw difference)	
V	MaxDeviation	LREAL	The largest deviation of the ComparisonFile from the ReferenceFile. (In modes Precent, PercentOfMax, and PercentOfAverage this value is a percentage, in RawWindow mode this is a raw difference.)	
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

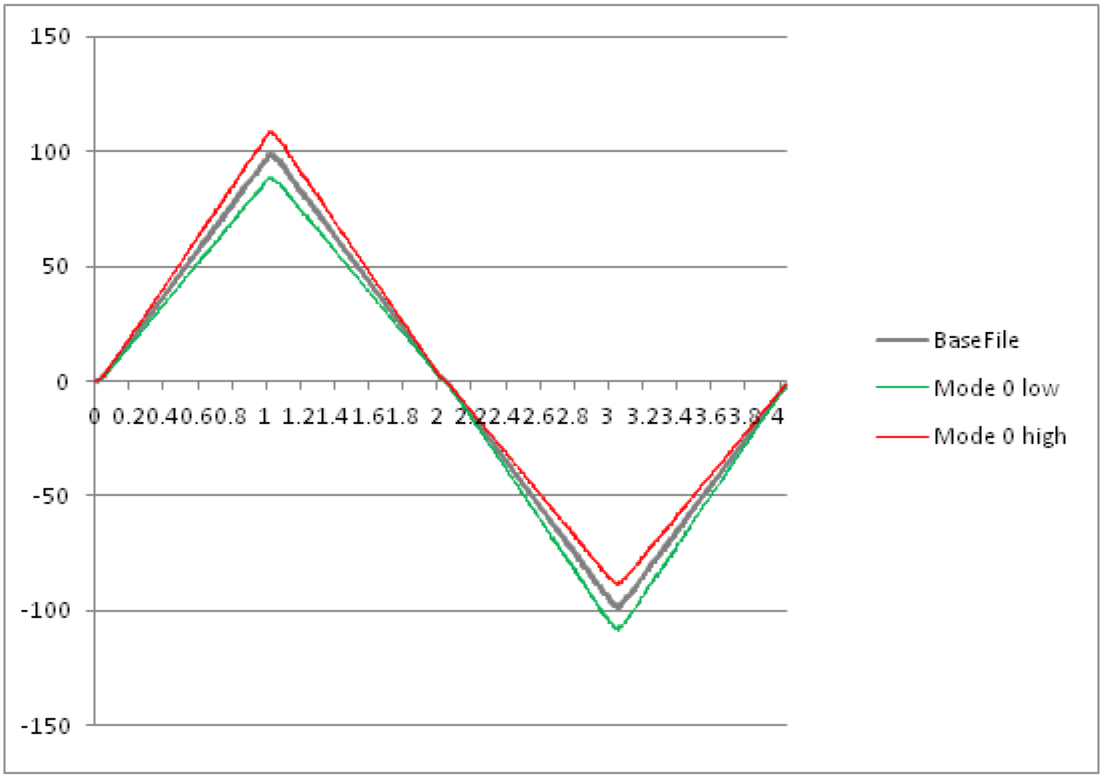
- Don't forget to include the ProConOS firmware library in the project. It is required for this function block.

Error Description

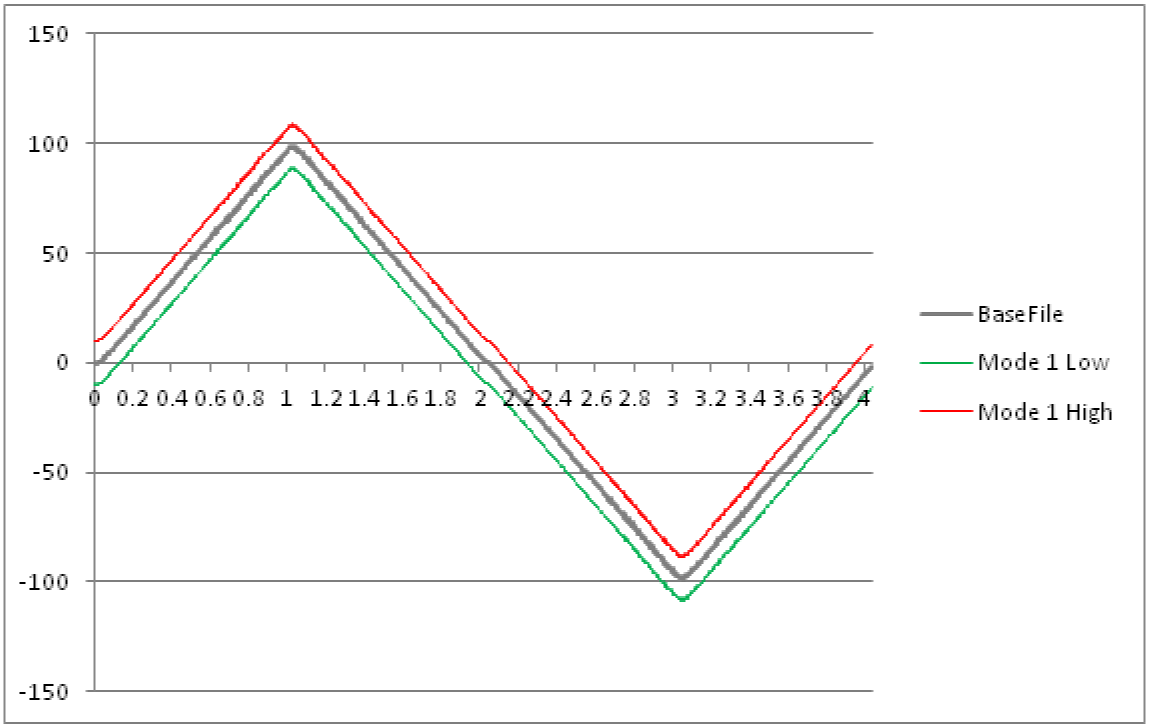
ErrorID	Meaning
0	No error.
4	File is already open.
5	File is opened, write protected or access denied.
6	File name not defined.
10	End of data reached.
12	The number of characters to be read is greater than the data buffer.
13	Invalid positioning mode or position specified is before the beginning of the file.
20	File could not be closed.
22	No data could be read.
24	Position could not be set.
11000	Header read error. Files being compared must have a header of "TimeStamp,Data". The function block GenerateDataLog automatically creates this header.
11001	Header mismatch error. Files being compared have a different header.
11002	File Name Error. File could not be read.
11003	Window Error. Window cannot be less then or equal to zero
11004	Timing mismatch. Time stamp in Reference file and Comparsion file do not allign. Data was possibly recoreded at different intervals
11005	Max Cycle Error.
11006	DeadBand value must be greater than or equal to zero.
11007	Invalid mode selection
11008	Moving average level must be greater than or equal to zero.
11009	Moving average level must be smaller than 1000.
11010	Was unable to pre load the comparison data buffer.
11011	Unable to load the reference data file fast enough. Comparison unable to complete.

Comparison Modes:

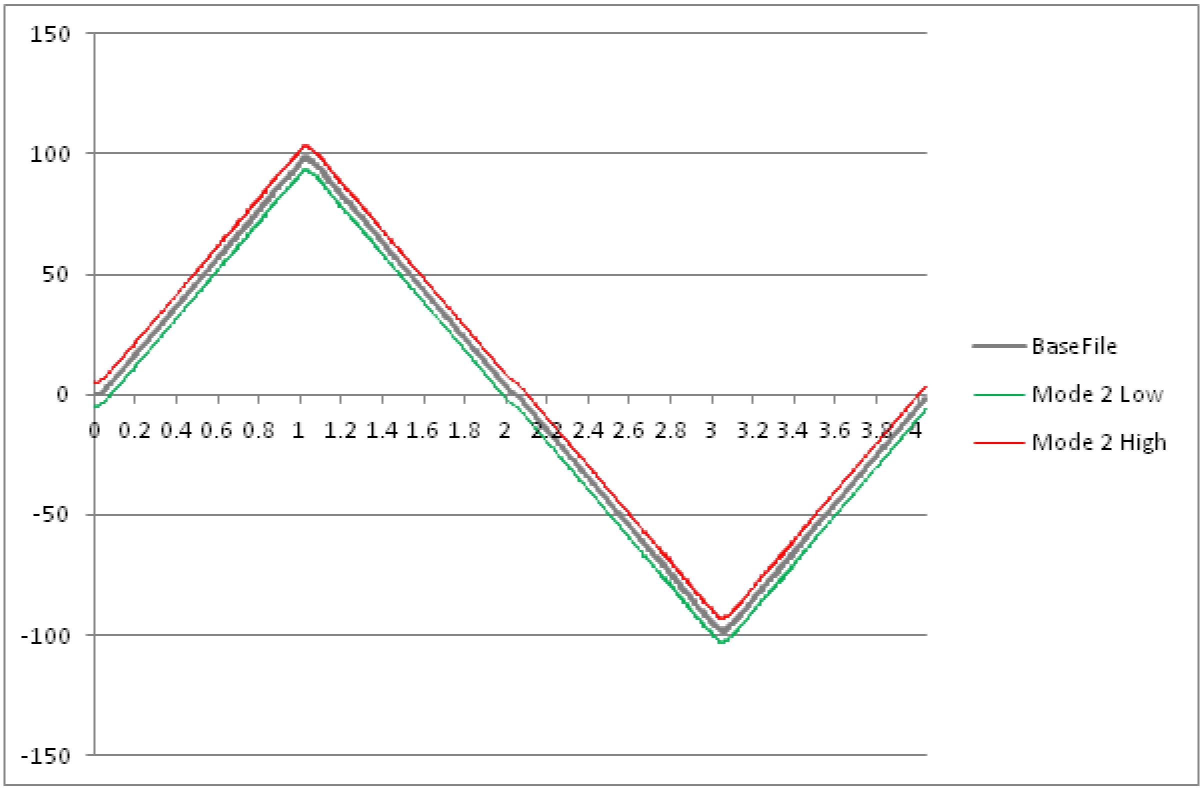
ComparisonMode#Percentage :



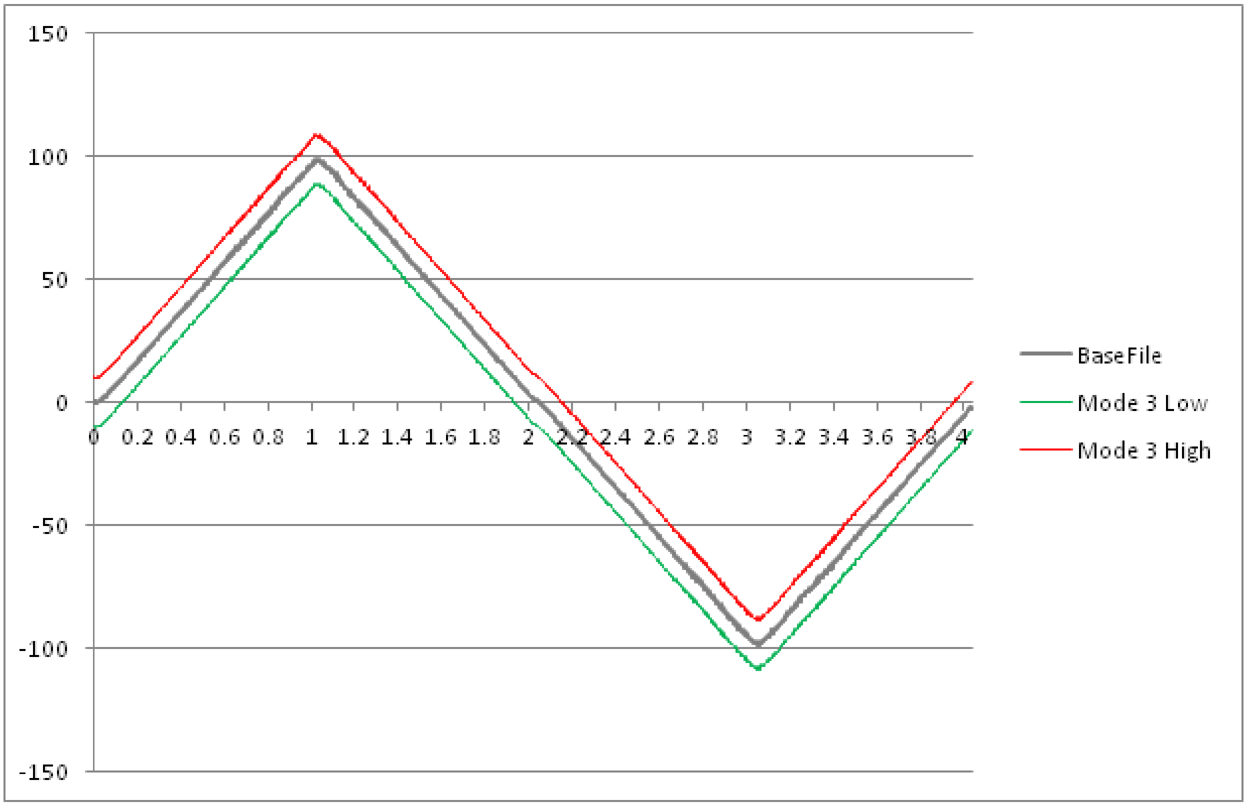
ComparisonMode#PercentOfMax :



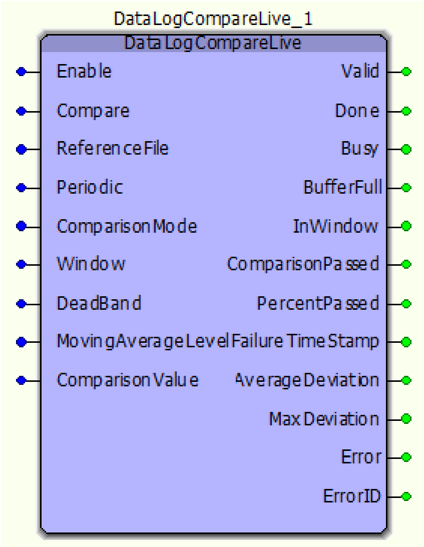
ComparisonMode#PercentOfAverage :



ComparisonMode#RawWindow :


150
100
50
0
-50
-100
-150
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4
3.6
3.8
4
BaseFile
Mode 3 Low
Mode 3 High

DataLogCompareLive



This function block will read in a Comparison File and determine how much of the data is within a specified window of the input Reference File.

Library

File RW 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	ReferenceFile	STRING	File used as reference to compare against. Example STRING#'flash/user-/data/mylog.csv'	STRING#''
V	Periodic	BOOL	When False, enable will only load the reference buffer for one comparison and hold outputs as long as Enable is true. When True, as long as enable is high, the reference buffer will be loaded continuously and outputs will be held as long as Compare is True. (Error and ErrorID being the exception, Enable will need to be toggled to clear errors still)	FALSE
V	ComparsionMode	ComparisonMode	Different modes of comparison,	ComparisonMode#Percentage

			changes the calculation used to determine if ComparisonValue within range of ReferenceFile. (example graphs below)	
V	Window	LREAL	Allowable deviation from ReferenceFile log the ComparisonFile log to be within and still pass comparison.	LREAL#0.0
V	DeadBand	LREAL	If the base value is within the deadband – no comparison will be made.	LREAL#0.0
V	MovingAverageLevel	INT	Applies a moving average filter to both the ReferenceFile and ComparisonFile before comparing the values – useful when comparing logs of noisy data such as Torque. Set the size of the moving average filter from 0 – 1000.	INT#0
V	Comparison Value	LREAL	Value to be compared to the Reference file.	LREAL#0.0
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)	
V	BufferFull	BOOL	Indicates that buffer of values read in from the BaseFile is full. To ensure that the comparison will complete it is recommended to wait until this signal is True before beginning a comparison.	
V	InWindow	BOOL	Indicates that the current Value being input is within the specified window of the BaseFile. This is a live indicator of how the comparison is doing.	
V	ComparisonPassed	BOOL	ComparisonFile is within allowable window of the ReferenceFile 100% of the time.	
V	PercentPassed	LREAL	Percentage of ComparisonFile data points within allowable range of ReferenceFile.	
V	FailureTimeStamp	LREAL	If the comparison failed, this is the first time at which the ComparsionFile deviated from the ReferenceFile	
V	AverageDeviation	LREAL	The average deviation of the ComparisonFile from the ReferenceFile. (In modes Precent, PercentOfMax, and PercentOfAverage this value is a percentage, in RawWindow mode this is a raw difference)	
V	MaxDeviation	LREAL	The largest deviation of the ComparisonFile	

			from the ReferenceFile. (In modes Precent, PercentOfMax, and PercentOfAverage this value is a percentage, in RawWindow mode this is a raw difference.)
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

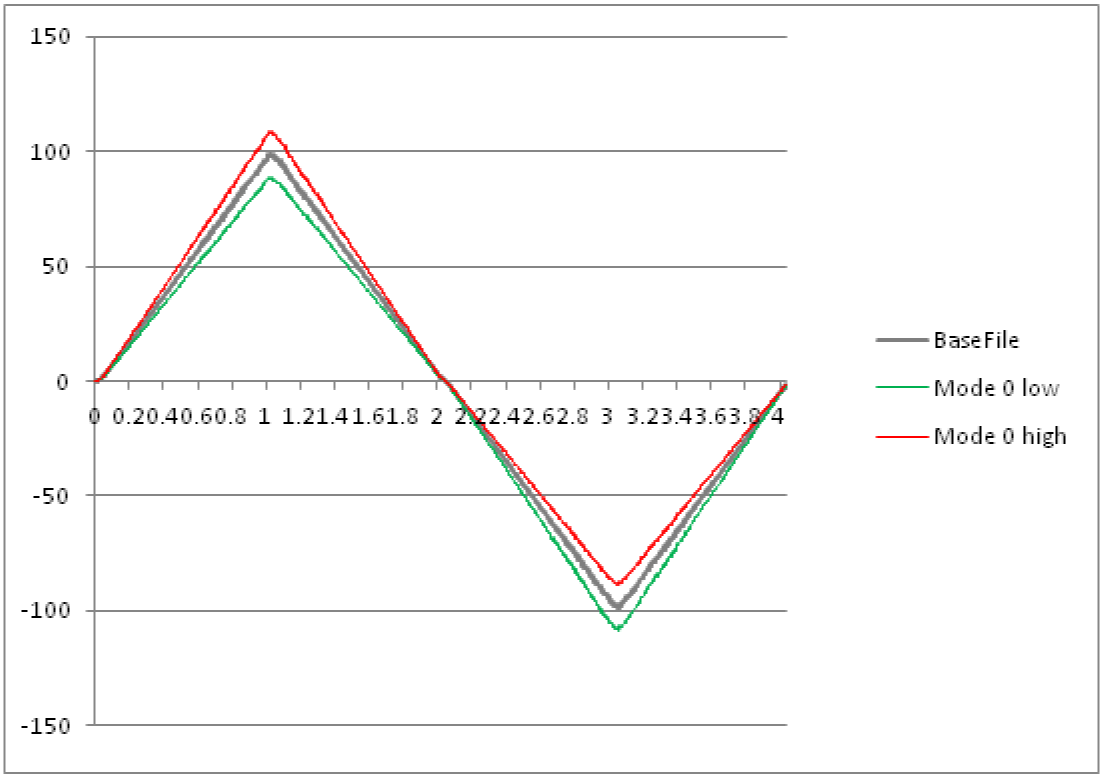
- Don't forget to include the ProConOS firmware library in the project. It is required for this function block.
- This function block requires that the variable PLC_SYS_TICK_CNT be defined as a global variable corresponding to a specified hardware address. This variable is automatically created and added to the System Variables when a new template project is opened.
- There is a known bug that if the ReferenceFile has < 1000 samples, setting periodic mode = TRUE will always cause the function block to error.

Error Description

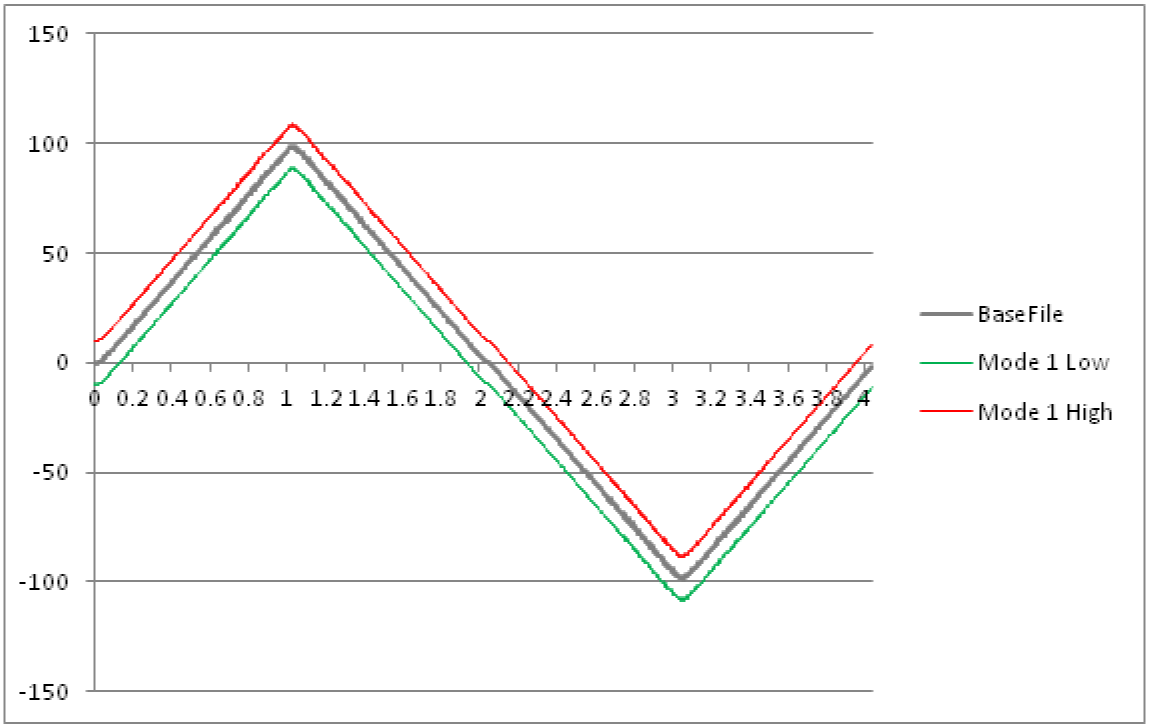
ErrorID	Meaning
0	No error.
4	File is already open.
5	File is opened, write protected or access denied.
6	File name not defined.
10	End of data reached.
12	The number of characters to be read is greater than the data buffer.
13	Invalid positioning mode or position specified is before the beginning of the file.
20	File could not be closed.
22	No data could be read.
24	Position could not be set.
11000	Header read error. Files being compared must have a header of "TimeStamp,Data". The function block GenerateDataLog automatically creates this header.
11001	Header mismatch error. Files being compared have a different header.
11002	File Name Error. File could not be read.
11003	Window Error. Window cannot be less then or equal to zero
11004	Timing mismatch. Time stamp in Reference file and Comparsion file do not allign. Data was possibly recoreded at different intervals
11005	Max Cycle Error.
11006	DeadBand value must be greater than or equal to zero.
11007	Invalid mode selection
11008	Moving average level must be greater than or equal to zero.
11009	Moving average level must be smaller than 1000.
11010	Was unable to pre load the comparison data buffer.
11011	Unable to load the reference data file fast enough. Comparison unable to complete.

Comparison Modes:

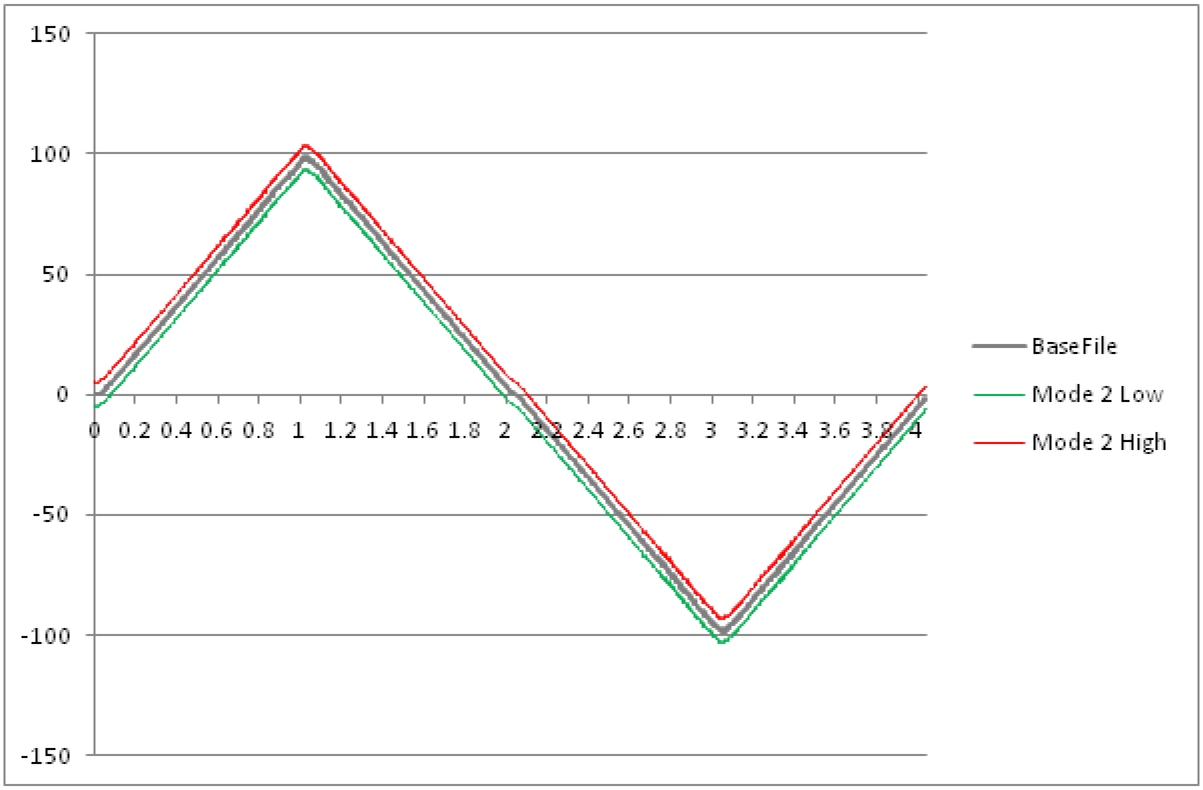
ComparisonMode#Percentage :



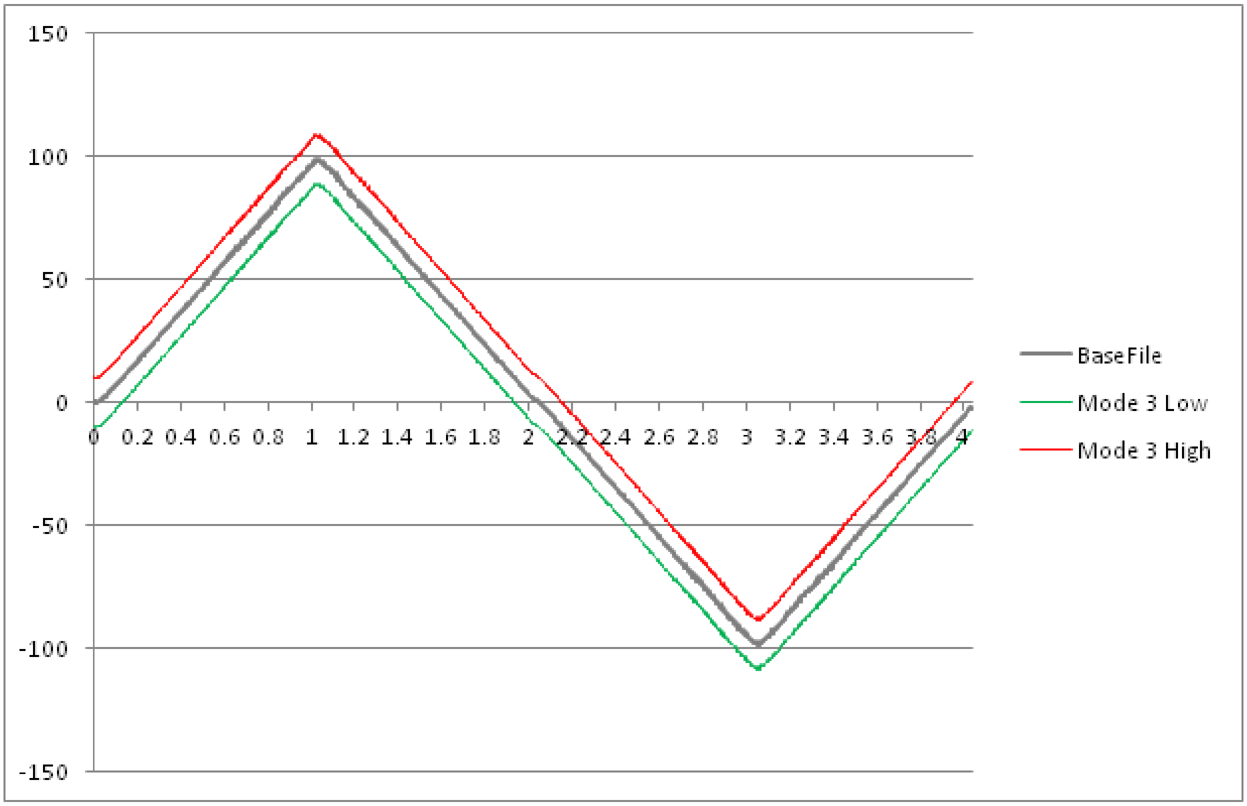
ComparisonMode#PercentOfMax :



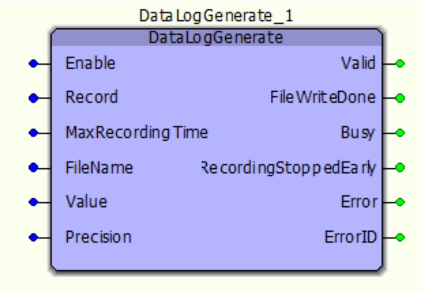
ComparisonMode#PercentOfAverage :



ComparisonMode#RawWindow :



DataLogGenerate



This function block will record a log file containing [TimeStamp, Value input] for a given duration. This CSV file can then be exported from the controller for post processing of the data logged.

Library

File RW 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	Record	BOOL	Will begin recording a new log when this input is set high. (this operation will delete previous log of same name if it exists on the controller). Logging stops when record input becomes false or MaxRecordingTime is reached.	FALSE
V	MaxRecordingTime	LREAL	Maximum length of time data will be logged for. If left unconnected, data will be logged as log as Record is TRUE.	LREAL#0.0
V	FileName	STRING	Name of file to be read or written. Example STRING#'/-flash/user/data/myFile.csv'	STRING#''
V	Value	LREAL	Desired value to be recorded.	LREAL#0.0
V	Precision	INT	Number of characters after the decimal point to be written to the log file. (Ex. Precision = 4, value = 15.346781, value written to log file = 15.3467). Default will write full LREAL value to log file.	INT#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	FileWriteDone	BOOL	Indicates when log has completed writing – depending on scan times and length of recording this might take longer to complete	
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	RecordingStoppedEarly	BOOL	Internal buffer ran out of space to record more data. Will complete writing all valid data in buffers and close log 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.

Notes

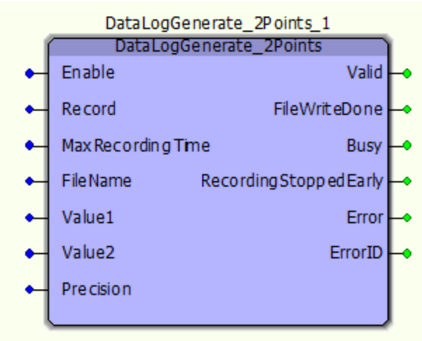
- Don't forget to include the ProConOS firmware library in the project. It is required for this function block.
- This function block requires that the variable PLC_SYS_TICK_CNT be defined as a global variable corresponding to a specified hardware address. This variable is automatically created and added to the System Variables when a new template project is opened.

Error Description

ErrorID	Meaning
0	No error.
4	File is already open.
5	File is opened, write protected or access denied.
6	File name not defined.
10	End of data reached.
12	The number of characters to be read is greater than the data buffer.
13	Invalid positioning mode or position specified is before the beginning of the file.
20	File could not be closed.
22	No data could be read.
24	Position could not be set.

Example

DataLogGenerate_2Points



This function block will record a log file containing [TimeStamp, Value1 input, Value2 input] for a given duration. This CSV file can then be exported from the controller for post processing of the data logged.

Library

File RW 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	Record	BOOL	Will begin recording a new log when this input is set high. (this operation will delete previous log of same name if it exists on the controller). Logging stops when record input becomes false or MaxRecordingTime is reached.	FALSE
V	MaxRecordingTime	LREAL	Maximum length of time data will be logged for. If left unconnected, data will be logged as log as Record is TRUE.	LREAL#0.0
V	FileName	STRING	Name of file to be read or written. Example STRING#'/-flash/user/data/myFile.csv'	STRING#''
V	Value1	LREAL	Desired value to be recorded.	LREAL#0.0
V	Value2	LREAL	Desired value to be recorded.	LREAL#0.0
V	Precision	INT	Number of characters after the decimal point to be written to the log file. (Ex. Precision = 4, value = 15.346781, value written to log file = 15.3467). Default will write full LREAL value to log file.	INT#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the function is operating normally and the outputs of the function are valid.	
V	FileWriteDone	BOOL	Indicates when log has completed writing – depending on scan times and length of recording this might take longer to complete	
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	RecordingStoppedEarly	BOOL	Internal buffer ran out of space to record more data. Will complete writing all valid data in buffers and close log 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.

Notes

- Don't forget to include the ProConOS firmware library in the project. It is required for this function block.

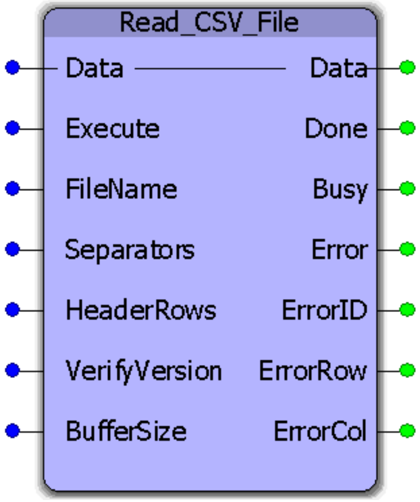
- This function block requires that the variable PLC_SYS_TICK_CNT be defined as a global variable corresponding to a specified hardware address. This variable is automatically created and added to the System Variables when a new template project is opened.

Error Description

ErrorID	Meaning
0	No error.
4	File is already open.
5	File is opened, write protected or access denied.
6	File name not defined.
10	End of data reached.
12	The number of characters to be read is greater than the data buffer.
13	Invalid positioning mode or position specified is before the beginning of the file.
20	File could not be closed.
22	No data could be read.
24	Position could not be set.

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 your 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**
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.	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. 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#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.	
B	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.	
E	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

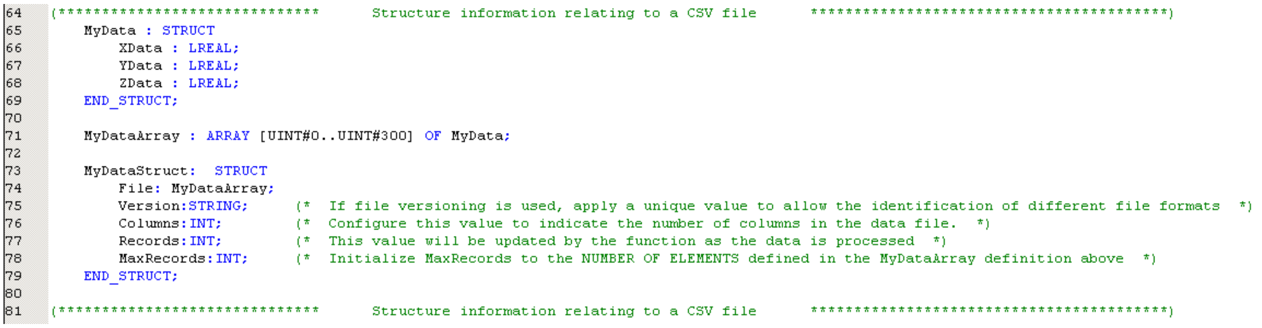
ErrorID	Meaning
0	No error.
4	File is already open.
5	File is opened, write protected or access denied.
6	File name not defined.
10	End of data reached.
12	The number of characters to be read is greater than the data buffer.
13	Invalid positioning mode or position specified is before the beginning of the file.
20	File could not be closed.
22	No data could be read.
24	Position could not be set.
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.
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.

Example Customization

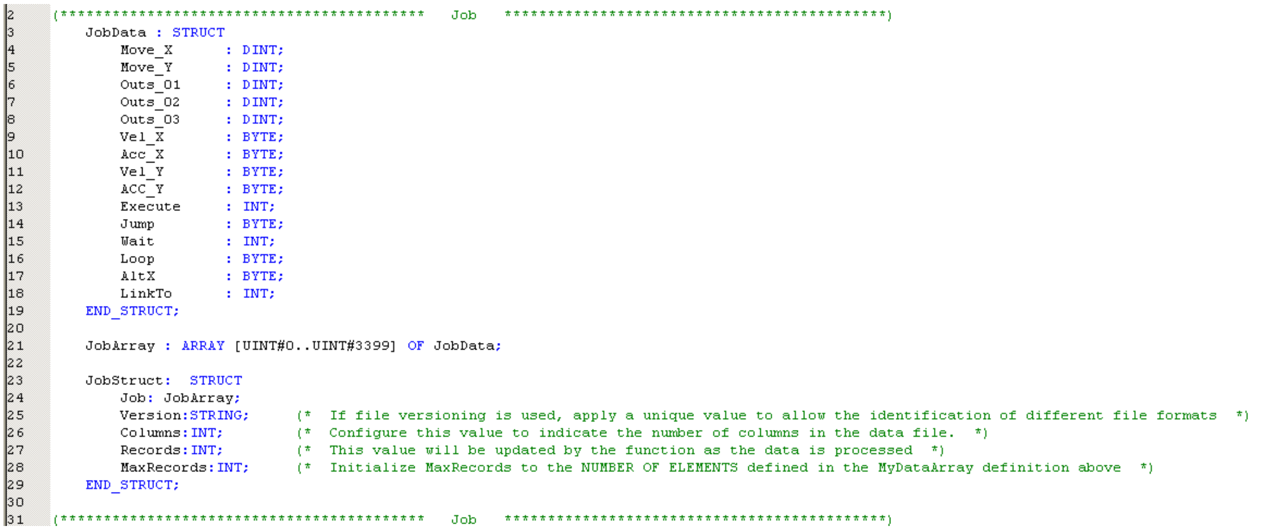
Read_CSV_File must be customized to accommodate your data. Some supporting functions used by Read_CSV_File (ReadBuffer and ReadValue) do not require customization and can remain in the File_RW_Toolbox. To effectively use this function, follow these steps:

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



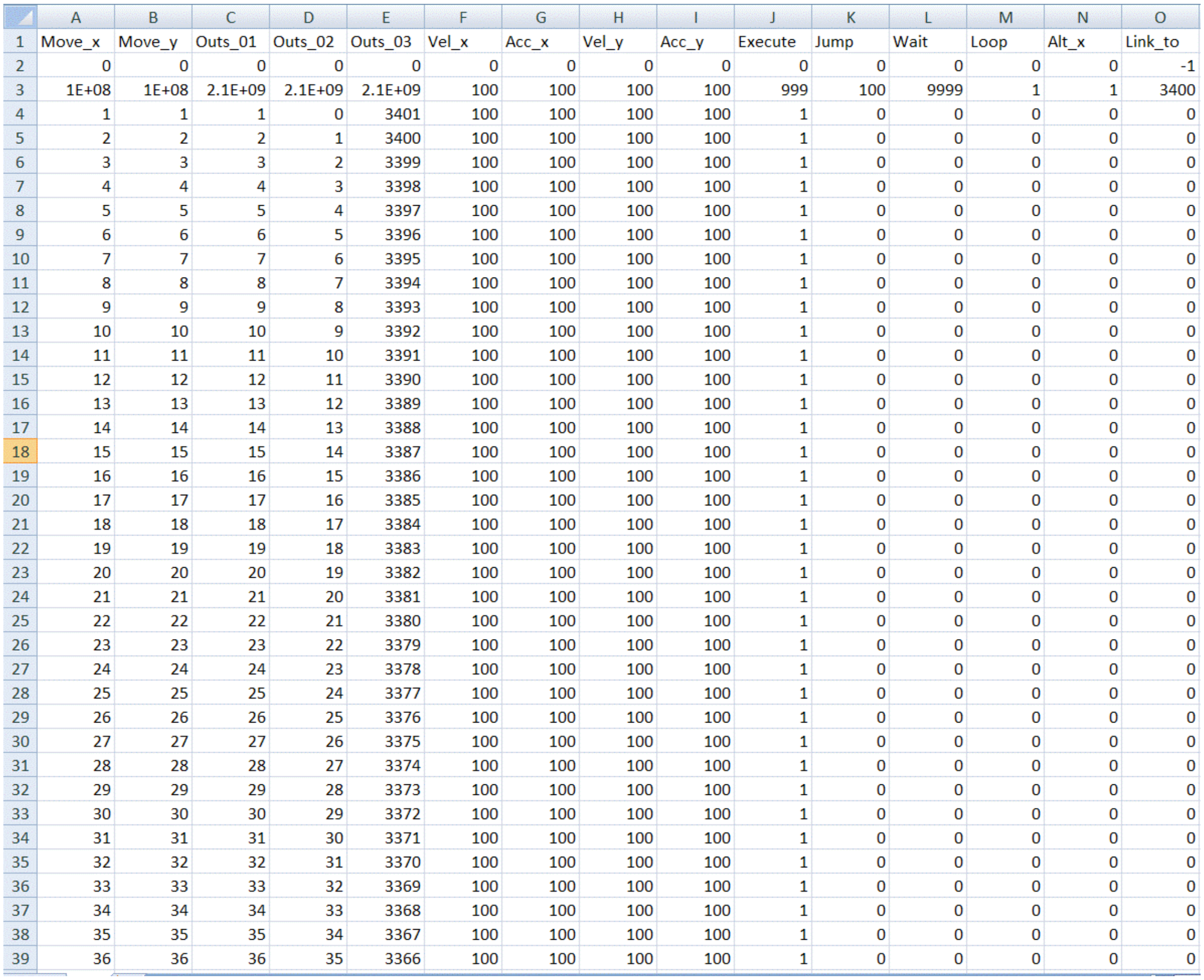
```
(******************************      Structure information relating to a CSV file      *****************************************)
    MyData : STRUCT
        XData : LREAL;
        YData : LREAL;
        ZData : LREAL;
    END_STRUCT;

    MyDataArray : ARRAY [UINT#0..UINT#300] OF MyData;

    MyDataStruct:  STRUCT
        File: MyDataArray;
        Version:STRING;     (*  If file versioning is used, apply a unique value to allow the identification of different file formats  *)
        Columns:INT;        (*  Configure this value to indicate the number of columns in the data file.  *)
        Records:INT;        (*  This value will be updated by the function as the data is processed  *)
        MaxRecords:INT;     (*  Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above  *)
    END_STRUCT;

(******************************      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 datatypes. An example follows:



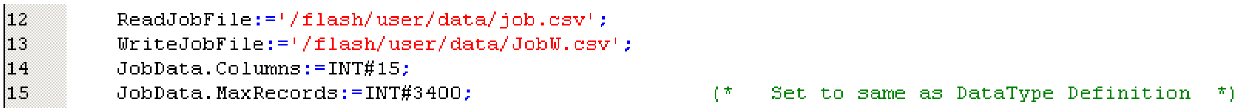
```
(******************************************    Job    ********************************************)
    JobData : STRUCT
        Move_X       : DINT;
        Move_Y       : DINT;
        Outs_01      : DINT;
        Outs_02      : DINT;
        Outs_03      : DINT;
        Vel_X        : BYTE;
        Acc_X        : BYTE;
        Vel_Y        : BYTE;
        ACC_Y        : BYTE;
        Execute      : INT;
        Jump         : BYTE;
        Wait         : INT;
        Loop         : BYTE;
        AltX         : BYTE;
        LinkTo       : INT;
    END_STRUCT;

    JobArray : ARRAY [UINT#0..UINT#3399] OF JobData;

    JobStruct:  STRUCT
        Job: JobArray;
        Version:STRING;     (*  If file versioning is used, apply a unique value to allow the identification of different file formats  *)
        Columns:INT;        (*  Configure this value to indicate the number of columns in the data file.  *)
        Records:INT;        (*  This value will be updated by the function as the data is processed  *)
        MaxRecords:INT;     (*  Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above  *)
    END_STRUCT;

(******************************************    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.



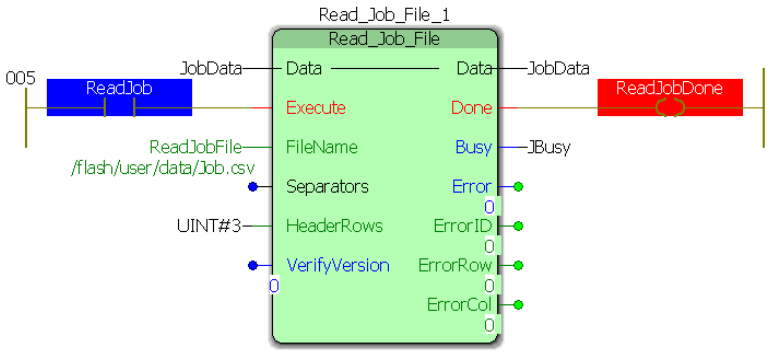
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Move_x	Move_y	Outs_01	Outs_02	Outs_03	Vel_x	Acc_x	Vel_y	Acc_y	Execute	Jump	Wait	Loop	Alt_x	Link_to
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
3	1E+08	1E+08	2.1E+09	2.1E+09	2.1E+09	100	100	100	100	999	100	9999	1	1	3400
4	1	1	1	0	3401	100	100	100	100	1	0	0	0	0	0
5	2	2	2	1	3400	100	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

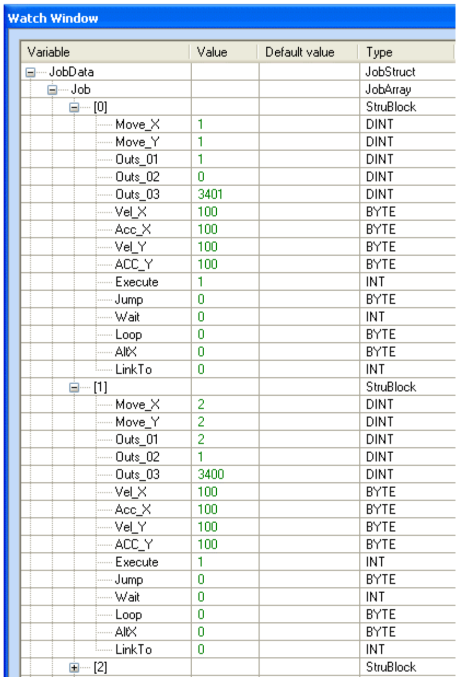
3) Initialize the data required for "MyDataStruct" as shown below. Most importantly, set Columns and MaxRecords.



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

4) Copy & paste the Read_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 Read_CSV_File in the Toolbox.



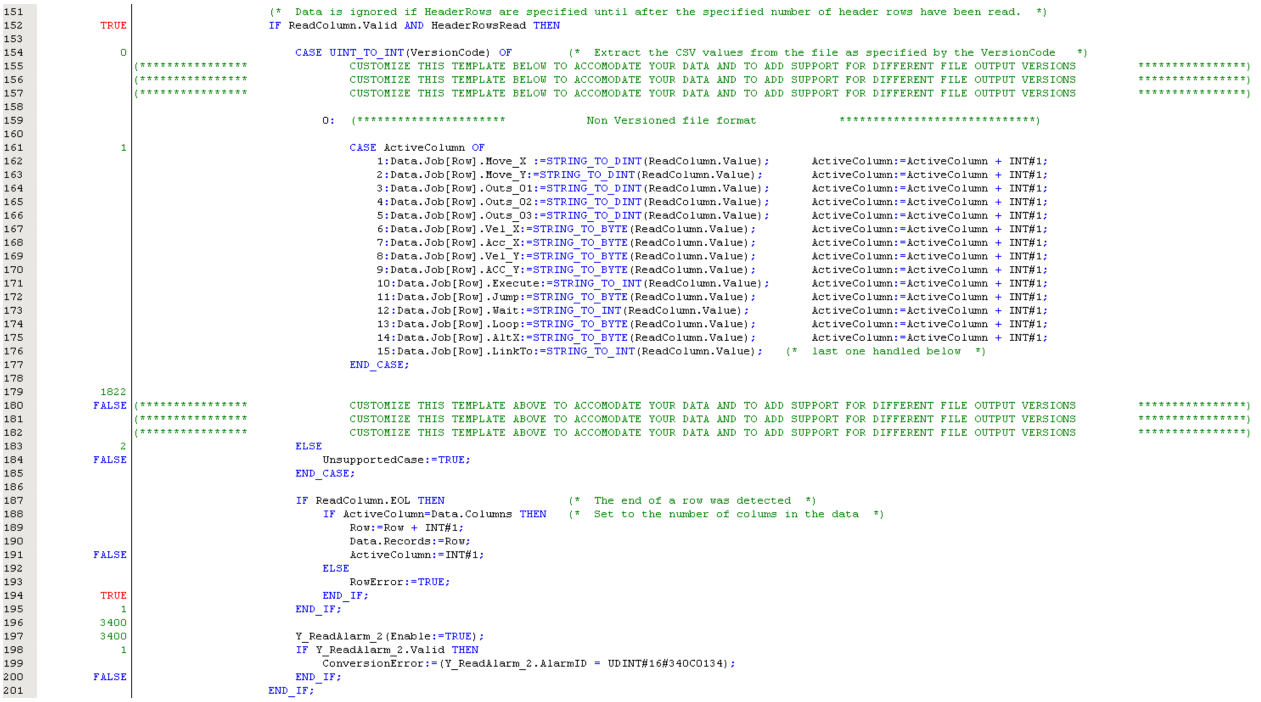

Watch Window

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
AltX	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
AltX	0		BYTE
LinkTo	0		INT
⊞ [2]			StruBlock

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 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.

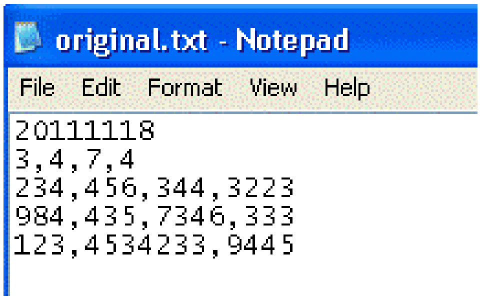


```
      |                    (*  Data is ignored if HeaderRows are specified until after the specified number of header rows have been read.  *)
 TRUE |                    IF ReadColumn.Valid AND HeaderRowsRead THEN
      |
    0 |                        CASE UINT_TO_INT(VersionCode) OF        (*  Extract the CSV values from the file as specified by the VersionCode   *)
      |(****************               CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
      |(****************               CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
      |(****************               CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
      |
      |                            0:  (**********************           Non Versioned file format           *****************************)
      |
    1 |                                CASE ActiveColumn OF
      |                                    1:Data.Job[Row].Move_X :=STRING_TO_DINT(ReadColumn.Value);      ActiveColumn:=ActiveColumn + INT#1;
      |                                    2:Data.Job[Row].Move_Y:=STRING_TO_DINT(ReadColumn.Value);       ActiveColumn:=ActiveColumn + INT#1;
      |                                    3:Data.Job[Row].Outs_01:=STRING_TO_DINT(ReadColumn.Value);      ActiveColumn:=ActiveColumn + INT#1;
      |                                    4:Data.Job[Row].Outs_02:=STRING_TO_DINT(ReadColumn.Value);      ActiveColumn:=ActiveColumn + INT#1;
      |                                    5:Data.Job[Row].Outs_03:=STRING_TO_DINT(ReadColumn.Value);      ActiveColumn:=ActiveColumn + INT#1;
      |                                    6:Data.Job[Row].Vel_X:=STRING_TO_BYTE(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
      |                                    7:Data.Job[Row].Acc_X:=STRING_TO_BYTE(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
      |                                    8:Data.Job[Row].Vel_Y:=STRING_TO_BYTE(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
      |                                    9:Data.Job[Row].ACC_Y:=STRING_TO_BYTE(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
      |                                    10:Data.Job[Row].Execute:=STRING_TO_INT(ReadColumn.Value);      ActiveColumn:=ActiveColumn + INT#1;
      |                                    11:Data.Job[Row].Jump:=STRING_TO_BYTE(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
      |                                    12:Data.Job[Row].Wait:=STRING_TO_INT(ReadColumn.Value);         ActiveColumn:=ActiveColumn + INT#1;
      |                                    13:Data.Job[Row].Loop:=STRING_TO_BYTE(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
      |                                    14:Data.Job[Row].AltX:=STRING_TO_BYTE(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
      |                                    15:Data.Job[Row].LinkTo:=STRING_TO_INT(ReadColumn.Value);   (*  last one handled below  *)
      |                                END_CASE;
      |
 1822 |
FALSE |(****************               CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
      |(****************               CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
      |(****************               CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
    2 |                        ELSE
FALSE |                            UnsupportedCase:=TRUE;
      |                        END_CASE;
      |
      |                        IF ReadColumn.EOL THEN                  (*  The end of a row was detected  *)
      |                            IF ActiveColumn=Data.Columns THEN   (*  Set to the number of colums in the data  *)
      |                                Row:=Row + INT#1;
      |                                Data.Records:=Row;
FALSE |                                ActiveColumn:=INT#1;
      |                            ELSE
      |                                RowError:=TRUE;
 TRUE |                            END_IF;
    1 |                        END_IF;
 3400 |
 3400 |                        Y_ReadAlarm_2(Enable:=TRUE);
    1 |                        IF Y_ReadAlarm_2.Valid THEN
      |                            ConversionError:=(Y_ReadAlarm_2.AlarmID = UDINT#16#340C0134);
FALSE |                        END_IF;
      |                    END_IF;
```

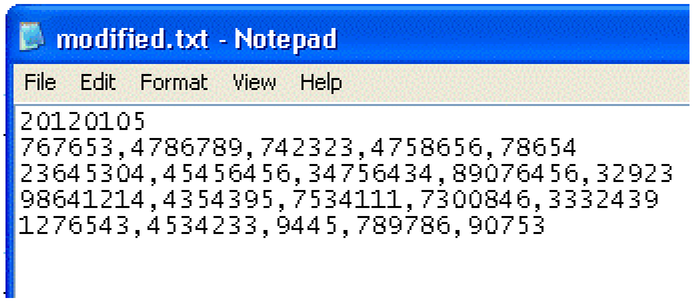
Customizing for file versioning

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 Modified file specification



```
20111118
3,4,7,4
234,456,344,3223
984,435,7346,333
123,4534233,9445
```

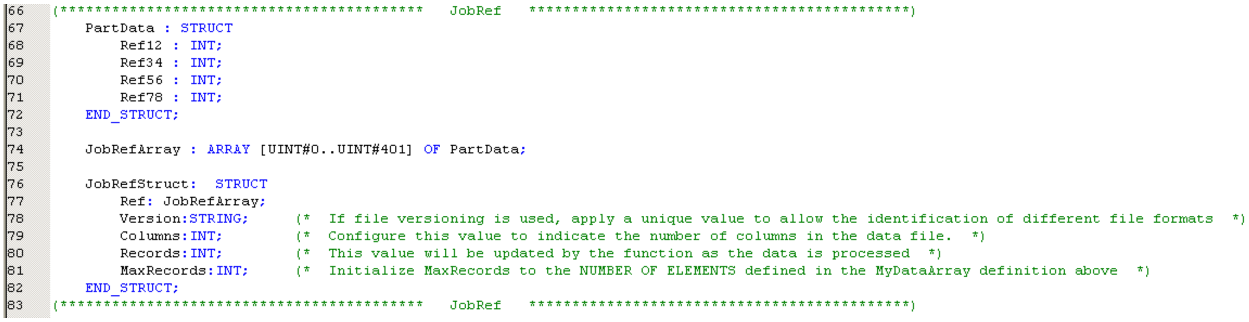


```
20120105
767653,4786789,742323,4758656,78654
23645304,45456456,34756434,89076456,32923
98641214,4354395,7534111,7300846,3332439
1276543,4534233,9445,789786,90753
```

To use file versioning, follow the steps below:

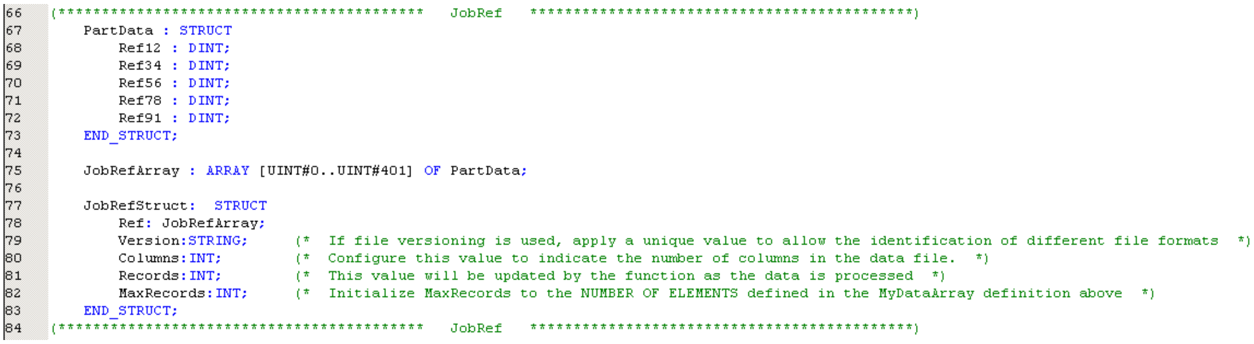
1. Set the VerifyVersion function block input to TRUE.
2. The first line of the data file must contain a version code. The version code does NOT count as a header row. See the graphics above showing original and modified file specification
3. Customize the DataType to reflect the most current data specification.

Original DataType:



```
(*******************************************   JobRef   *********************************************)
    PartData : STRUCT
        Ref12 : INT;
        Ref34 : INT;
        Ref56 : INT;
        Ref78 : INT;
    END_STRUCT;

    JobRefArray : ARRAY [UINT#0..UINT#401] OF PartData;

    JobRefStruct:  STRUCT
        Ref: JobRefArray;
        Version:STRING;      (*  If file versioning is used, apply a unique value to allow the identification of different file formats  *)
        Columns:INT;         (*  Configure this value to indicate the number of columns in the data file.  *)
        Records:INT;         (*  This value will be updated by the function as the data is processed  *)
        MaxRecords:INT;      (*  Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above  *)
    END_STRUCT;
(*******************************************   JobRef   *********************************************)
```

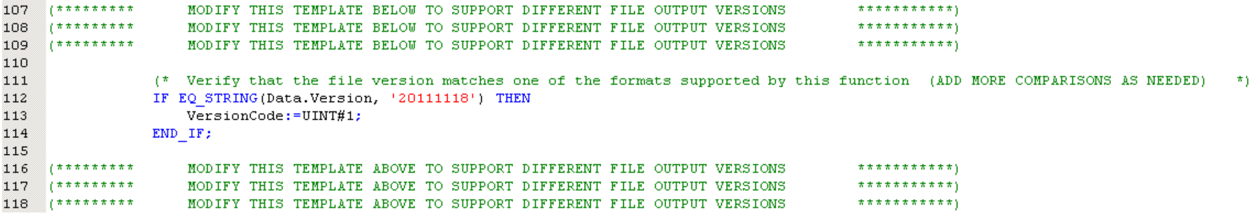
Modified DataType:



```
(*******************************************   JobRef   *********************************************)
    PartData : STRUCT
        Ref12 : DINT;
        Ref34 : DINT;
        Ref56 : DINT;
        Ref78 : DINT;
        Ref91 : DINT;
    END_STRUCT;

    JobRefArray : ARRAY [UINT#0..UINT#401] OF PartData;

    JobRefStruct:  STRUCT
        Ref: JobRefArray;
        Version:STRING;      (*  If file versioning is used, apply a unique value to allow the identification of different file formats  *)
        Columns:INT;         (*  Configure this value to indicate the number of columns in the data file.  *)
        Records:INT;         (*  This value will be updated by the function as the data is processed  *)
        MaxRecords:INT;      (*  Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above  *)
    END_STRUCT;
(*******************************************   JobRef   *********************************************)
```

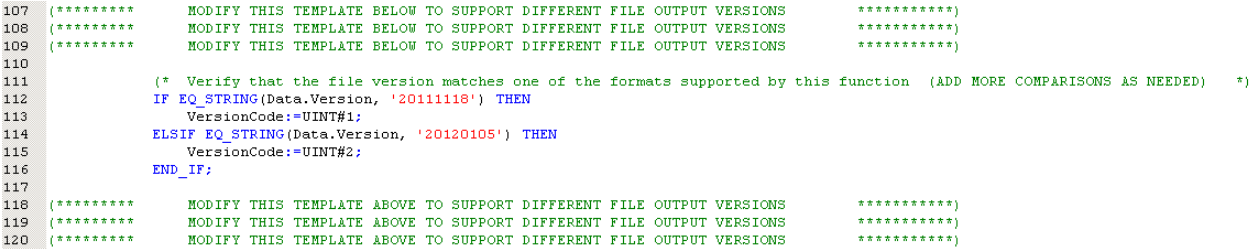
3) Customize the Read-CSV_File function block to determine if the version code detected is supported.

Original code:



```
(*********      MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
(*********      MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
(*********      MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)

            (*  Verify that the file version matches one of the formats supported by this function  (ADD MORE COMPARISONS AS NEEDED)   *)
            IF EQ_STRING(Data.Version, '20111118') THEN
                VersionCode:=UINT#1;
            END_IF;

(*********      MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
(*********      MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
(*********      MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
```

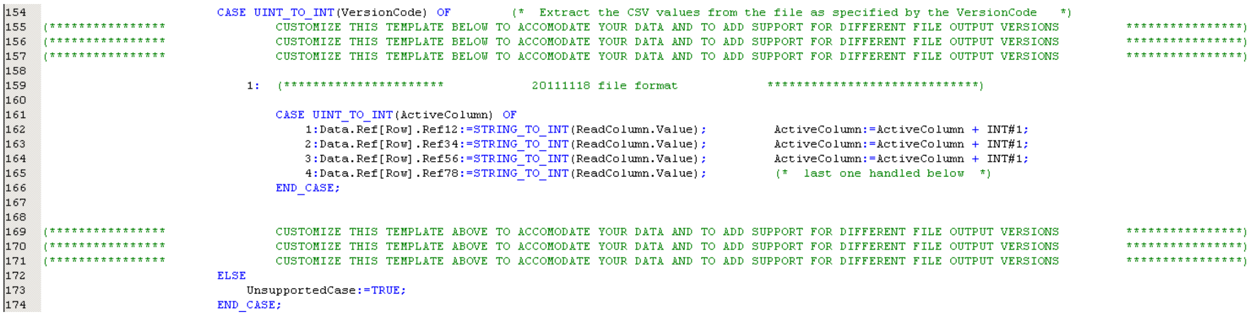
Modified code:



```
(*********      MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
(*********      MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
(*********      MODIFY THIS TEMPLATE BELOW TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)

            (*  Verify that the file version matches one of the formats supported by this function  (ADD MORE COMPARISONS AS NEEDED)   *)
            IF EQ_STRING(Data.Version, '20111118') THEN
                VersionCode:=UINT#1;
            ELSIF EQ_STRING(Data.Version, '20120105') THEN
                VersionCode:=UINT#2;
            END_IF;

(*********      MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
(*********      MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
(*********      MODIFY THIS TEMPLATE ABOVE TO SUPPORT DIFFERENT FILE OUTPUT VERSIONS        ***********)
```

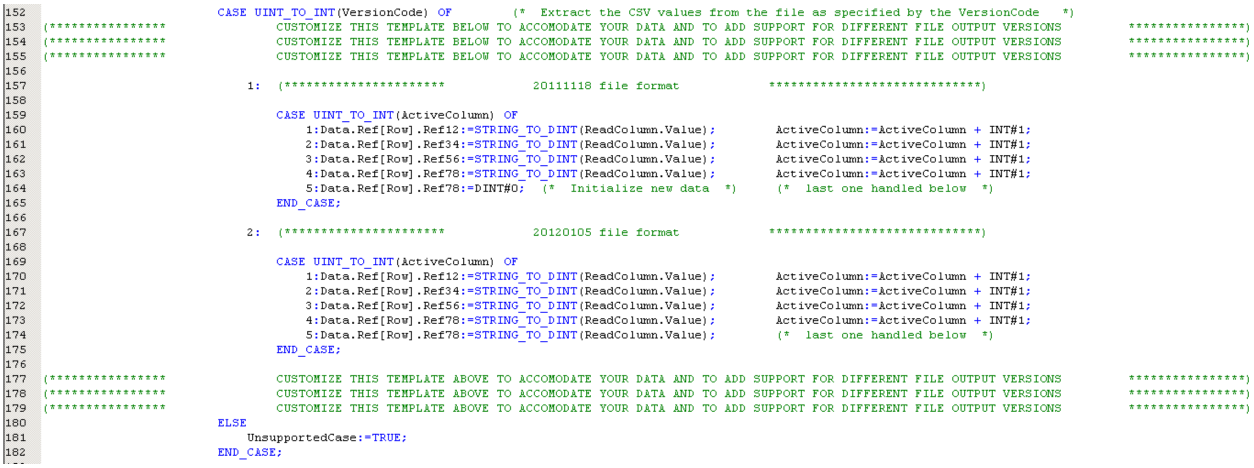
4) Customize the Read_CSV_File function block to read multiple versions.

Original code:



```
                        CASE UINT_TO_INT(VersionCode) OF        (*  Extract the CSV values from the file as specified by the VersionCode   *)
(****************               CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
(****************               CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
(****************               CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)

                            1:  (*********************           20111118 file format            ****************************)

                                CASE UINT_TO_INT(ActiveColumn) OF
                                    1:Data.Ref[Row].Ref12:=STRING_TO_INT(ReadColumn.Value);         ActiveColumn:=ActiveColumn + INT#1;
                                    2:Data.Ref[Row].Ref34:=STRING_TO_INT(ReadColumn.Value);         ActiveColumn:=ActiveColumn + INT#1;
                                    3:Data.Ref[Row].Ref56:=STRING_TO_INT(ReadColumn.Value);         ActiveColumn:=ActiveColumn + INT#1;
                                    4:Data.Ref[Row].Ref78:=STRING_TO_INT(ReadColumn.Value);         (*  last one handled below  *)
                                END_CASE;


(****************               CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
(****************               CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
(****************               CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
                        ELSE
                            UnsupportedCase:=TRUE;
                        END_CASE;
```

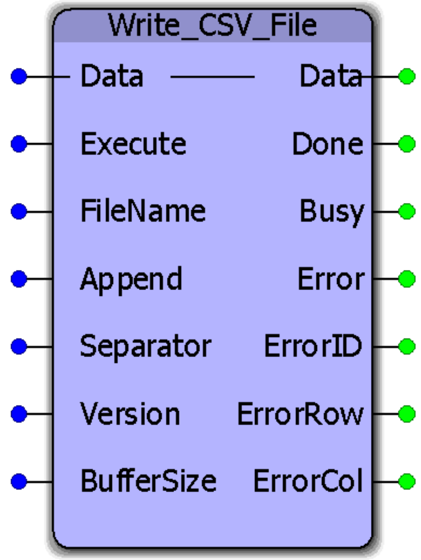
Modified code:



```
                        CASE UINT_TO_INT(VersionCode) OF        (*  Extract the CSV values from the file as specified by the VersionCode   *)
(****************               CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
(****************               CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
(****************               CUSTOMIZE THIS TEMPLATE BELOW TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)

                            1:  (**********************            20111118 file format            ****************************)

                                CASE UINT_TO_INT(ActiveColumn) OF
                                    1:Data.Ref[Row].Ref12:=STRING_TO_DINT(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
                                    2:Data.Ref[Row].Ref34:=STRING_TO_DINT(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
                                    3:Data.Ref[Row].Ref56:=STRING_TO_DINT(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
                                    4:Data.Ref[Row].Ref78:=STRING_TO_DINT(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
                                    5:Data.Ref[Row].Ref78:=DINT#0;  (*  Initialize new data  *)     (*  last one handled below  *)
                                END_CASE;

                            2:  (**********************            20120105 file format            ****************************)

                                CASE UINT_TO_INT(ActiveColumn) OF
                                    1:Data.Ref[Row].Ref12:=STRING_TO_DINT(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
                                    2:Data.Ref[Row].Ref34:=STRING_TO_DINT(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
                                    3:Data.Ref[Row].Ref56:=STRING_TO_DINT(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
                                    4:Data.Ref[Row].Ref78:=STRING_TO_DINT(ReadColumn.Value);        ActiveColumn:=ActiveColumn + INT#1;
                                    5:Data.Ref[Row].Ref78:=STRING_TO_DINT(ReadColumn.Value);        (*  last one handled below  *)
                                END_CASE;

(****************               CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
(****************               CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
(****************               CUSTOMIZE THIS TEMPLATE ABOVE TO ACCOMODATE YOUR DATA AND TO ADD SUPPORT FOR DIFFERENT FILE OUTPUT VERSIONS         ****************)
                        ELSE
                            UnsupportedCase:=TRUE;
                        END_CASE;
```

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.

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. Write_CSV_File must be customized to accommodate your data. 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#'ramdisk/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 Appended=TRUE, data will be appended.	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	UDINT#0

			be customized to select between multiple output formats.	
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

- 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, and not the 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.

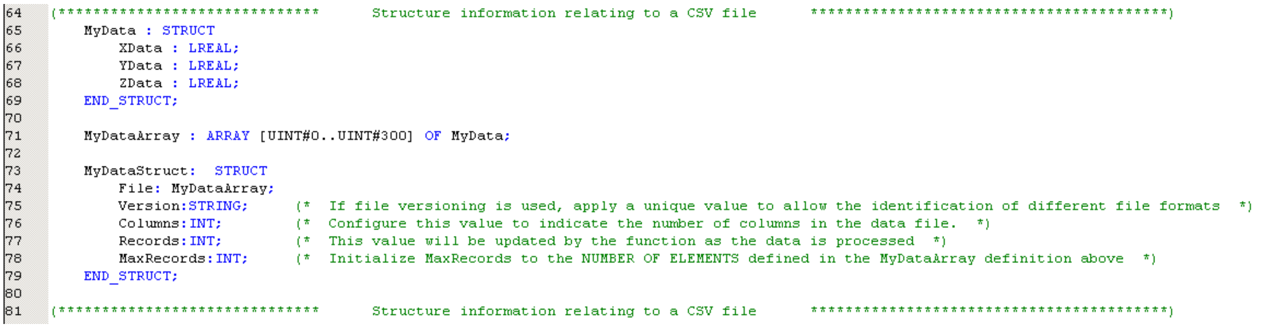
Error Description

ErrorID	Meaning
0	No error.
4	File is already open.
5	File is opened, write protected or access denied.
6	File name not defined.
10	End of data reached.
12	The number of characters to be read is greater than the data buffer.
13	Invalid positioning mode or position specified is before the beginning of the file.
20	File could not be closed.
22	No data could be read.
24	Position could not be set.
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.
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.
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.

Customization Example 1

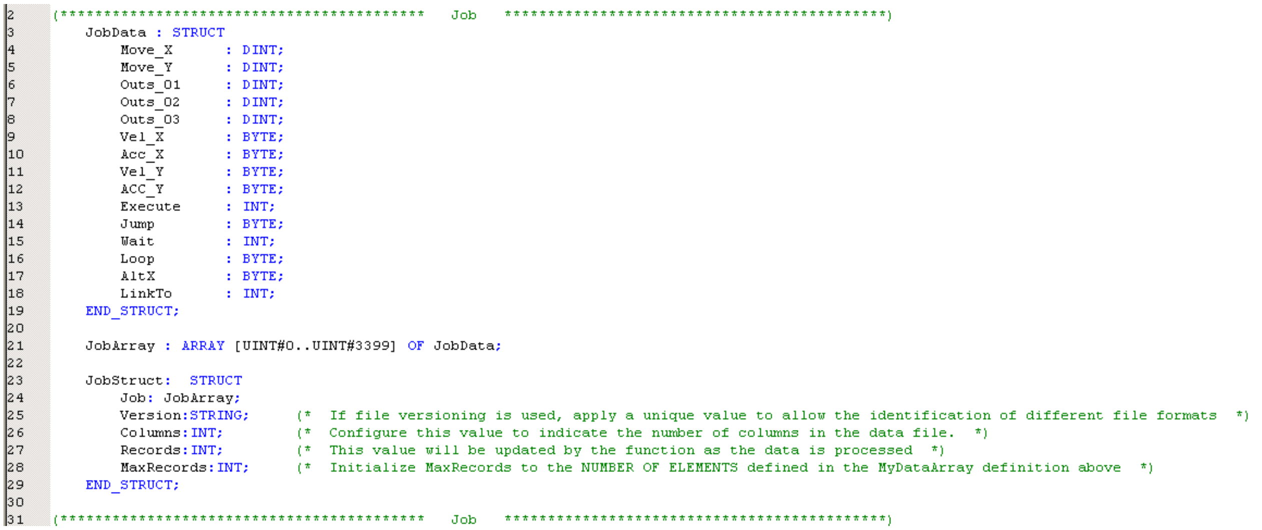
Write_CSV_File must be customized to accommodate your data. Some supporting functions used by Write_CSV_File (ReadBuffer and ReadValue) do not require customization and can remain in the File_RW_Toolbox. Two locations requiring customization are identified in the function block by several rows of comments 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.



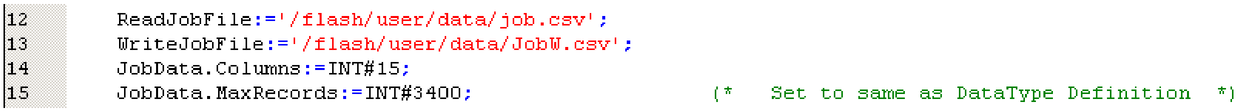
```
(*******************************      Structure information relating to a CSV file      ******************************************)
    MyData : STRUCT
        XData : LREAL;
        YData : LREAL;
        ZData : LREAL;
    END_STRUCT;

    MyDataArray : ARRAY [UINT#0..UINT#300] OF MyData;

    MyDataStruct:  STRUCT
        File: MyDataArray;
        Version:STRING;     (*  If file versioning is used, apply a unique value to allow the identification of different file formats  *)
        Columns:INT;        (*  Configure this value to indicate the number of columns in the data file.  *)
        Records:INT;        (*  This value will be updated by the function as the data is processed  *)
        MaxRecords:INT;     (*  Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above  *)
    END_STRUCT;

(*******************************      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:



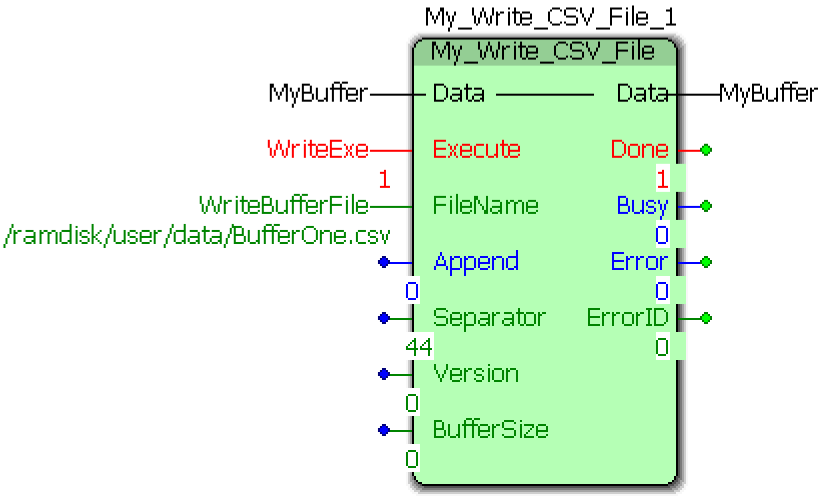
```
(*******************************************   Job   ********************************************)
    JobData : STRUCT
        Move_X      : DINT;
        Move_Y      : DINT;
        Outs_01     : DINT;
        Outs_02     : DINT;
        Outs_03     : DINT;
        Vel_X       : BYTE;
        Acc_X       : BYTE;
        Vel_Y       : BYTE;
        ACC_Y       : BYTE;
        Execute     : INT;
        Jump        : BYTE;
        Wait        : INT;
        Loop        : BYTE;
        AltX        : BYTE;
        LinkTo      : INT;
    END_STRUCT;

    JobArray : ARRAY [UINT#0..UINT#3399] OF JobData;

    JobStruct:  STRUCT
        Job: JobArray;
        Version:STRING;     (*  If file versioning is used, apply a unique value to allow the identification of different file formats  *)
        Columns:INT;        (*  Configure this value to indicate the number of columns in the data file.  *)
        Records:INT;        (*  This value will be updated by the function as the data is processed  *)
        MaxRecords:INT;     (*  Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above  *)
    END_STRUCT;

(*******************************************   Job   ********************************************)
```

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



```
    ReadJobFile:='/flash/user/data/job.csv';
    WriteJobFile:='/flash/user/data/JobW.csv';
    JobData.Columns:=INT#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

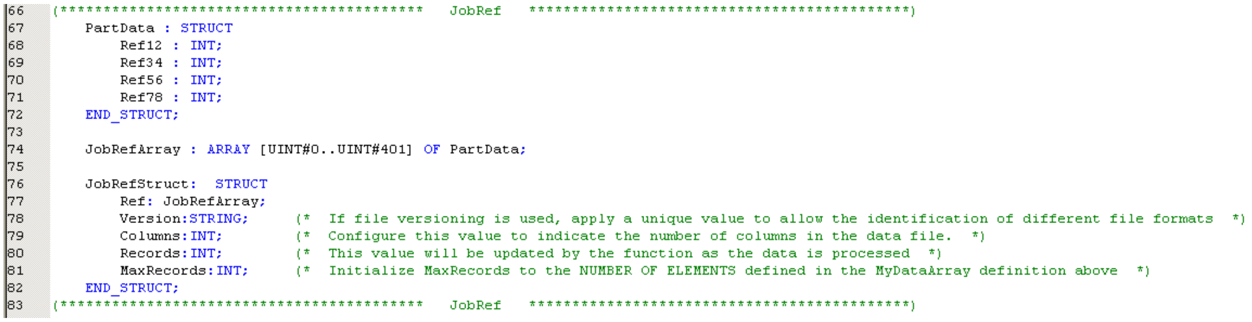
The function has the capability to write multiple versions of the same structure. For example, a portion of the data from the structure can be written to one file, and a different set of data can be written to another file.

To use file versioning, follow the steps below:

1) Set the 'Version' function block input to a unique value (Non zero).

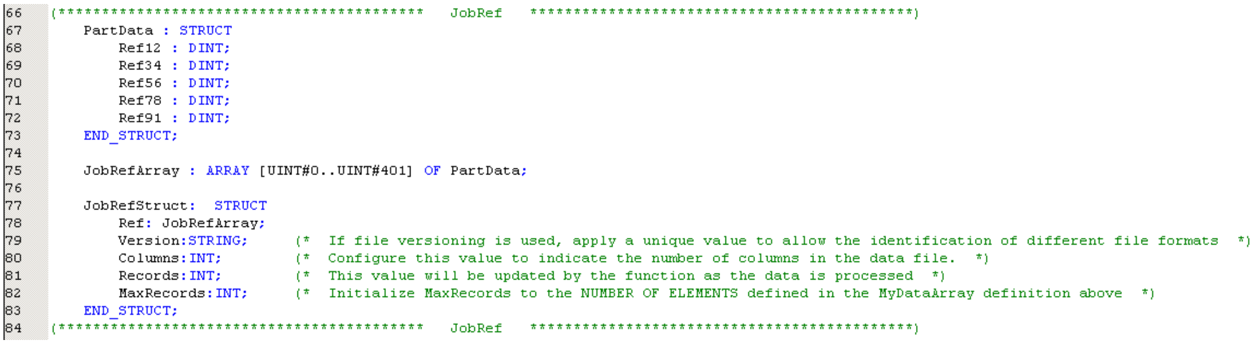
2) Customize the DataType to reflect the most current data specification.

Original DataType:



```
(******************************************   JobRef   ******************************************)
    PartData : STRUCT
        Ref12 : INT;
        Ref34 : INT;
        Ref56 : INT;
        Ref78 : INT;
    END_STRUCT;

    JobRefArray : ARRAY [UINT#0..UINT#401] OF PartData;

    JobRefStruct:  STRUCT
        Ref: JobRefArray;
        Version:STRING;     (*  If file versioning is used, apply a unique value to allow the identification of different file formats  *)
        Columns:INT;        (*  Configure this value to indicate the number of columns in the data file.  *)
        Records:INT;        (*  This value will be updated by the function as the data is processed  *)
        MaxRecords:INT;     (*  Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above  *)
    END_STRUCT;
(******************************************   JobRef   ******************************************)
```

Modified DataType:



```
(******************************************   JobRef   ******************************************)
    PartData : STRUCT
        Ref12 : DINT;
        Ref34 : DINT;
        Ref56 : DINT;
        Ref78 : DINT;
        Ref91 : DINT;
    END_STRUCT;

    JobRefArray : ARRAY [UINT#0..UINT#401] OF PartData;

    JobRefStruct:  STRUCT
        Ref: JobRefArray;
        Version:STRING;     (*  If file versioning is used, apply a unique value to allow the identification of different file formats  *)
        Columns:INT;        (*  Configure this value to indicate the number of columns in the data file.  *)
        Records:INT;        (*  This value will be updated by the function as the data is processed  *)
        MaxRecords:INT;     (*  Initialize MaxRecords to the NUMBER OF ELEMENTS defined in the MyDataArray definition above  *)
    END_STRUCT;
(******************************************   JobRef   ******************************************)
```

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

Original code:

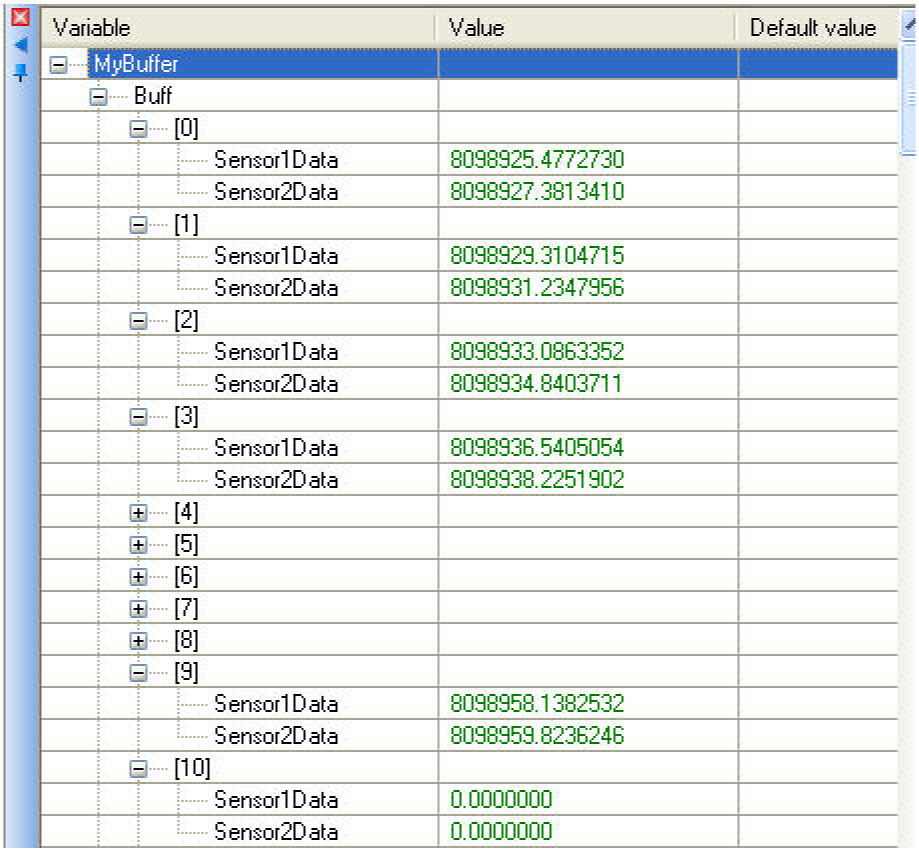
Modified code:

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

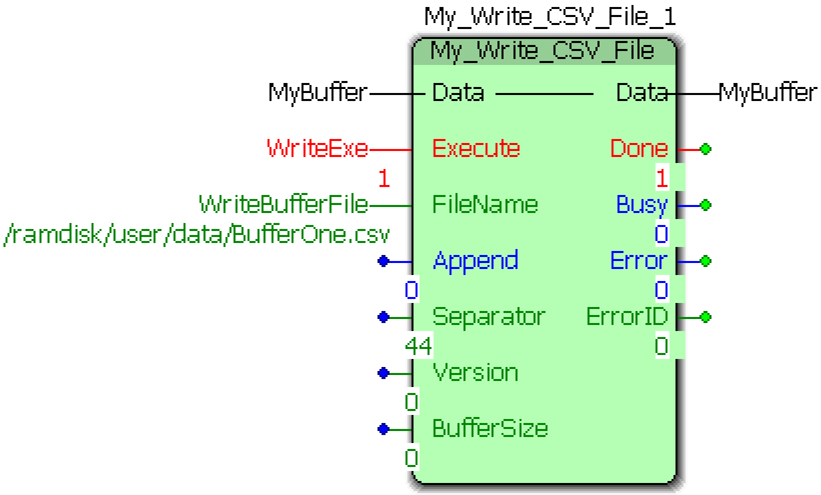
Original code:

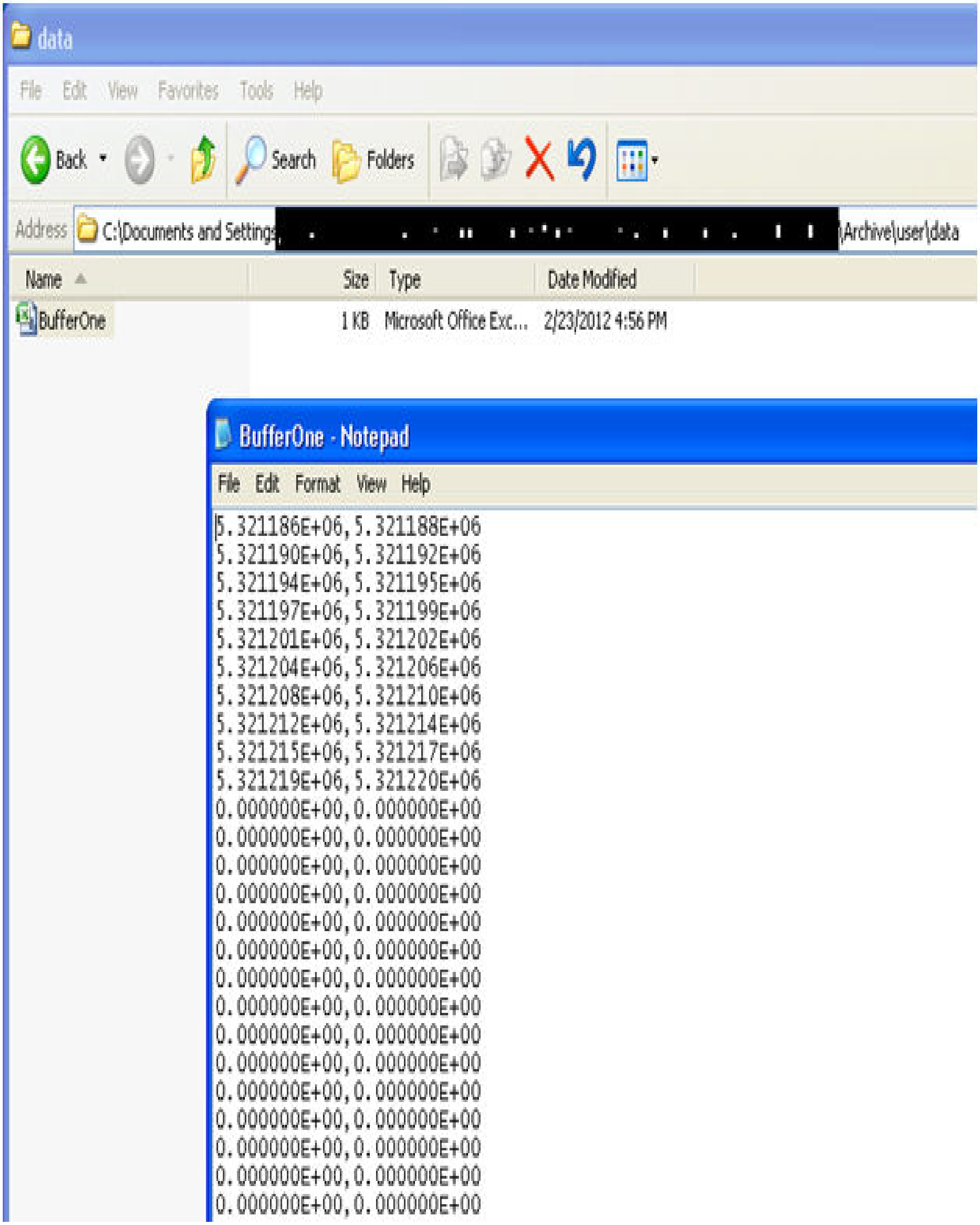
Modified code:

Application Example



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	




data
File Edit View Favorites Tools Help
Back
Search
Folders
Address

C:\Documents and Settings[redacted]\Archive\user\data

Name	Size	Type	Date Modified
BufferOne	1 KB	Microsoft Office Exc...	2/23/2012 4:56 PM

BufferOne - Notepad

File Edit Format View Help

```
5.321186E+06,5.321188E+06
5.321190E+06,5.321192E+06
5.321194E+06,5.321195E+06
5.321197E+06,5.321199E+06
5.321201E+06,5.321202E+06
5.321204E+06,5.321206E+06
5.321208E+06,5.321210E+06
5.321212E+06,5.321214E+06
5.321215E+06,5.321217E+06
5.321219E+06,5.321220E+06
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
0.000000E+00,0.000000E+00
```

Gantry Toolbox

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Getting Started with Gantry Toolbox

Requirements for v301

To use the Gantry Toolbox, your project must also contain the following:

Firmware libraries:

- YMotion

User libraries:

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

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

Using the Gantry Toolbox

See Yaskawa's Youtube Webinar - XY Interpolation via the Gantry Toolbox for more info.

Gantry Toolbox Revision History

New in v203 – 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 Gantry Toolbox, you must do the following:
 1) Include the DataTypes Toolbox in your project.
 2) Remove any other Yaskawa supplied datatype files with firmware library definitions such as
   a. ControllInfoTypes
   b. YDeviceCommTypes

Current Version:

(******************************** 2016-11-16 v301 released ***********************************)

1) PathGenerator - DCR 862, fixed circle segments when gantry has Z axis. No master cam position was being set for Z.

Previous Versions:

(******************************** 2015-01-31 v300 released ***********************************)

1) Identical to v204, but recompiled specifically for MotionWorks IEC v3.x.

(************************************* 2015-01-07 v204 released
***)

1) Added Z axis to Gantry Struct for G-Code experimentation.

2) Move_Path - improved ability to change speed on the fly, including zero velocity (Pause.)

3) Gantry_Power - Added missing line to clear ErrorID when Execute goes low.

4) Math Toolbox - Removed references to the toolbox where possible. Calculate_Angles still requires ATAN2 function.

5) Gantry_Stop - Fixed typos pertaining to setting the ErrorID for many internal errors.

(***********************************   2013-03-15 v203 released   *********************************)

Created from Gantry_Toolbox_v203_d_KH

1) GantryDataTypes file - Added Tangent Axis to Gantry Struct. This axis will be tangential to X, Y axes.

2) GantryDataTypes file - Added InputConditions and StandStillDuration to Path details structure. These will be used for pause sections in the path.

3) GantryDataTypes file - Made PathPointArray size 1000.

4) GantryDataTypes file - Added StandStill and WaitForInputs enum types to TB_PatternType.

5) GantryDataTypes file - Added TangentAxisTable to PathIDStruct.

6) GantryDataTypes file - Added InputConditions and StandStillDuration to SegmentDetails.

7) GantryDataTypes file - Made SegmentArray size 1000.

8) GantryDataTypes file - Created SegmentMapArray to map between managed segments and user defined segments

9) GantryDataTypes file - Added ManagedSegment, LastManagedSegment, AbortPath and SegmentMap to Segmentstruct

10) GantryDataTypes file - Added TangentActive to PathDetails. Used to decide if a segment requires a tangent axis to be oriented correctly at the beginning and/or end.

11) GantryDataTypes file - PathPointArray increased to 2047.

12) Gantry_Power - Removed Alarm and Warning outputs.

13) Gantry_Power - Added support for a Tangent axis.

14) Gantry_Power - Added status word output. This word shows which axes are powered on.

15) Gantry_Stop - Added support to stop all configured Gantry Axes

16) PathGenerator - Added support for a tangent axis

17) PathGenerator - Added support for intermittent motion and pauses

18) Move_Path - Added ability to move and pause virtual master based on the segment details

19) Move_Path - Added InputCondtions as a FB input for user inputs to restart motion at WaitForInputs segment

20) PathIDManager - Function block added. Removes paths from memory that are no longer needed.

Gantry DataTypes

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Data Type: GantryPositions

This datatype can be used to store absolute positions within the coordinate system. It is not used directly with any function block in the Gantry toolbox, however data from this structure can be moved into the GantryStruct prior to executing a motion function.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyGantryPositions	**GantryPositions**	**Three dimensional locations for positioning a gantry system**	
U	X	XPos	Array for grid coordinate positions.	MyGantryPositions.X[0]
U	Y	YPos	Array for grid coordinate positions.	MyGantryPositions.Y[0]
U	Z	ZPos	Array for grid coordinate positions.	MyGantryPositions.Z[0]
U	W	WPos	Array for grid coordinate positions.	MyGantryPositions.W[0]

Data Type: GantryStruct

This datatype contains all information pertaining to a gantry system.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyGantryStruct	**GantryStruct**		
U	ID	INT	Can be used to uniquely identify more than one gantry in a system.	MyGantryStruct.ID
U	Virtual	AxisStruct	All data pertaining to the Virtual axis.	MyGantryStruct.Virtual
U	X	AxisStruct	All data pertaining to the X axis.	MyGantryStruct.X
U	Y	AxisStruct	All data pertaining to the Y axis.	MyGantryStruct.Y
U	Z	AxisStruct	All data pertaining to the Z axis.	MyGantryStruct.Z
U	W	AxisStruct	All data pertaining to the W axis.	MyGantryStruct.W
U	XPrime	AxisStruct	All data pertaining to the XPrime axis.	MyGantryStruct.XPrime
U	YPrime	AxisStruct	All data pertaining to the YPrime axis.	MyGantryStruct.YPrime
U	ZPrime	AxisStruct	All data pertaining to the ZPrime axis.	MyGantryStruct.ZPrime
U	Opened	BOOL	Gripper status.	MyGantryStruct.Opened
U	Closed	BOOL	Gripper status.	MyGantryStruct.Closed
U	OpenCommand	BOOL	Gripper open request.	MyGantryStruct.OpenCommand
U	CloseCommand	BOOL	Gripper close request.	MyGantryStruct.CloseCommand
U	GripperValue	INT	Constant that equates to the gripper.	MyGantryStruct.GripperValue
U	Pick	INT	Commanded picking location row or column to be used as array index to actual position.	MyGantryStruct.Pick
U	Place	INT	Commanded placing location row or column to be used as array index to actual position.	MyGantryStruct.Place
U	Up	LREAL	mm Position of the vertical axis when "UP." Alternate usage: ZPosition.	MyGantryStruct.Up
U	Down	LREAL	mm Position of the vertical axis when "Down." Alternate usage ZPosition.	MyGantryStruct.Down
U	Velocity	LREAL	Velocity of the gantry workpiece.	MyGantryStruct.Velocity
U	Accel	LREAL	Acceleration of the gantry workpiece.	MyGantryStruct.Accel
U	Decel	LREAL	Deceleration of the gantry workpiece.	MyGantryStruct.Decel
U	ZVelocityUp	LREAL	Velocity of the vertical axis.	MyGantryStruct.ZVelocityUp
U	ZVelocityDown	LREAL	Velocity of the vertical axis.	MyGantryStruct.ZVelocityDown
U	ZAccel	LREAL	Acceleration of the vertical axis.	MyGantryStruct.ZAccel
U	ZDecel	LREAL	Deceleration of the vertical axis.	MyGantryStruct.ZDecel

Data Type: PathDetails

For use with the PathGenerator Function Block

Data Type Declaration

```
PathDetails:STRUCT
  SegmentType:INT;      (* Indicates linear or arc, see TB_PatternType *)
  XCoord:LREAL;         (* If Linear segment, the absolute coordinate of the X axis relative to the start of the path. *)
  YCoord:LREAL;         (* If Linear segment, the absolute coordinate of the Y axis relative to the start of the path. *)
  Radius:LREAL;         (* If Arc segment, the radius of the arc in XY user units. *)
  StartAngle:LREAL;     (* If Arc segment, the starting angle on a unit circle, 0 degree = 3 O'Clock position *)
  TraversedAngle:LREAL; (* If Arc segment, the traversed angle, where CW = negative, CCW = positive *)
  Resolution:REAL;
  OutputFlags:DWORD;    (* Indicator that can be used to control outputs along the path motion *)
  VectorPosition:LREAL; (* Calculated relative travel of the tool point for the current segment *)
END_STRUCT;
```

Data Type: PathIDStruct

This datatype contains all information pertaining to a gantry system.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyPathIDStruct	**PathIDStruct**		
U	XAxisTable	UINT	The CamTableID for the X axis	MyPathIDStruct.XAxisTable
U	YAxisTable	UINT	The CamTableID for the Y axis	MyPathIDStruct.YAxisTable
U	PathLength	LREAL	The total length of the path motion of the toolpoint, the distance the virtual master will travel to complete the path.	MyPathIDStruct.PathLength

Data Type: PathPairs

For use with the PathGenerator Function Block .

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyPathPairs	**PathPairs**		
U	PathPairs	UDINT	For use internally by the PathGenerator FB.	MyPathPairs[0]

Data Type: PathPointArray

For use with the PathGenerator Function Block.

Data Type Declaration

PathPointArray: ARRAY[0..100] OF PathDetails;

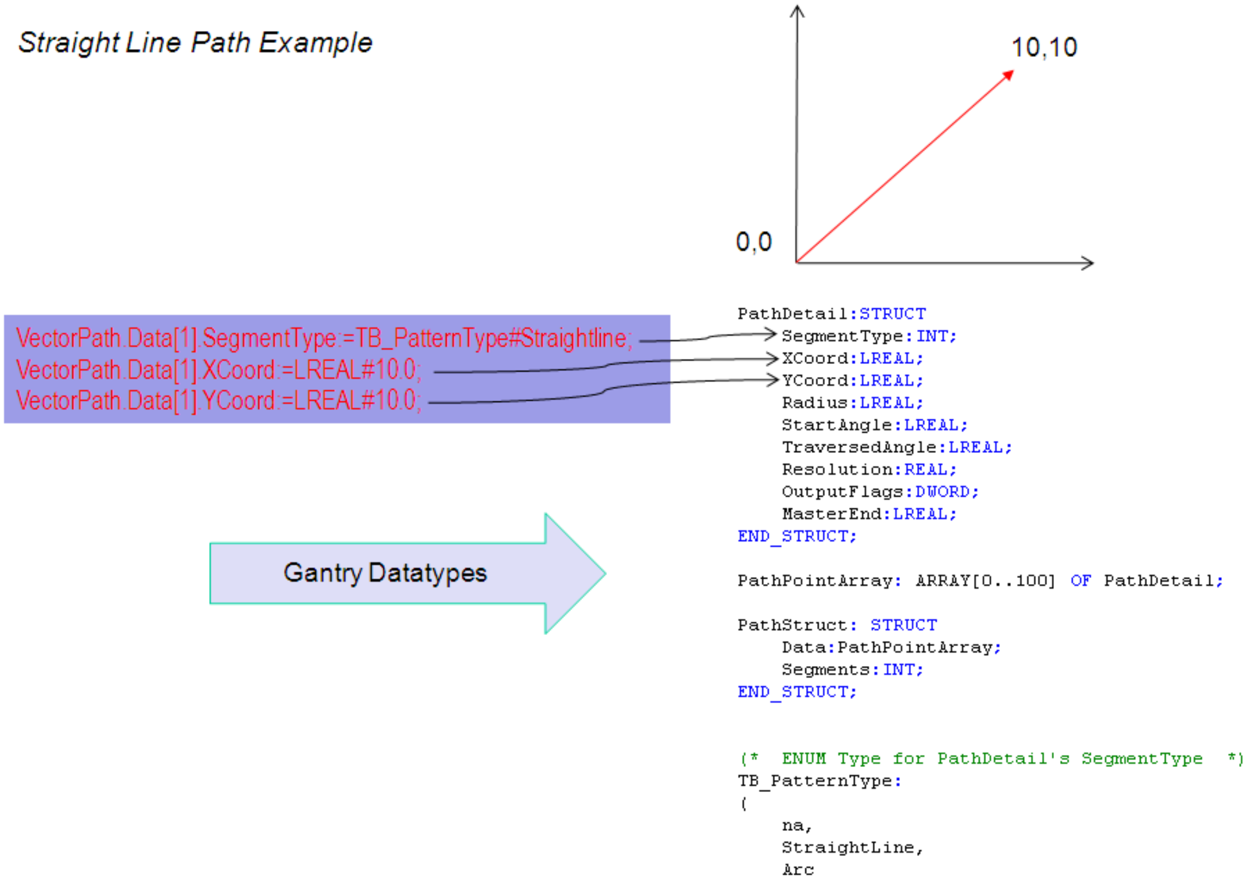
Data Type: PathStruct

For use with the PathGenerator Function Block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyPathStruct	**PathStruct**		
	Data	**PathPointArray**	**Data structure used with the PathGenerator function block.**	
U	SegmentType	INT	See TB_PatternType.	MyPathStruct.Data [0].SegmentType
U	XCoord	LREAL	If Linear segment, the absolute coordinate of the X axis relative to the start of the path.	MyPathStruct.Data [0].XCoord
U	YCoord	LREAL	If Linear segment, the absolute coordinate of the Y axis relative to the start of the path.	MyPathStruct.Data [0].YCoord
U	ZCoord	LREAL	If Linear segment, the absolute coordinate of the Z axis relative to the start of the path.	MyPathStruct.Data [0].ZCoord
U	Radius	LREAL	If Arc segment, the radius of the arc in XY user units.	MyPathStruct.Data [0].Radius
U	StartAngle	LREAL	If Arc segment, the starting angle on a unit circle, 0 degree = 3 O'Clock position.	MyPathStruct.Data [0].StartAngle
U	TraversedAngle	LREAL	If Arc segment, the traversed angle, where CW = negative, CCW = positive.	MyPathStruct.Data [0].TraversedAngle
U	Resolution	REAL	This value determines the number of interpolated points that will be calculated along the segment. Resolution is in the same units as the X and Y axes which perform arc motion.	MyPathStruct.Data [0].Resolution
U	OutputFlags	DWORD	Controls outputs along the path.	MyPathStruct.Data [0].OutputFlags
U	VectorPosition	LREAL	Calculated travel of the tool point for the Path relative to the start of all segments.	MyPathStruct.Data [0].VectorPosition
U	Segments	INT	Total datapoints specified in the path. If you need more than defined in the PathPointArray, just increase.	MyPathStruct.Segments

PathStruct Example 1

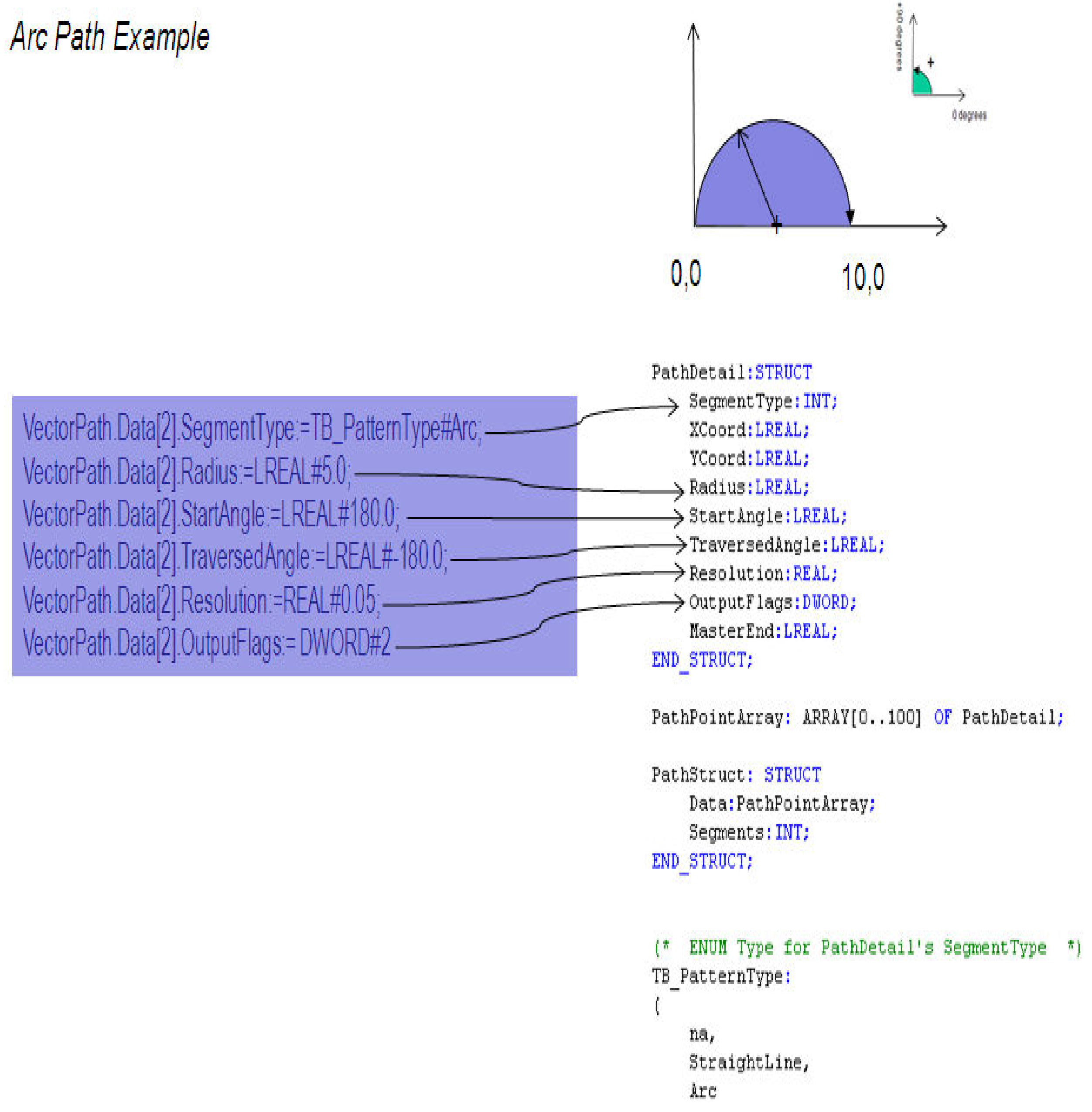


Straight Line Path Example

```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#10.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
```

```
PathDetail:STRUCT
    SegmentType:INT;
    XCoord:LREAL;
    YCoord:LREAL;
    Radius:LREAL;
    StartAngle:LREAL;
    TraversedAngle:LREAL;
    Resolution:REAL;
    OutputFlags:DWORD;
    MasterEnd:LREAL;
END_STRUCT;

PathPointArray: ARRAY[0..100] OF PathDetail;

PathStruct: STRUCT
    Data:PathPointArray;
    Segments:INT;
END_STRUCT;


(*  ENUM Type for PathDetail's SegmentType  *)
TB_PatternType:
(
    na,
    StraightLine,
    Arc
```

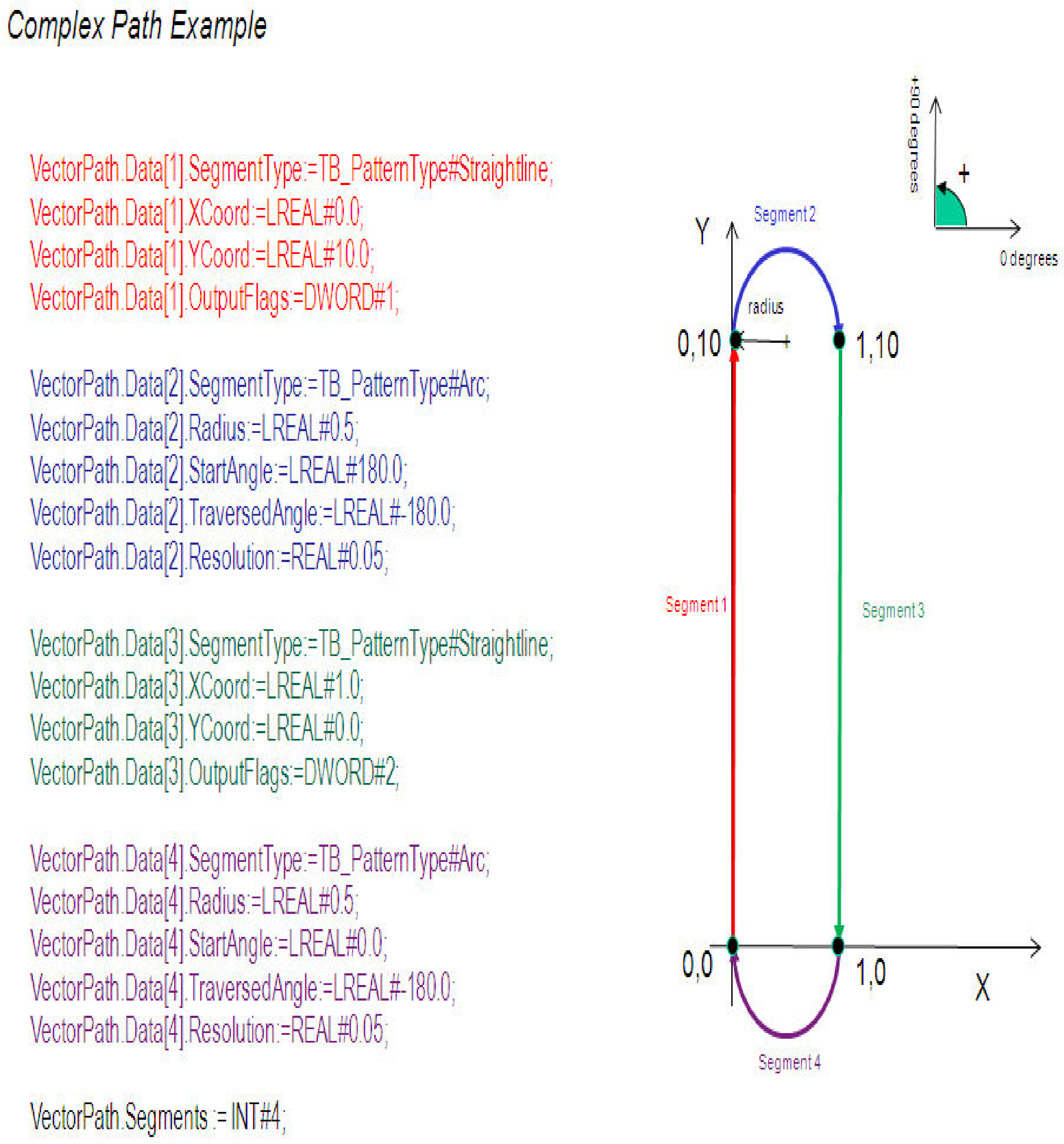
PathStruct Example 2



```
VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[2].Radius:=LREAL#5.0;
VectorPath.Data[2].StartAngle:=LREAL#180.0;
VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[2].Resolution:=REAL#0.05;
VectorPath.Data[2].OutputFlags:= DWORD#2
```

```
PathDetail:STRUCT
    SegmentType:INT;
    XCoord:LREAL;
    YCoord:LREAL;
    Radius:LREAL;
    StartAngle:LREAL;
    TraversedAngle:LREAL;
    Resolution:REAL;
    OutputFlags:DWORD;
    MasterEnd:LREAL;
END_STRUCT;

PathPointArray: ARRAY[0..100] OF PathDetail;

PathStruct: STRUCT
    Data:PathPointArray;
    Segments:INT;
END_STRUCT;


(*  ENUM Type for PathDetail's SegmentType  *)
TB_PatternType:
(
    na,
    StraightLine,
    Arc
```

PathStruct Example 3



Complex Path Example

```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#0.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
VectorPath.Data[1].OutputFlags:=DWORD#1;

VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[2].Radius:=LREAL#0.5;
VectorPath.Data[2].StartAngle:=LREAL#180.0;
VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[2].Resolution:=REAL#0.05;

VectorPath.Data[3].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[3].XCoord:=LREAL#1.0;
VectorPath.Data[3].YCoord:=LREAL#0.0;
VectorPath.Data[3].OutputFlags:=DWORD#2;

VectorPath.Data[4].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[4].Radius:=LREAL#0.5;
VectorPath.Data[4].StartAngle:=LREAL#0.0;
VectorPath.Data[4].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[4].Resolution:=REAL#0.05;

VectorPath.Segments := INT#4;
```

Data Type: SegmentArray

For use with the PathGenerator and MovePath function blocks.

Data Type Declaration

```
TYPE
   SegmentArray: ARRAY[0..200] OF SegmentDetails;
END_TYPE
```

Data Type: SegmentDetails

For use with the PathGenerator and MovePath function blocks.

Data Type Declaration

```
  TYPE
   SegmentDetails: STRUCT
        Segment: INT;                          (*    Current segment number being processed    *)
        OutputFlags: DWORD;                    (*    The output flags DWORD corresponding to the segment  *)
        VectorDistance: LREAL;                 (*    Master end point for the segment, the path travelled
                                                     up to the end of this segment  *)
    END_STRUCT;
END_TYPE
```

Data Type: SegmentStruct

For use with the PathGenerator and MovePath function blocks.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MySegmentStruct	**SegmentStruct**		
	Segment	**SegmentArray**	**For use with the PathGenerator and MovePath function blocks.**	
U	Segment	INT	Current segment number being processed.	MySegmentStruct.Segment [0].Segment
U	OutputFlags	DWORD	The output flags DWORD corresponding to the segment.	MySegmentStruct.Segment [0].OutputFlags
U	VectorDistance	LREAL	Master end point for the segment, the path traveled up to the end of this segment.	MySegmentStruct.Segment [0].VectorDistance
U	LastSegment	INT		MySegmentStruct.LastSegment

Data Type: WPos

Supporting structure for GantryPositions.

Data Type Declaration

```
TYPE
WPos: ARRAY [0..11] OF LREAL; (*   Array for grid coordinate positions   *)
END_TYPE
```

Data Type: XPos

Supporting structure for GantryPositions.

Data Type Declaration

```
TYPE
XPos: ARRAY [0..11] OF LREAL; (*   Array for grid coordinate positions   *)
END_TYPE
```

Data Type: YPos

Supporting structure for GantryPositions.

Data Type Declaration

```
TYPE
YPos: ARRAY [0..11] OF LREAL; (*   Array for grid coordinate positions   *)
END_TYPE
```

Data Type: ZPos

Supporting structure for GantryPositions.

Data Type Declaration

TYPE

ZPos: ARRAY [0..11] OF LREAL; (*   Array for grid coordinate positions   *)

END_TYPE

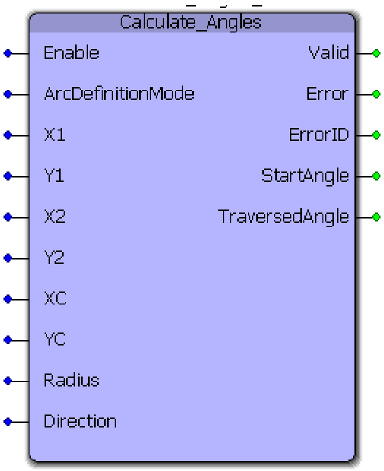
Enumerated Types for Gantry Toolbox

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_PatternType	**ENUM Type for PathDetails' SegmentType**		
	0	n/a	Not a valid PatternType
	1	StraightLine	Straight line motion between two world coordinate locations.
	2	Arc	Circle or portion of a circular path in the XY plane.
	3	StandStill	For pause between segments.
	4	WaitForInputs	Path processing will wait until the specified input conditions are met.
	5	SetTangent	Move a tangent axis to remain tangent to the XY vector path.

Calculate_Angles



This function block uses either a) two co-ordinates and center point of an arc or b) two co-ordinates and radius of an arc to calculate start and traversed angles required for PathStruct data type in the PathGenerator function block

Library

Gantry 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
B	ArcDefinitionMode	INT	Data entry mode the user wants to use. 0: Two coordinates + Center coordinate of arc, 1: Two coordinates + radius of arc	INT#0
B	X1	LREAL	X coordinate of the first coordinate.	LREAL#0.0
B	Y1	LREAL	Y coordinate of the first coordinate.	LREAL#0.0
B	X2	LREAL	X coordinate of the second coordinate.	LREAL#0.0
B	Y2	LREAL	Y coordinate of the second coordinate.	LREAL#0.0
B	XC	LREAL	X coordinate of the center coordinate.	LREAL#0.0
B	YC	LREAL	Y coordinate of the center coordinate.	LREAL#0.0
B	Radius	LREAL	Radius of arc	LREAL#0.0
B	Direction	MC_Direction	0: clockwise, 1: counter clockwise	MC_Direction#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.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
B	StartAngle	LREAL	Angle subtended by a line drawn from the arc center to the start point of the arc with the positive X axis on an XY plane
B	TraversedAngle	LREAL	Angle traversed by the arc generated

Notes

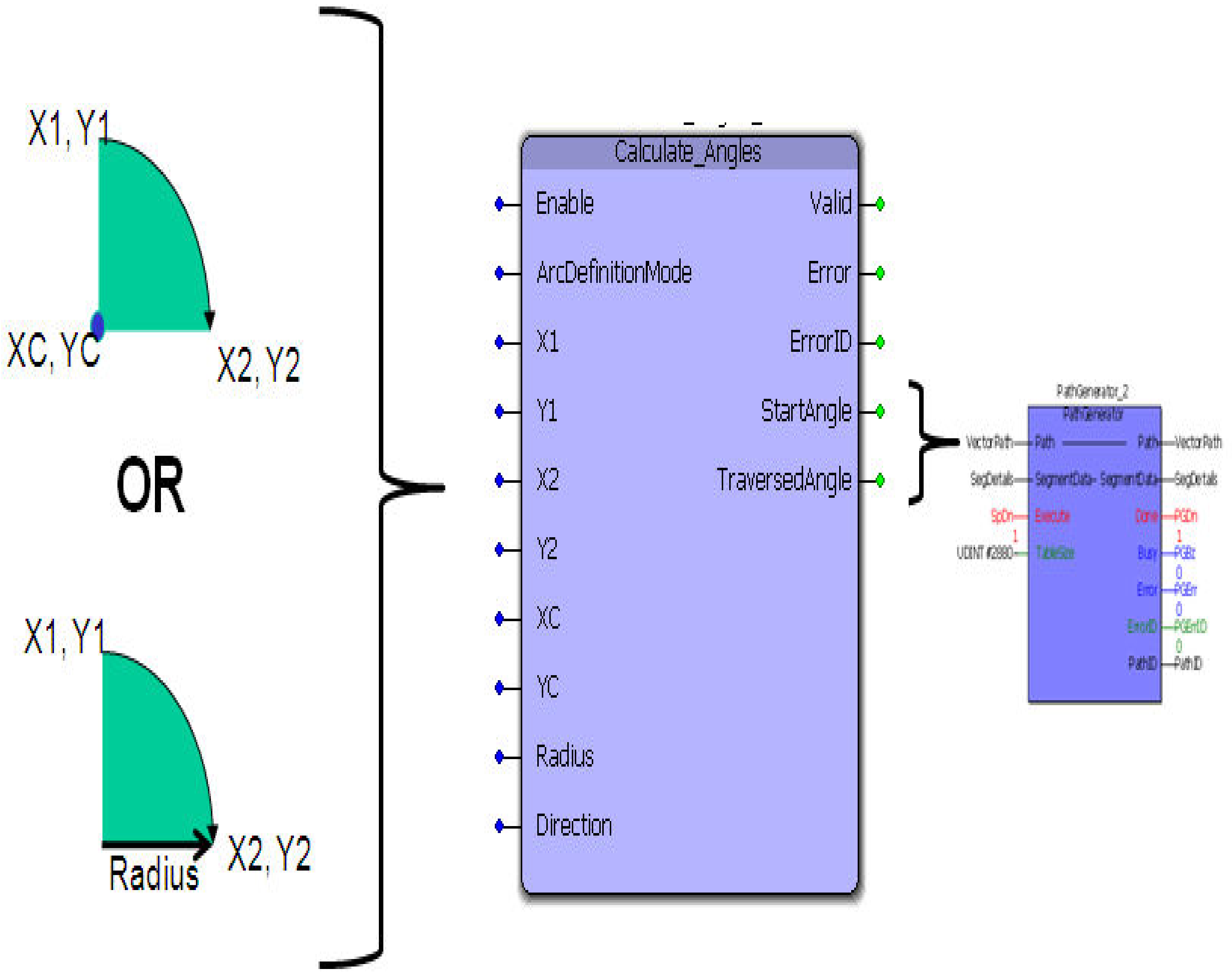
- See Yaskawa's Youtube channel for more info, details, and examples.

Error Description

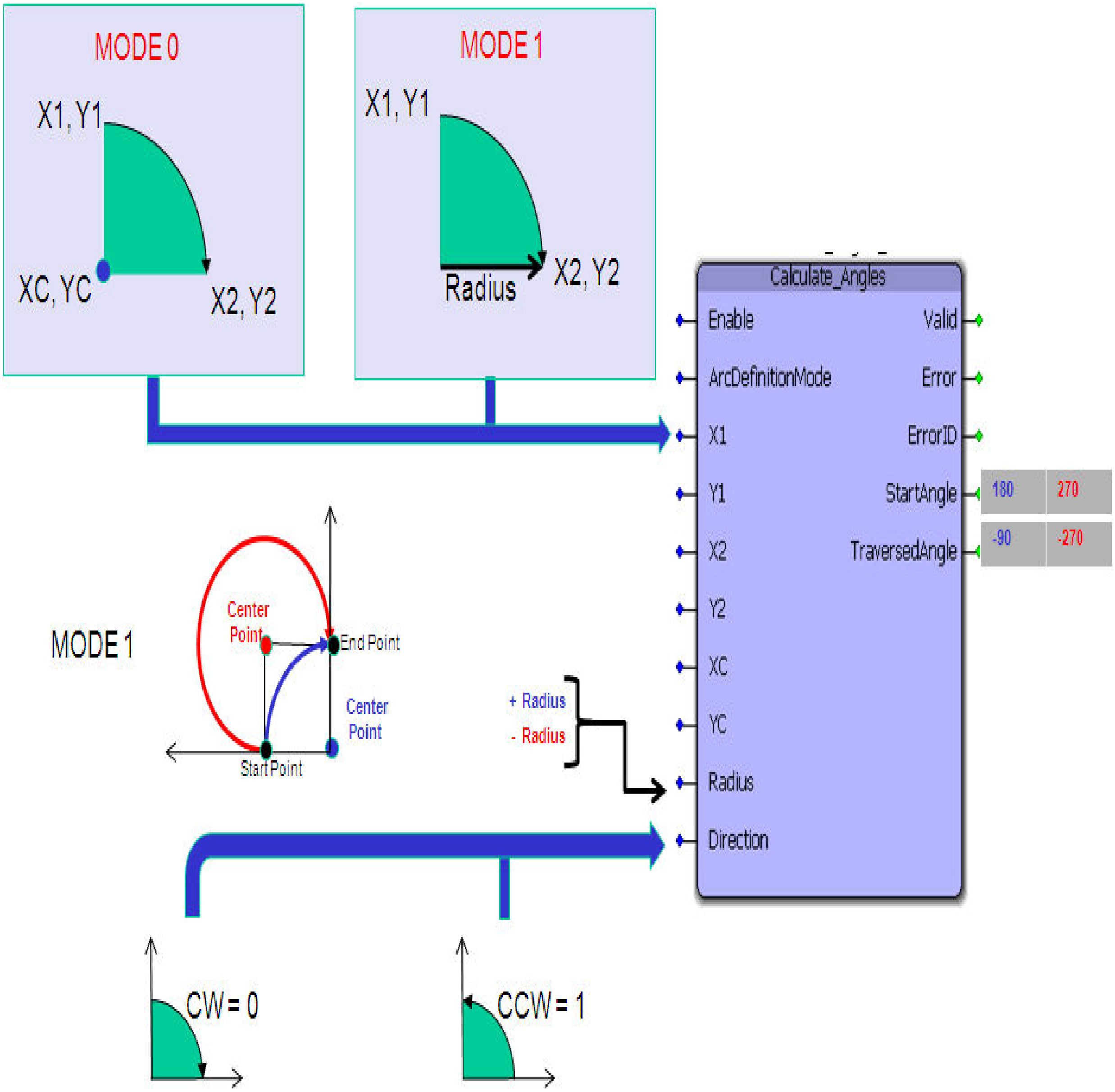
ErrorID	Meaning
0	No error.
10130	The center to co-ordinate distance for the two input co-ordinates are not the same
10131	Zero radius is invalid.
10132	Only modes 0 (center + 2 co-ordinates) and 1 (radius + 2 coordinates) are supported.
10133	The coordinates of the two data points are the same.
10140	Must be greater than zero and less than 20.

Example

The Calculate_Angles function block is used to calculate Start and Traversed angles which can be used by the PathStruct structure to create a path in the PathGenerator function block. The two modes of data entry for an arc are a) two co-ordinates and center point of an arc or b) two co-ordinates and radius as shown below.

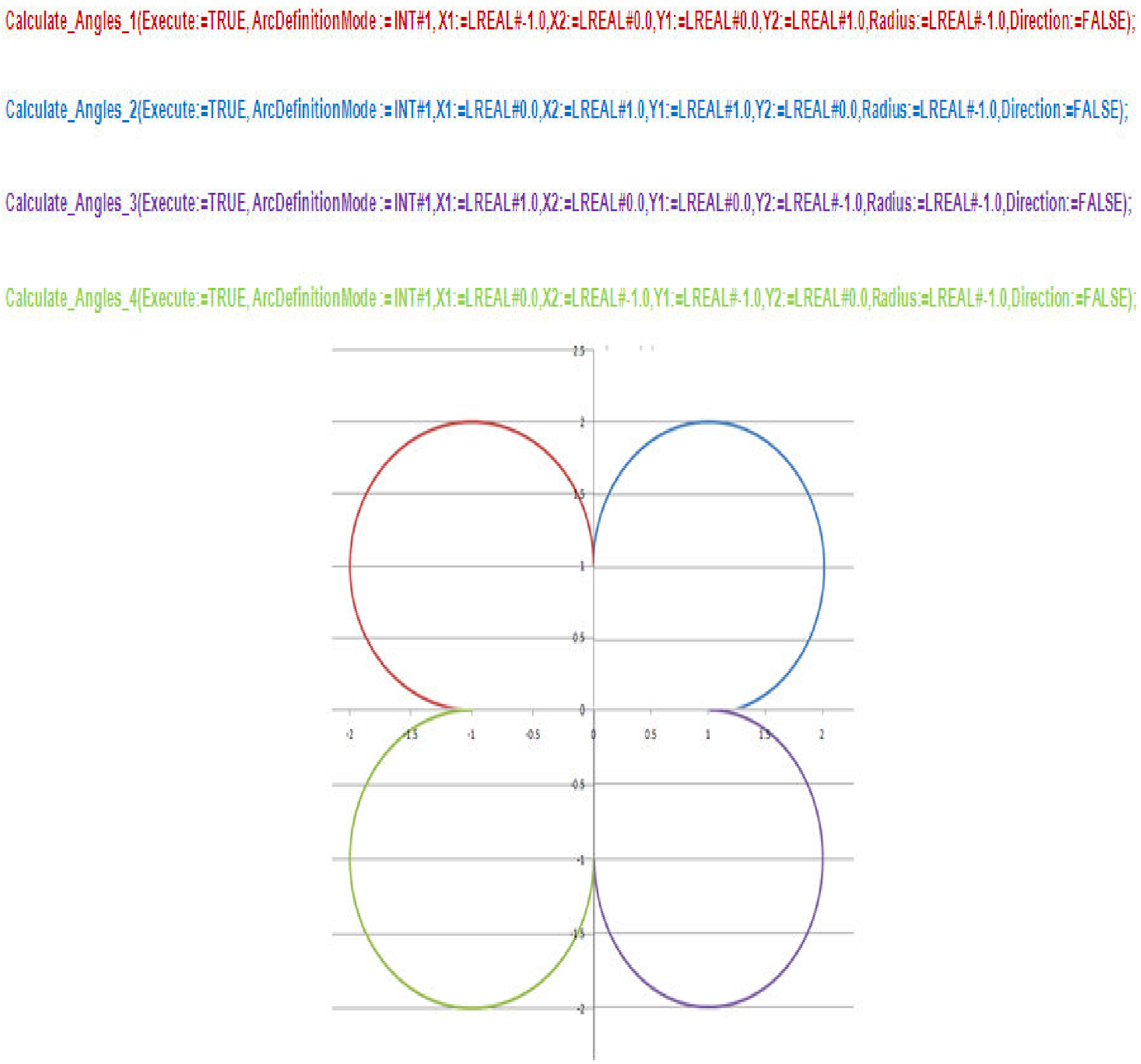


The two modes of data entry are shown in detail below. Mode 0: 2 coordinates + center coordinate, Mode 1: 2 coordinates + radius. If the user plans to use Mode 1, the sign of the radius is important. this is illustrated in the figure below. The two arcs (red and blue) have the same start and end coordinates and they have the same radii. A negative radius would give rise to an obtuse arc (shown as red) and the start angle and traversed angle are 270 and -270 respectively. If a positive radius is specified, an acute arc (shown in blue) is generated. The start angle and traversed angle for the acute arc are 180 and -90 respectively.



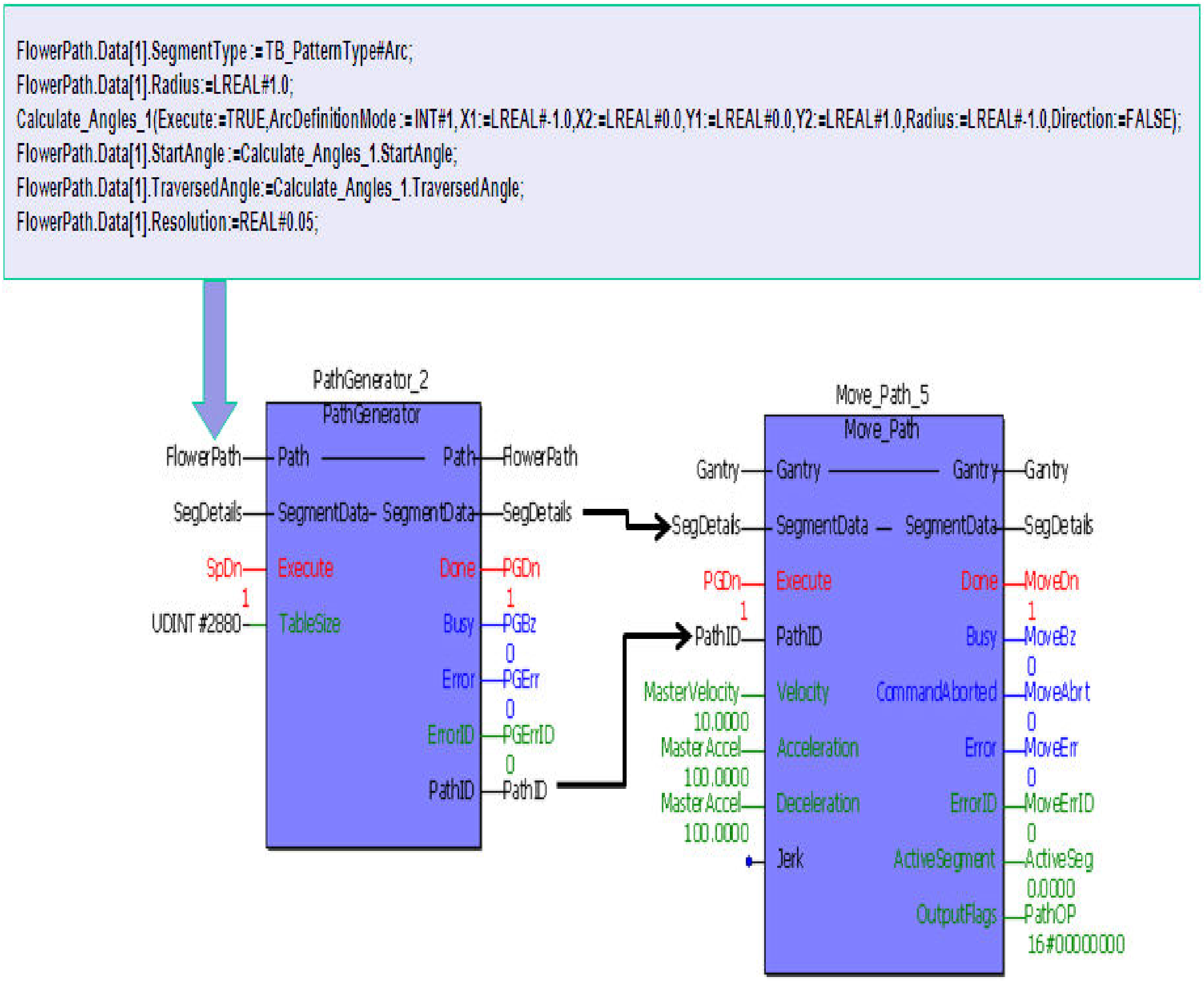
Application example

Step1: Using Calculate_Angles to calculate start and traverse angles for the flower path shown below



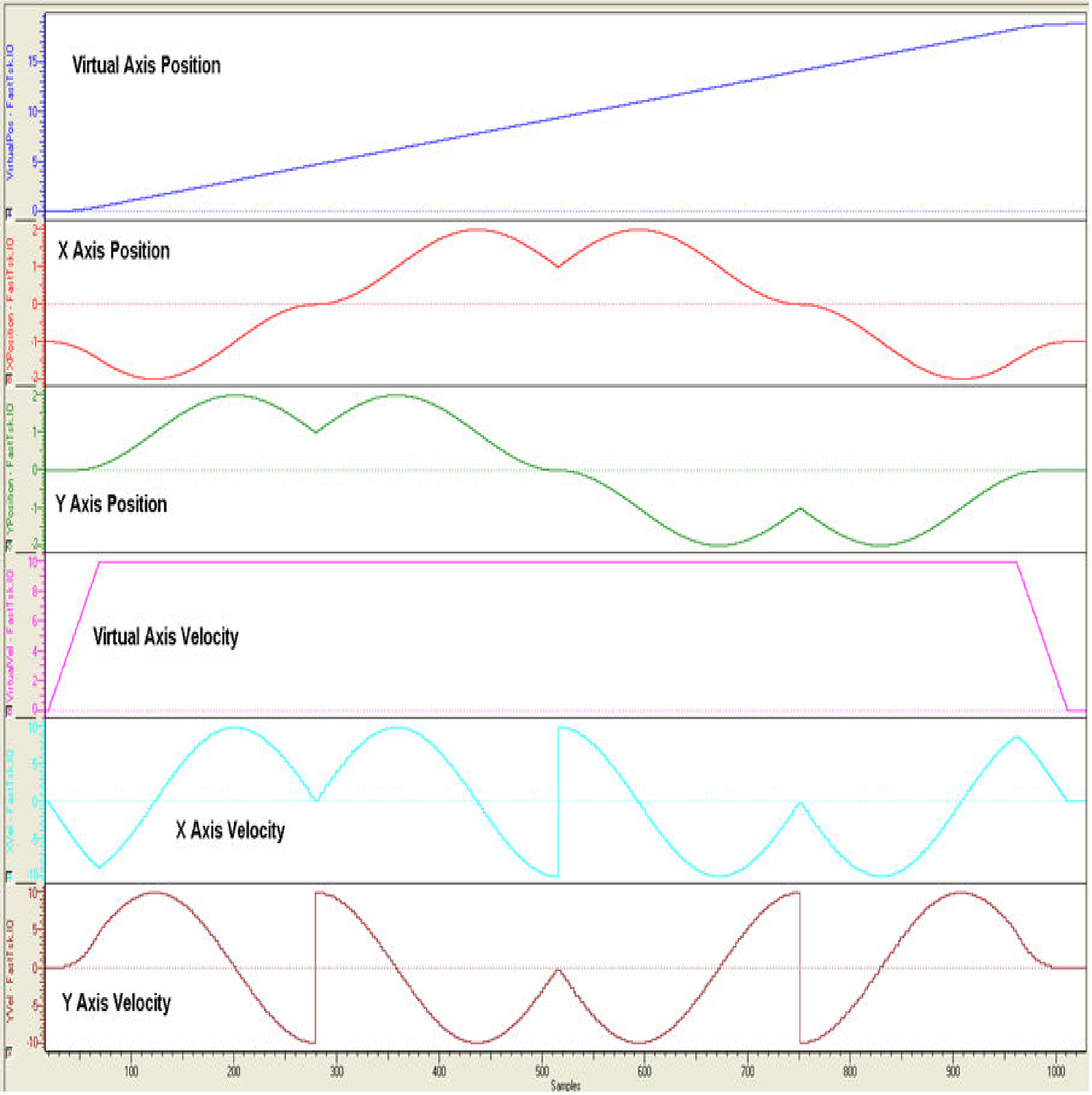
```
Calculate_Angles_1(Execute:=TRUE, ArcDefinitionMode := INT#1, X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_2(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#0.0,X2:=LREAL#1.0,Y1:=LREAL#1.0,Y2:=LREAL#0.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_3(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#-1.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_4(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#0.0,X2:=LREAL#-1.0,Y1:=LREAL#-1.0,Y2:=LREAL#0.0,Radius:=LREAL#-1.0,Direction:=FALSE);
```

Step 2: Use PathGenerator to create the path and Move_Path to implement XY motion

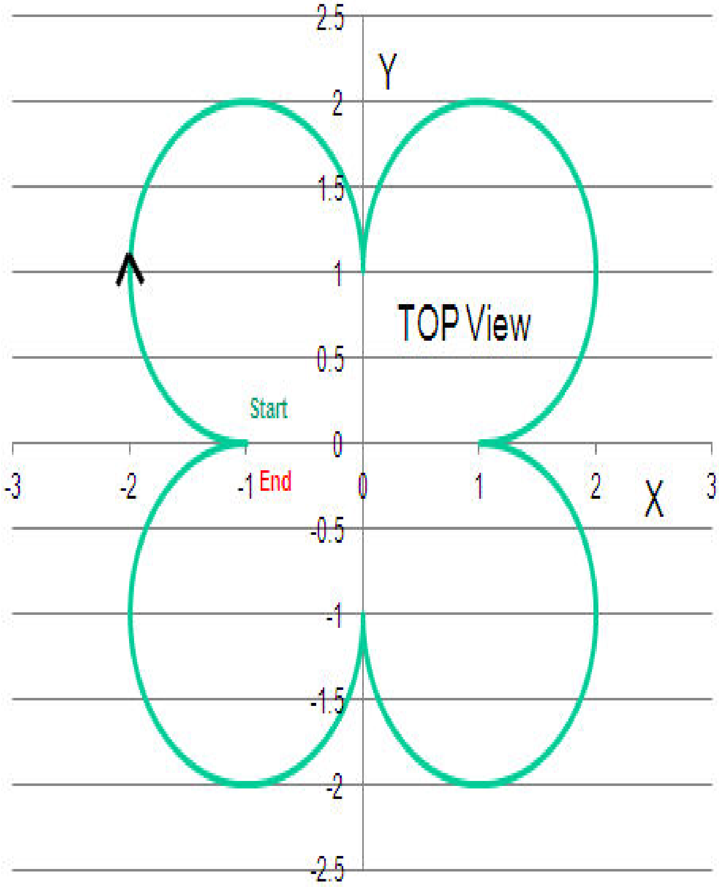


```
FlowerPath.Data[1].SegmentType:=TB_PatternType#Arc;
FlowerPath.Data[1].Radius:=LREAL#1.0;
Calculate_Angles_1(Execute:=TRUE,ArcDefinitionMode:=INT#1,X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);
FlowerPath.Data[1].StartAngle:=Calculate_Angles_1.StartAngle;
FlowerPath.Data[1].TraversedAngle:=Calculate_Angles_1.TraversedAngle;
FlowerPath.Data[1].Resolution:=REAL#0.05;
```

Step 3: Validation using logic analyzer

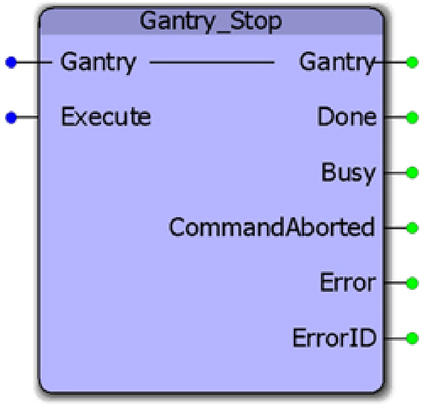


Step 4: Result on XY system



2.5
Y
2
1.5
1
TOP View
0.5
Start
0
-3
-2
-1
End
0
1
2
X
3
-0.5
-1
-1.5
-2
-2.5

Gantry_Stop



This function block will execute the MC_Stop block for all axes configured as part of a gantry system.

Library

Gantry Toolbox

Parameters

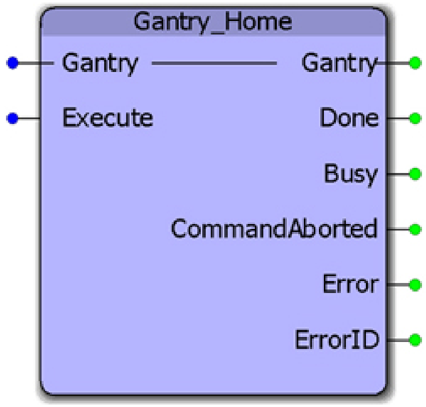
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger 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.	
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

ErrorID	Meaning
0	No error.
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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4660	Deceleration is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
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.

Example

Gantry_Home



This function block will move all gantry axes in search of home by first seeking one of the limit switches, and then searching in the other direction for the C channel or index pulse. This block uses the Home_LS_Pulse function block from the PLCopen Toolbox. If configured, the Z axis will search for home first, then the X and Y axes will search simultaneously. This sequence was designed to prevent mechanical interferences with objects in the work coordinate system during the homing process.

Library

Gantry Toolbox

Parameters

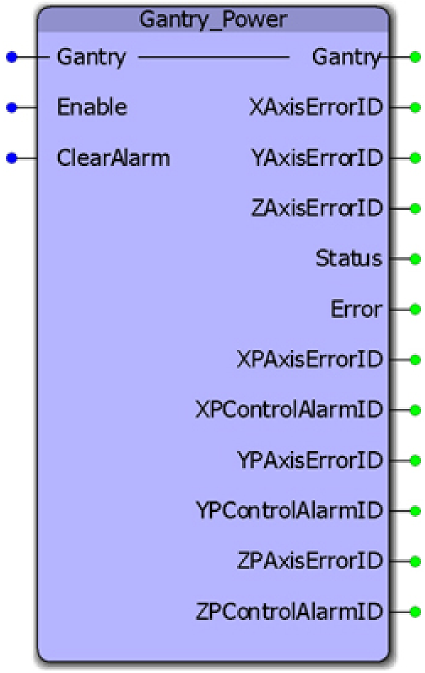
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger 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.	
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

ErrorID	Meaning
0	No error.
1	Time limit exceeded.
2	Distance limit exceeded.
3	Torque limit exceeded.
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. 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.
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 aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
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.
4396	Axis latch function already in use.
4397	Over travel is limit still ON after attempting to move away from it.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4646	Mode does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10037	Offset cannot be in the same direction as the original motion into the limit switch.
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.
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.

Example

Gantry_Power



This function block will enable or disable all axes configured as part of a gantry system. This block uses the AxisControl function block from the PLCopen Toolbox. If the gantry is configured with dual motors on the same physical axis, then the secondary or prime axes are geared to the other axis in the same physical motion plane.

Library

Gantry Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
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	ClearAlarm	BOOL	This input will clear any axis specific alarms on the Gantry axes.	FALSE
VAR_OUTPUT				
V	XAxisErrorID	UINT	ErrorID on the X axis.	
V	YAxisErrorID	UINT	ErrorID on the Y axis.	

V	ZAxisErrorID	UINT	ErrorID on the Z axis.
B	Status	BOOL	TRUE if the drive is enabled. This output is derived from the Status output of MC_Power.
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.
V	XPAxisErrorID	UINT	ErrorID on the X' axis.
V	XPControlAlarmID	UINT	Controller ErrorID caused by the X' axis.
V	YPAxisErrorID	UINT	ErrorID on the Y' axis.
V	YPControlAlarmID	UINT	Controller ErrorID caused by the Y' axis.
V	ZPAxisErrorID	UINT	ErrorID on the Z' axis.
V	ZPControlAlarmID	UINT	Controller ErrorID caused by the Z' axis.

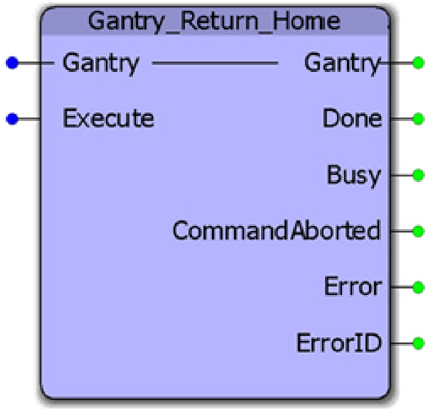
Error Description

This function block uses the AxisControl function block from the PLCopen Toolbox. Refer to the Error IDs from the Axis Control function block.

ErrorID	Meaning
0	No error.
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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
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.
45332	Sending clear alarms command to servo drive failed.
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.
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.

Example

Gantry_Return_Home



This function block will move all gantry axes back to the home position as defined by the home positions in the GantryStruct. If configured, the Z axis will move to home first, then the X and Y axes will move together. This sequence was designed to prevent mechanical interferences with objects in the work coordinate system during the homing process. This block uses the MC_MoveAbsolute function block from the PLCopenPlus firmware library. It is assumed that the home location has been previously determined either by using the Gantry_Home function block or because the system uses absolute encoders that have been calibrated to the physical machine.

Library

Gantry Toolbox

Parameters

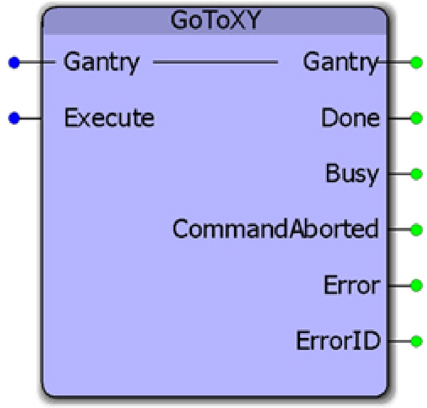
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and 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

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10034	Interpolation calculation 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.

Example

GotoXY



This function block will perform an absolute move the X and Y axes to a specific location within the gantry coordinate system. The absolute X and Y positions must be specified in GantryStruct before executing this function block. This block calculates the required acceleration, deceleration and velocity for each axis and then executes an MC_MoveAbsolute function block simultaneously for each to create straight line motion at the tool point, however this is not considered an interpolated motion. If configured, no motion on the Z axis will occur.

Library

Gantry Toolbox

Parameters

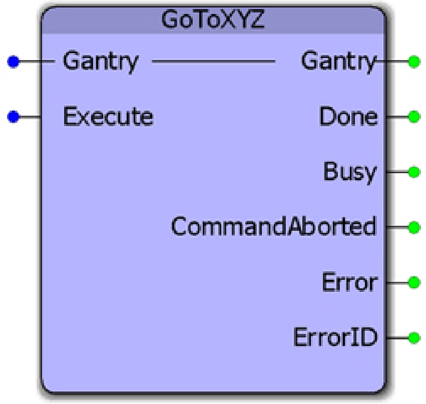
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger 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.	
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

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10034	Interpolation calculation 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.

Example

GoToXYZ



This function block will perform an absolute move the X, Y, and Z axes to a specific location within the gantry coordinate system. The absolute positions must be specified in GantryStruct before executing this function block. This block calculates the required acceleration, deceleration and velocity for each axis and then executes an MC_MoveAbsolute function block simultaneously for each to create straight line motion at the tool point, however this is not considered an interpolated motion.

Library

Gantry Toolbox

Parameters

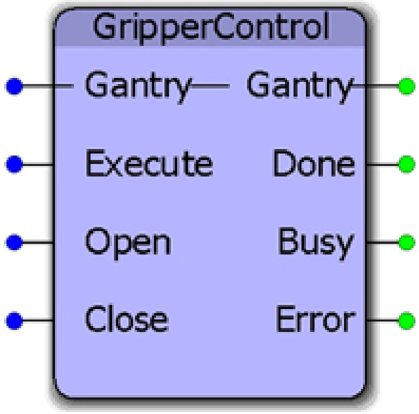
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger 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

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10034	Interpolation calculation 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.

Example

GripperControl



This function block can operate a simple gripper device if the actuator can be controlled via a digital output. It will activate an output while waiting for confirmation that a corresponding input has changed state to indicate that the gripper has successfully opened or closed.

Library

Gantry Toolbox

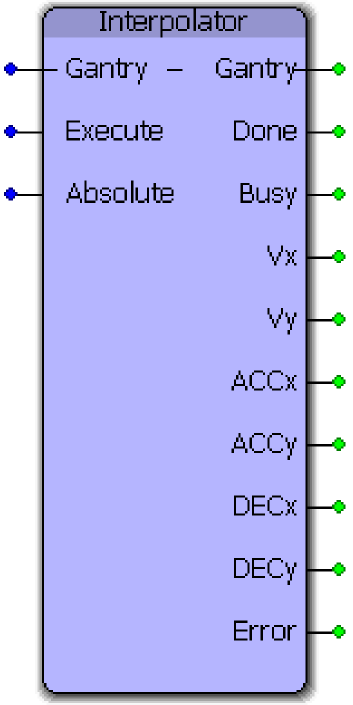
Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
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	Open	BOOL	Command to open the gripper	
V	Close	BOOL	Command to close the gripper	
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.	

Error Description

ErrorID	Meaning
0	No error.
10035	Gripper Close Error (Timeout).
10036	Gripper Open Error (Timeout).

Interpolator



This function block calculates the required acceleration, deceleration, and velocity for both X and Y axes so that straight line motion can occur between any two points in the XY (two dimensional) coordinate system. This function block is used by the GotoXY function block.

Library

Gantry Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
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	

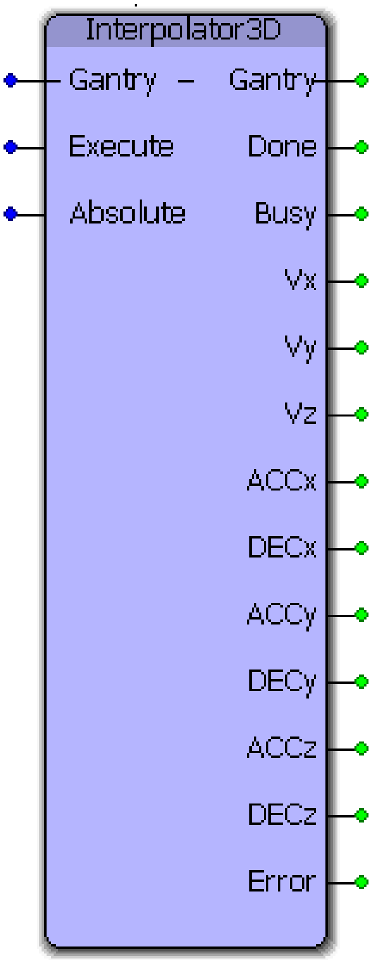
			output is reset when Execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the Execute input, and reset when Done, CommandAborted, or Error is true. In the case of a function block with an Enable input, a Busy output indicates the function is operating, but not ready to provide Valid information. (No Error)
V	Vx	LREAL	X axis component of gantry velocity
V	Vy	LREAL	Y axis component of gantry velocity
V	ACCx	LREAL	X axis component of gantry acceleration
V	ACCy	LREAL	Y axis component of gantry acceleration
V	DECx	LREAL	X axis component of gantry deceleration
V	DECy	LREAL	Y axis component of gantry deceleration
V	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.

Error Description

ErrorID	Meaning
0	No error.
10034	Interpolation calculation error.

Example

Interpolator3D



This function block calculates the required acceleration, deceleration, and velocity for X, Y and Z axes so that straight line motion can occur between any two points in three dimensional space within the gantry coordinate system. This function block is used by the GotoXYZ function block.

Library

Gantry Toolbox

Parameters

*	Parameter	Data Type	Description

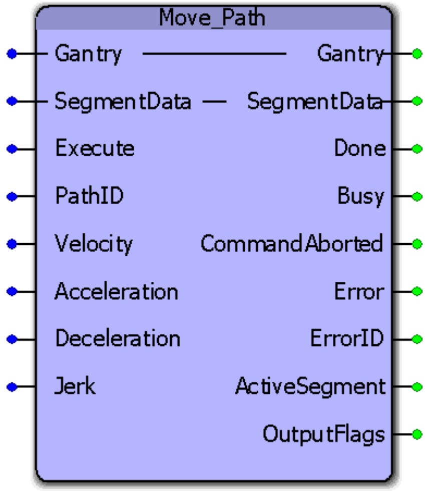
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
B	Absolute	BOOL		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	Vx	LREAL	X axis component of gantry velocity.	
V	Vy	LREAL	Y axis component of gantry velocity.	
V	Vz	LREAL	Z axis component of gantry velocity.	
V	ACCx	LREAL	X axis component of gantry acceleration.	
V	DECx	LREAL	X axis component of gantry deceleration.	
V	ACCy	LREAL	Y axis component of gantry acceleration.	
V	DECy	LREAL	Y axis component of gantry deceleration.	
V	ACCz	LREAL	Z axis component of gantry acceleration.	
V	DECz	LREAL	Z axis component of gantry deceleration.	
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.	

Error Description

ErrorID	Meaning
0	No error.
10034	Interpolation calculation error.

Example

Move_Path



Based on the axes specified in the GantryStruct, this function block can move X,Y,Z and Tangent axes according to a path profile generated by the PathGenerator and specified in the PathStruct structure. This function block typically uses the output from the PathGenerator to operate. Inputs and outputs can be monitored and controlled along the path.

Library

Gantry Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
V	SegmentData	SegmentStruct	Structure of data that contains the segment number, output code, and tool path endpoint for each segment in the motion path.	
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	PathID	PathIDStruct	Structure containing data to be shared between PathGenerator and MovePath functions.	n/a
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	*Jerk*	*LREAL*	*[[[Undefined variable Primary.ParameterNotSupported]]]Value of the jerk in [user units / second^3].*	*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	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	ActiveSegment	INT	Indicates the active segment as the tool point moves along the path.	
V	OutputFlags	DWORD	Code which can be used to set up to 32 different outputs at various points along the motion path.	

Notes

- The motion path described is relative to the start point of the move. The axes can be moved using other motion blocks prior to executing Move_Path to account for offsets required.
- The Gantry structure used with this function block must include a Virtual axis. Configure a virtual axis in the Hardware Configuration. The virtual axis must have the same units as the XYZ axes.
- Gantry Toolbox v204 has improved code to allow for changing the Velocity, Acceleration, and Deceleration on the fly, including zero speed velocity to pause a path in progress.
- See Yaskawa's YouTube channel for more info, details, and examples.

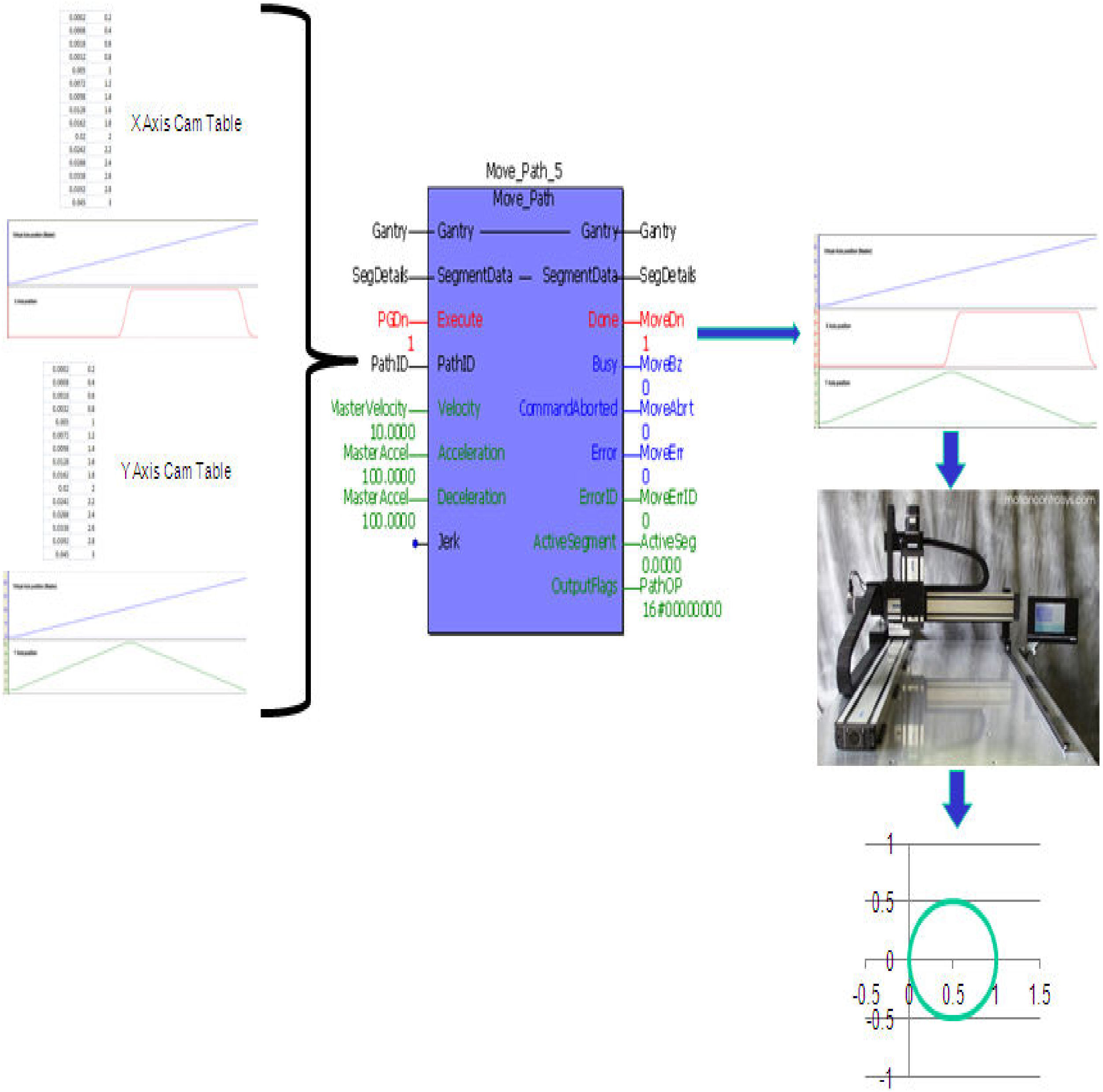
Error Description

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4380	MC_SetPosition cannot be executed while the axis is already moving.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
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.
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.

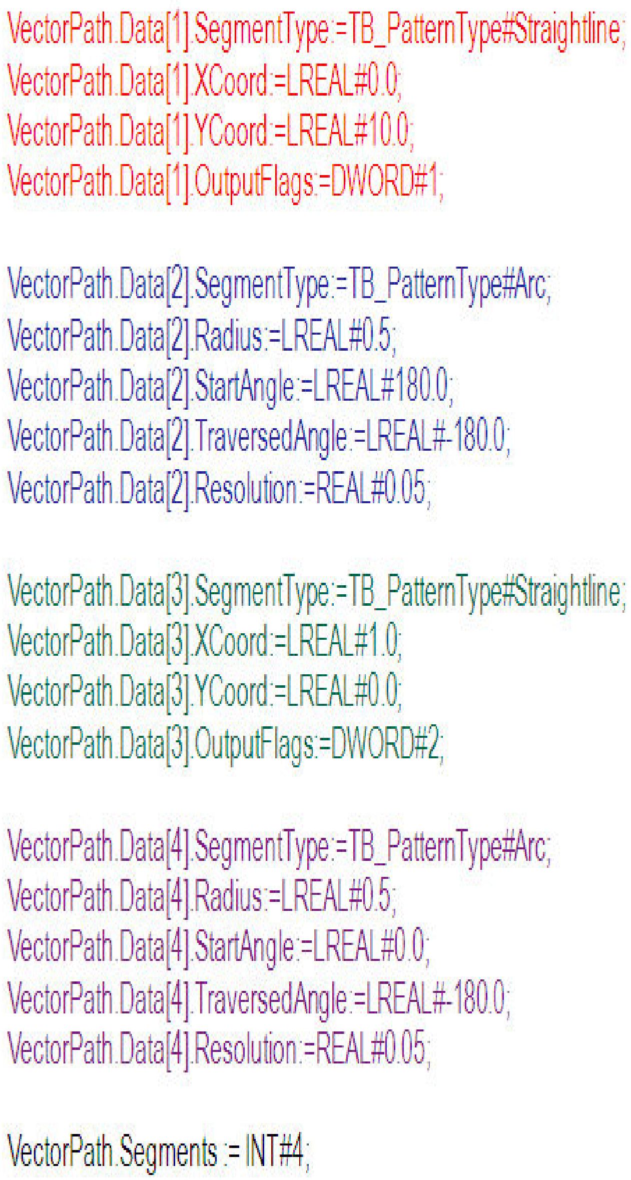
4395	Window parameters are outside of the master axis' machine cycle. (0 to Prm 1502, the last master position in the active cam table.)
4625	AXIS_REF.AxisNum does not correspond to an axis configured on the system. Verify the value of AxisNum matches a logical axis number in the configuration. Tip: Make sure a variable of type AXIS_REF is properly declared as a VAR or VAR_GLOBAL in all relevant POUs.
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.
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.
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.
4646	Mode does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4669	Engage position is outside the cam table domain.
4670	Engage window is less than zero.
4887	CamTableID does not refer to a valid cam table.
4891	The slave axis can not be the same as the master axis.
4893	The specified external axis may not be used. A physical axis is required.
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.
57617	Instance object is NULL.
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.
57874	Argument data is NULL. The EngageData input must be connected.
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.

Example

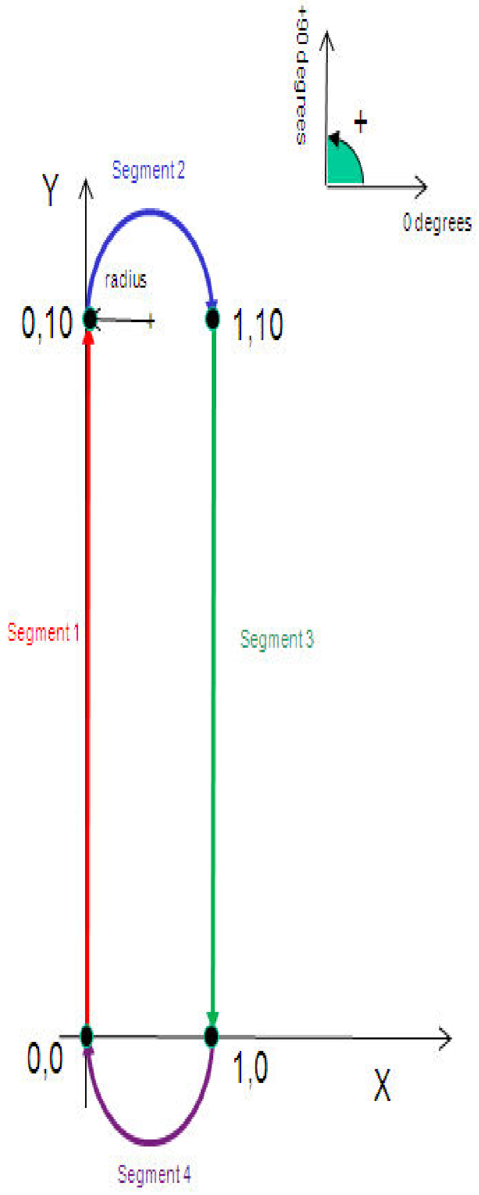
Uses the profile described by the PathStruct data type and commands motion to the X, Y axes using a virtual axis as the master. This is shown in the figure below.



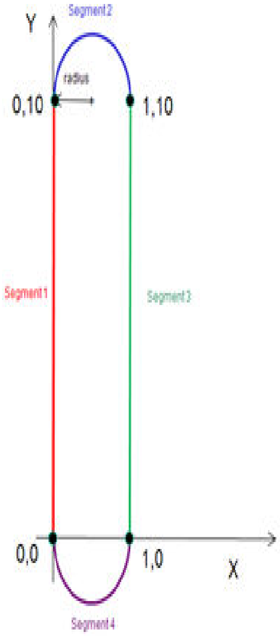
Consider the following contour:

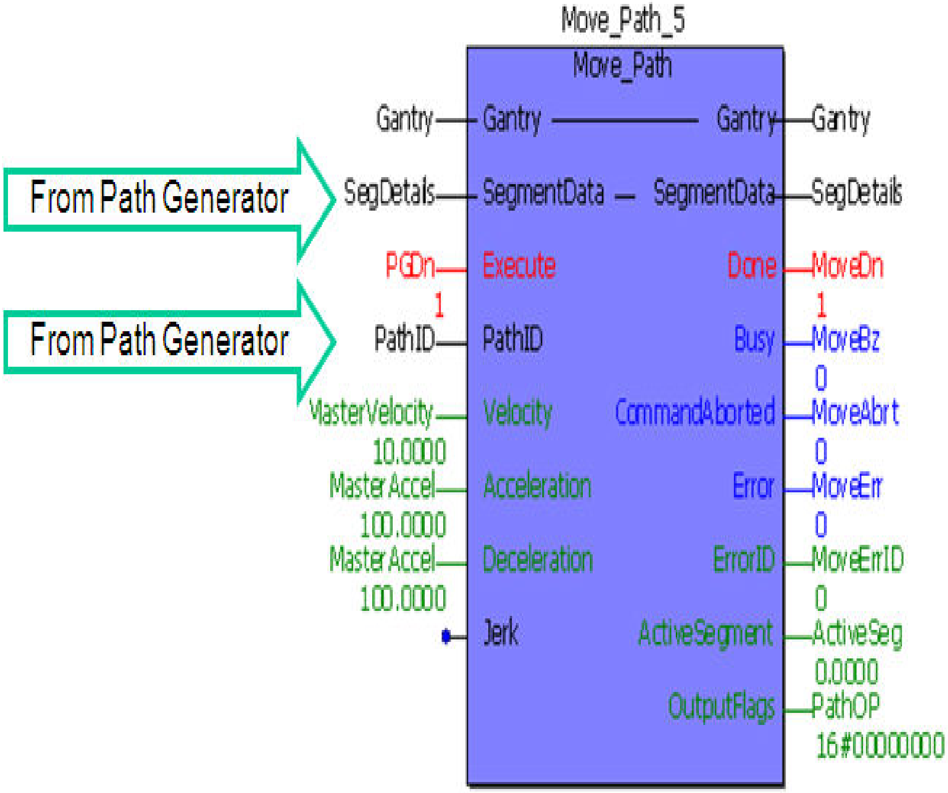


```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#0.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
VectorPath.Data[1].OutputFlags:=DWORD#1;

VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[2].Radius:=LREAL#0.5;
VectorPath.Data[2].StartAngle:=LREAL#180.0;
VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[2].Resolution:=REAL#0.05;

VectorPath.Data[3].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[3].XCoord:=LREAL#1.0;
VectorPath.Data[3].YCoord:=LREAL#0.0;
VectorPath.Data[3].OutputFlags:=DWORD#2;

VectorPath.Data[4].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[4].Radius:=LREAL#0.5;
VectorPath.Data[4].StartAngle:=LREAL#0.0;
VectorPath.Data[4].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[4].Resolution:=REAL#0.05;

VectorPath.Segments := INT#4;
```

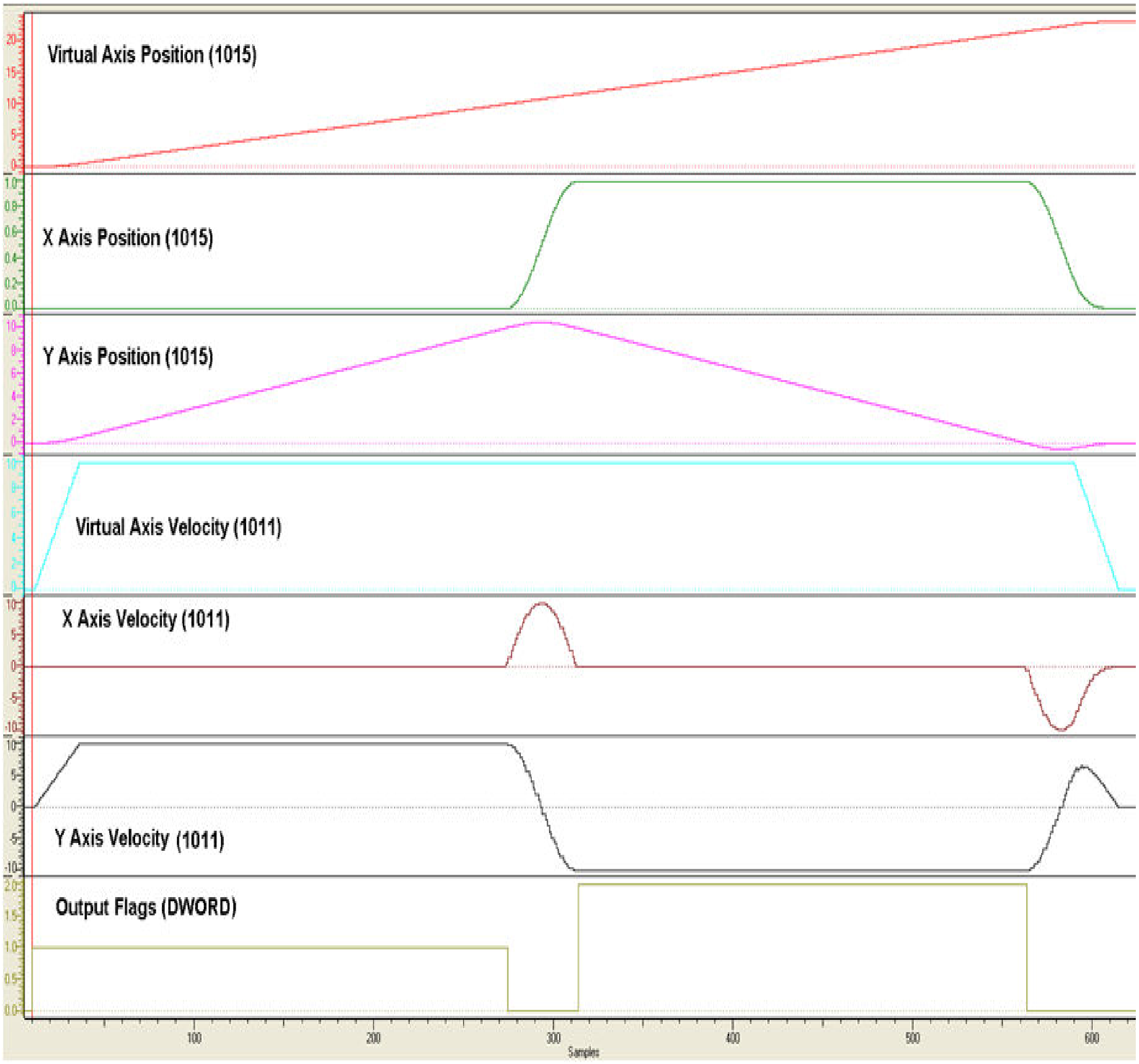


The MovePath function block uses SegmentData and PathID from the PathGenerator function block and executes moves on the X and Y axes. If a profile is made up of multiple segments (4 in the example below), the active segment output indicates which segment is being run. Output flags can be set from this function block to turn outputs on. this can be useful for applications like cutting, scoring or glue dispensing where digital outputs can be used to fire end effectors.

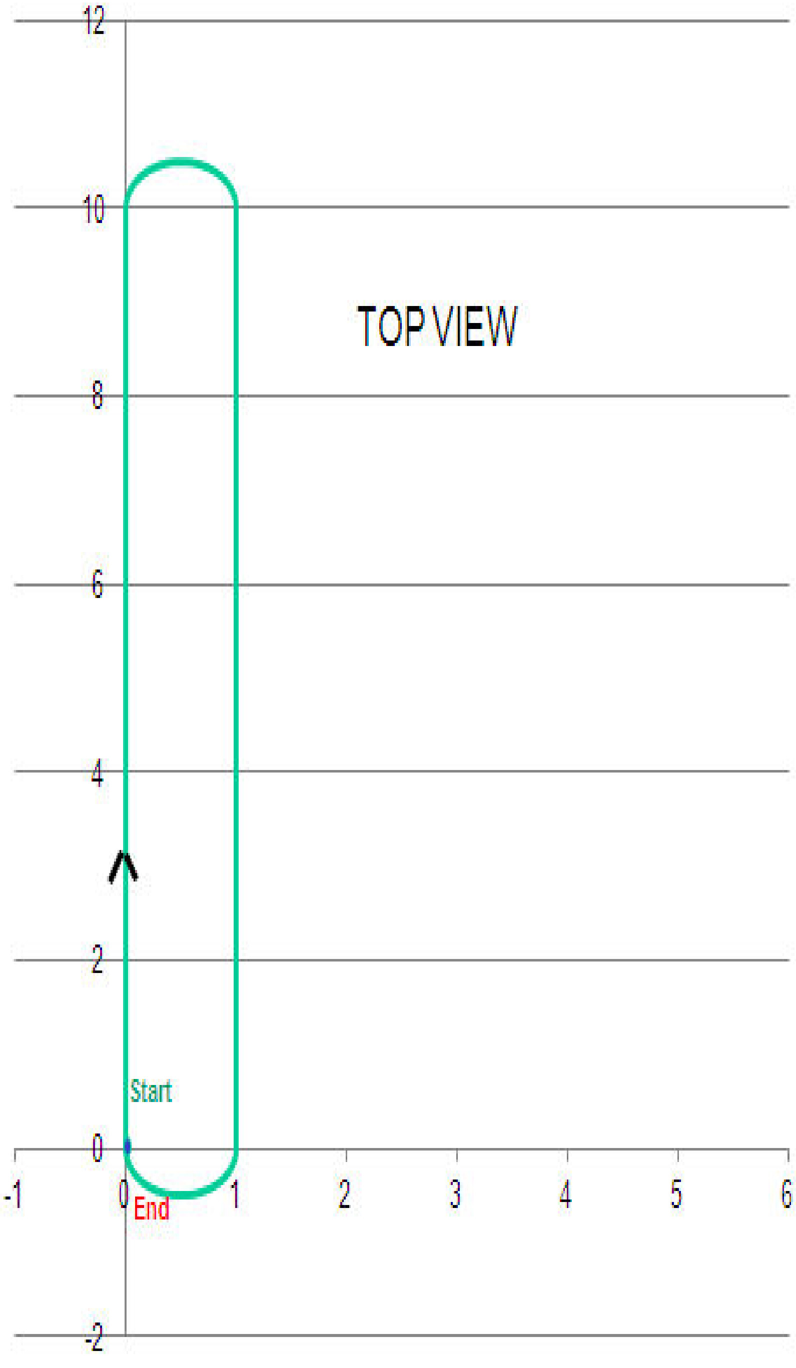




The logic analyzer plot of independent axis parameters from the above profile is given below. It can be seen that the outputs flags are set during segments 1 and 3. (defined in PathStruct)

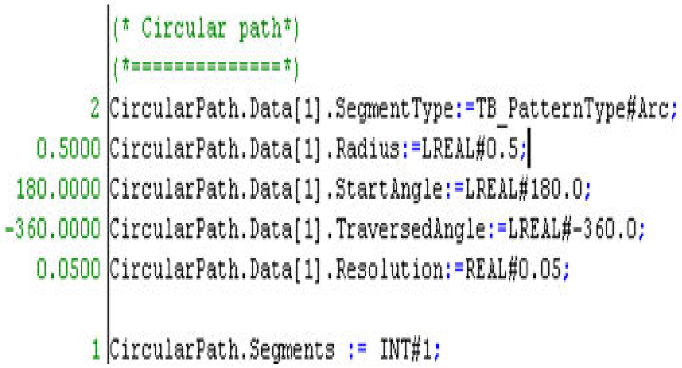


The actual profile plotted by the XY system is shown below

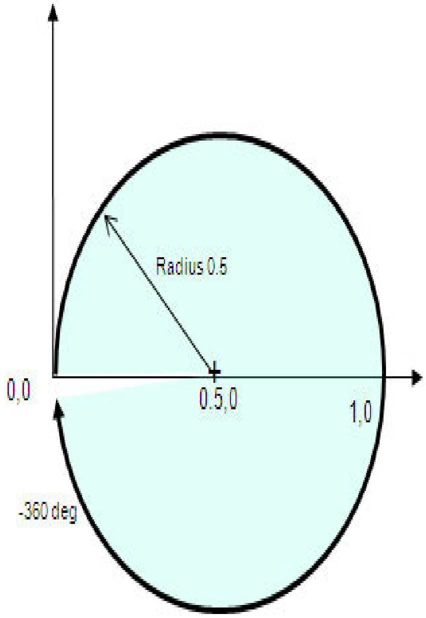


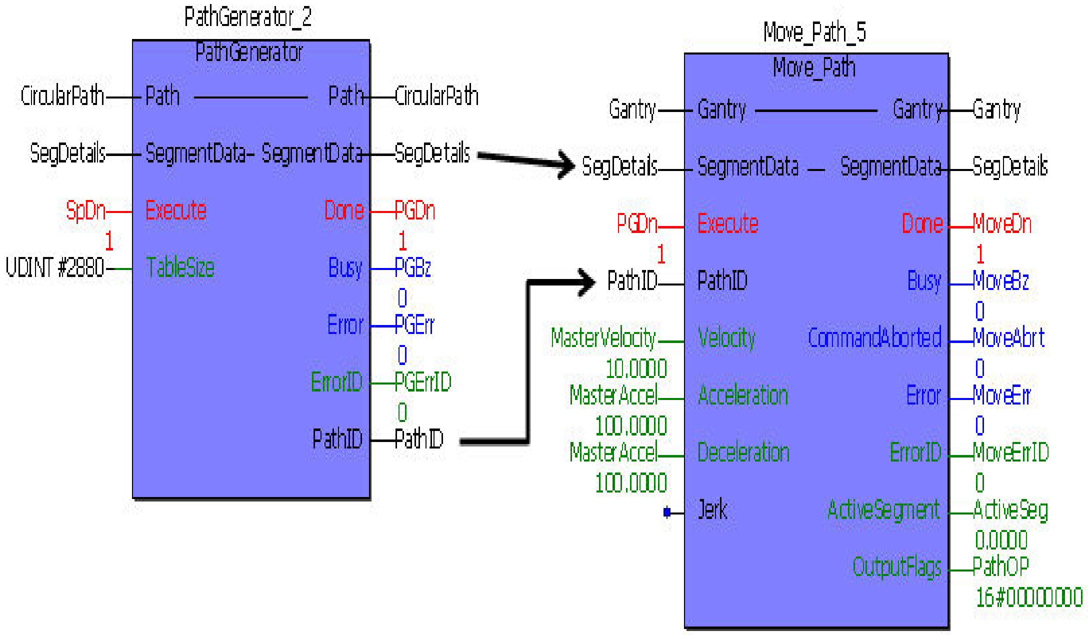
Code Example 2

Consider the following circular profile

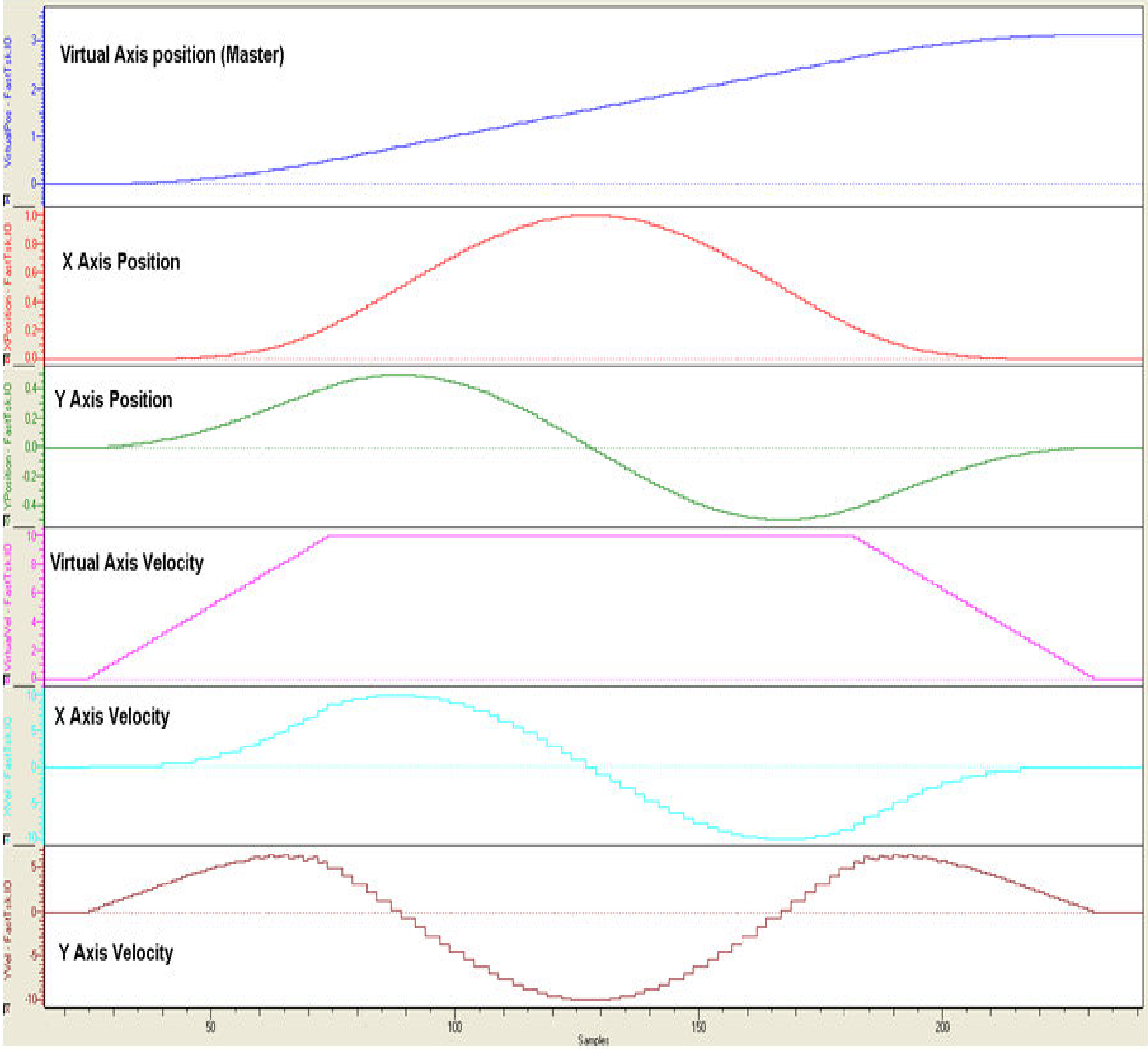


```
         (* Circular path*)
         (*===============*)
       2 CircularPath.Data[1].SegmentType:=TB_PatternType#Arc;
  0.5000 CircularPath.Data[1].Radius:=LREAL#0.5;
180.0000 CircularPath.Data[1].StartAngle:=LREAL#180.0;
-360.0000 CircularPath.Data[1].TraversedAngle:=LREAL#-360.0;
  0.0500 CircularPath.Data[1].Resolution:=REAL#0.05;

       1 CircularPath.Segments := INT#1;
```

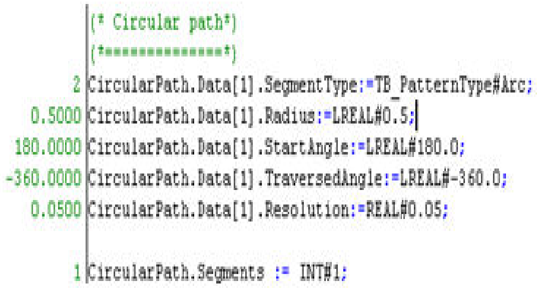




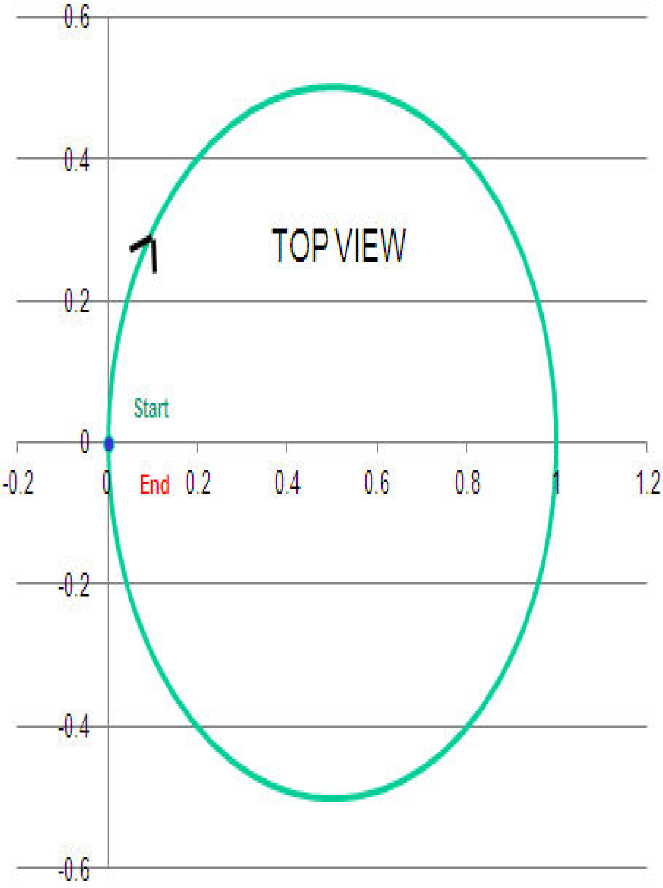
The logic analyzer traces from individual axes while Move_Path was busy is shown in the plot below



The actual profile plotted by the XY system is:

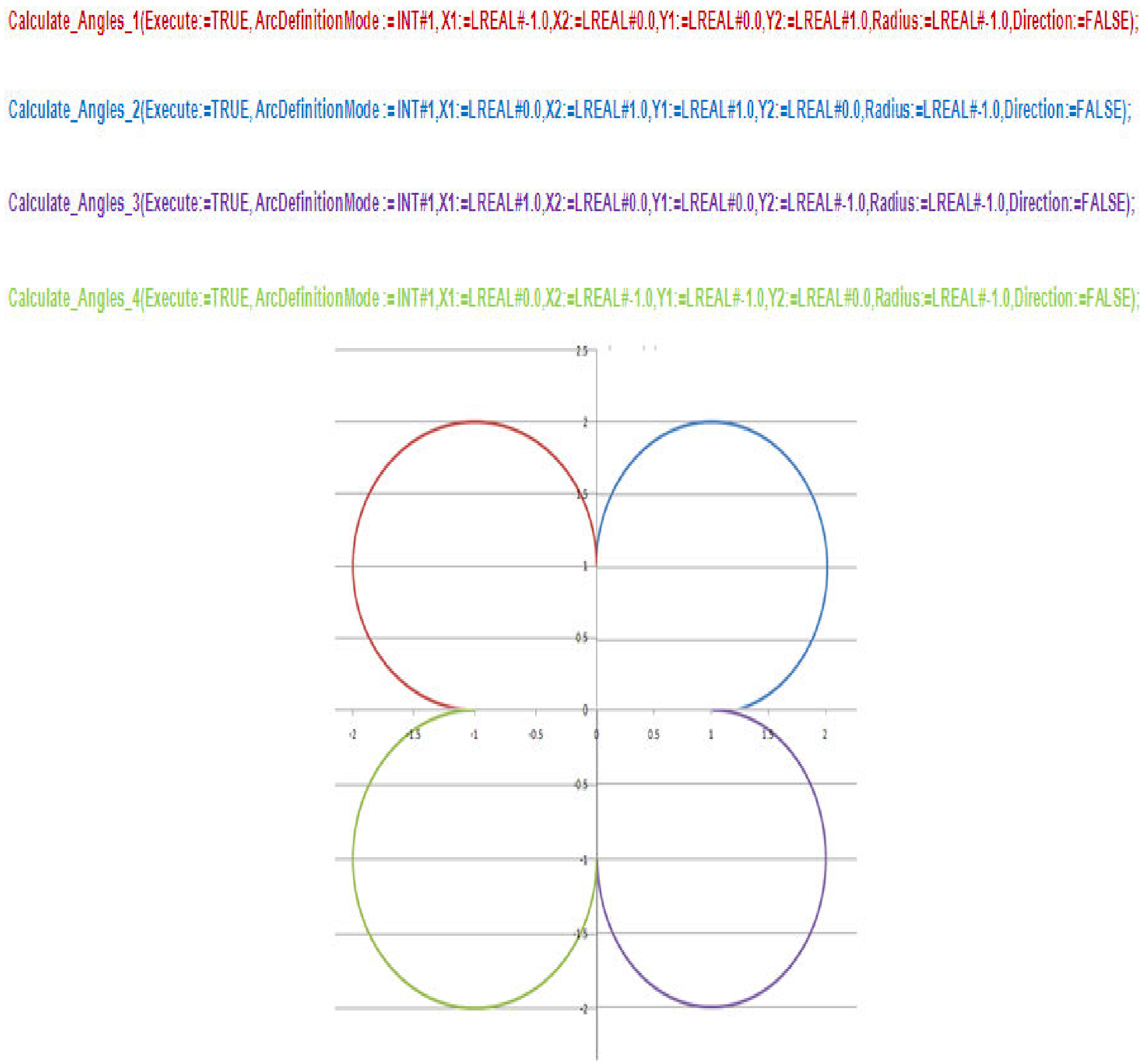


```
          (* Circular path*)
          (*===============*)
        2 CircularPath.Data[1].SegmentType:=TB_PatternType#Arc;
   0.5000 CircularPath.Data[1].Radius:=LREAL#0.5;
 180.0000 CircularPath.Data[1].StartAngle:=LREAL#180.0;
-360.0000 CircularPath.Data[1].TraversedAngle:=LREAL#-360.0;
   0.0500 CircularPath.Data[1].Resolution:=REAL#0.05;

        1 CircularPath.Segments := INT#1;
```



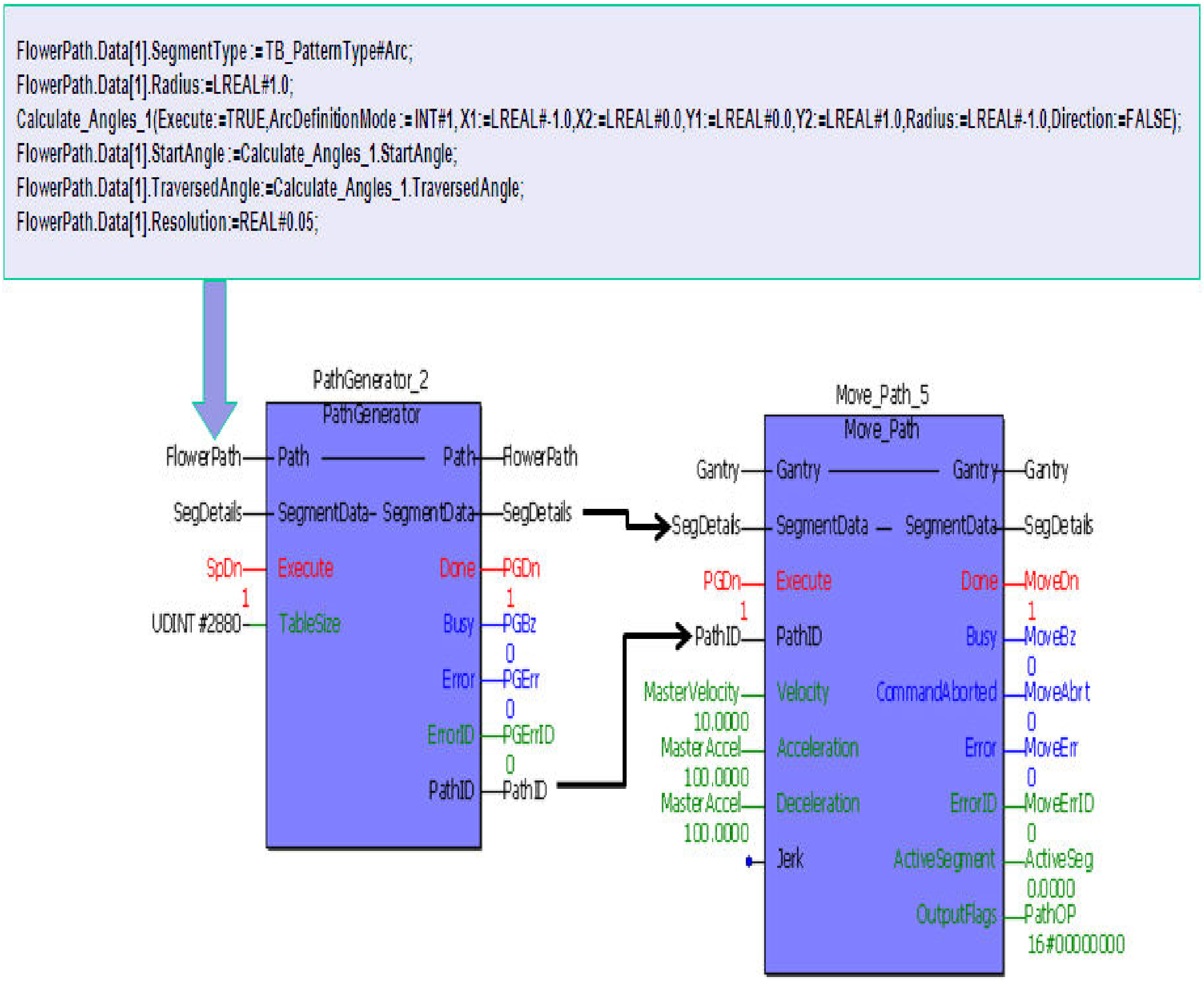
Application Example

Step1: Using Calculate_Angles FB to calculate the Start and Traverse angles for the flower path shown below.



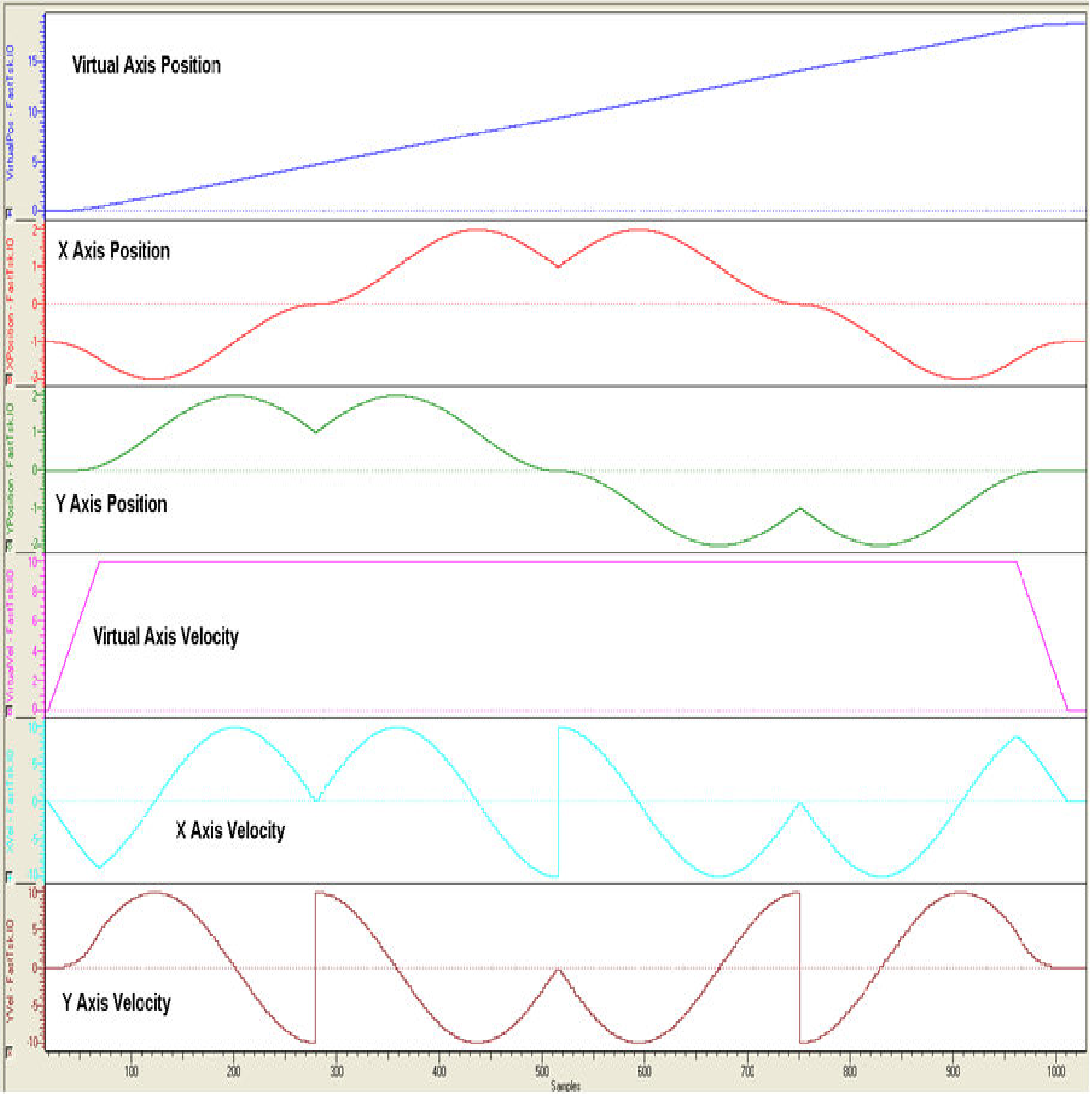
```
Calculate_Angles_1(Execute:=TRUE, ArcDefinitionMode := INT#1, X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_2(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#0.0,X2:=LREAL#1.0,Y1:=LREAL#1.0,Y2:=LREAL#0.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_3(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#-1.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_4(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#0.0,X2:=LREAL#-1.0,Y1:=LREAL#-1.0,Y2:=LREAL#0.0,Radius:=LREAL#-1.0,Direction:=FALSE);
```

Step 2: Use the PathGenerator FB to create the path and the Move_Path FB to implement XY motion.

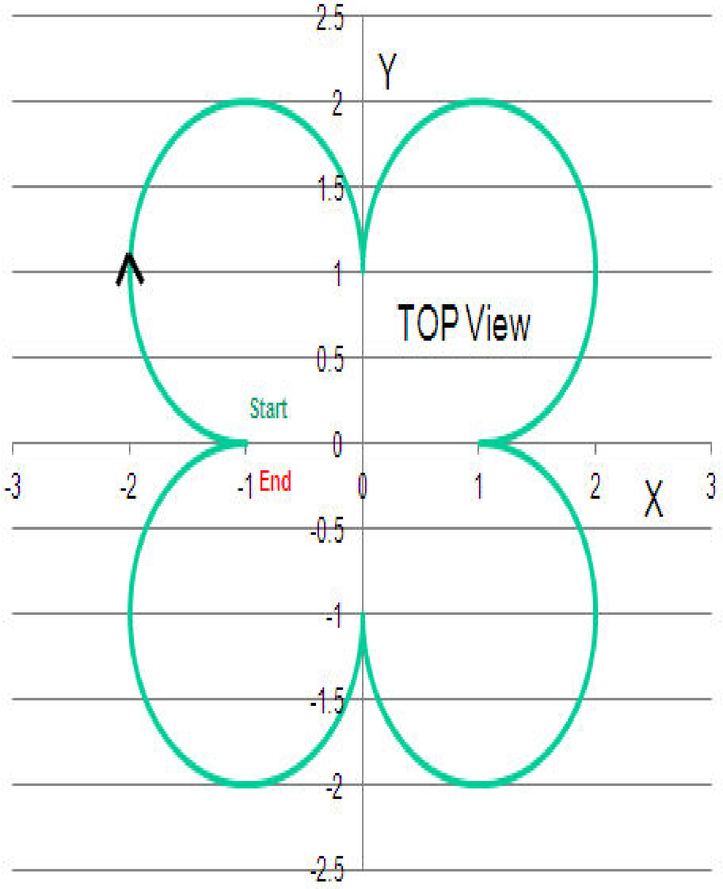


```
FlowerPath.Data[1].SegmentType :=TB_PatternType#Arc;
FlowerPath.Data[1].Radius:=LREAL#1.0;
Calculate_Angles_1(Execute:=TRUE,ArcDefinitionMode := INT#1, X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);
FlowerPath.Data[1].StartAngle :=Calculate_Angles_1.StartAngle;
FlowerPath.Data[1].TraversedAngle:=Calculate_Angles_1.TraversedAngle;
FlowerPath.Data[1].Resolution:=REAL#0.05;
```

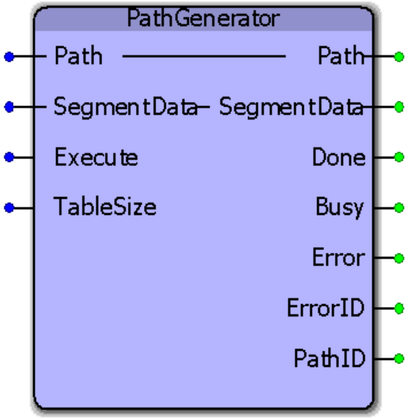
Step 3: Validation using logic analyzer.



Step 4: Result on XY system.


Y
TOP View
Start
End
X
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-2
-1
0
1
2
3

PathGenerator



This function block pre processes path data to provide coordinated motion using the Move_Path function block. Support for X, XPrime, Y, Z, and a Tangent axis are provided.

Library

Gantry Toolbox

Parameters

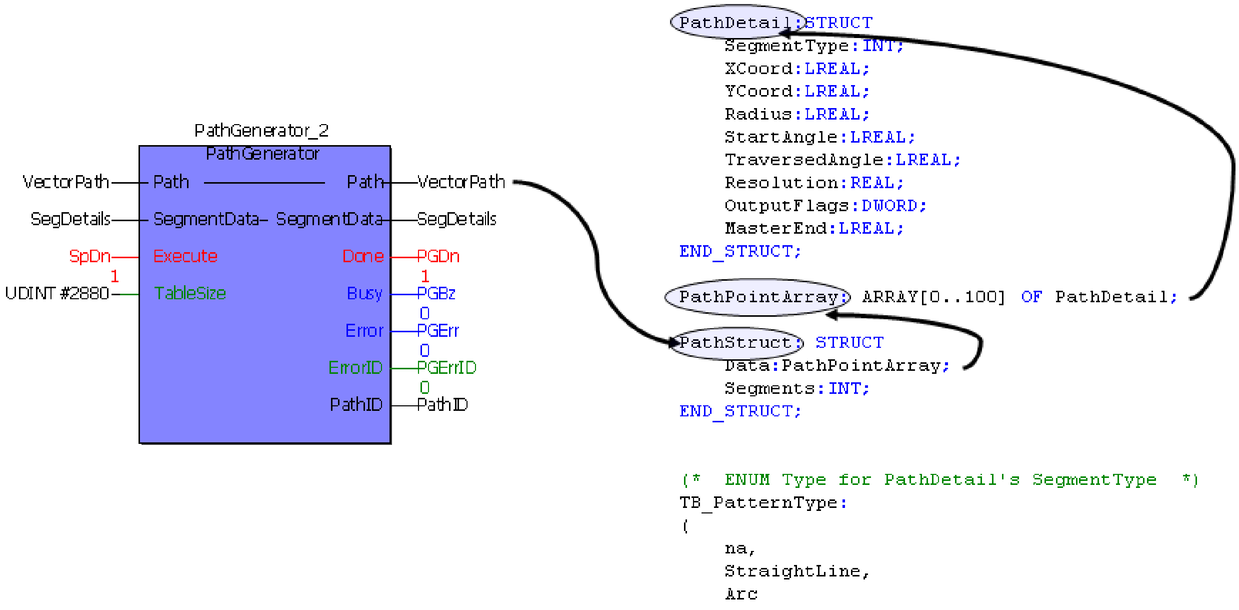
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Path	PathStruct	Structure of data that describes a motion path containing straight line and arc segments, dwell, input and output instructions.	
V	SegmentData	SegmentStruct	Structure of data that contains the segment number, output code, and tool path endpoint for each segment in the motion path.	
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 value must be the same as the definition of the ARRAY size of the MS_Array_Type in the MotionInfo DataTypes folder of either the PLCopen or DataTypes Toolbox.	UDINT#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.	

B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
V	PathID	PathIDStruct	For use by the Move_Path and PathIDManager function block.

Notes

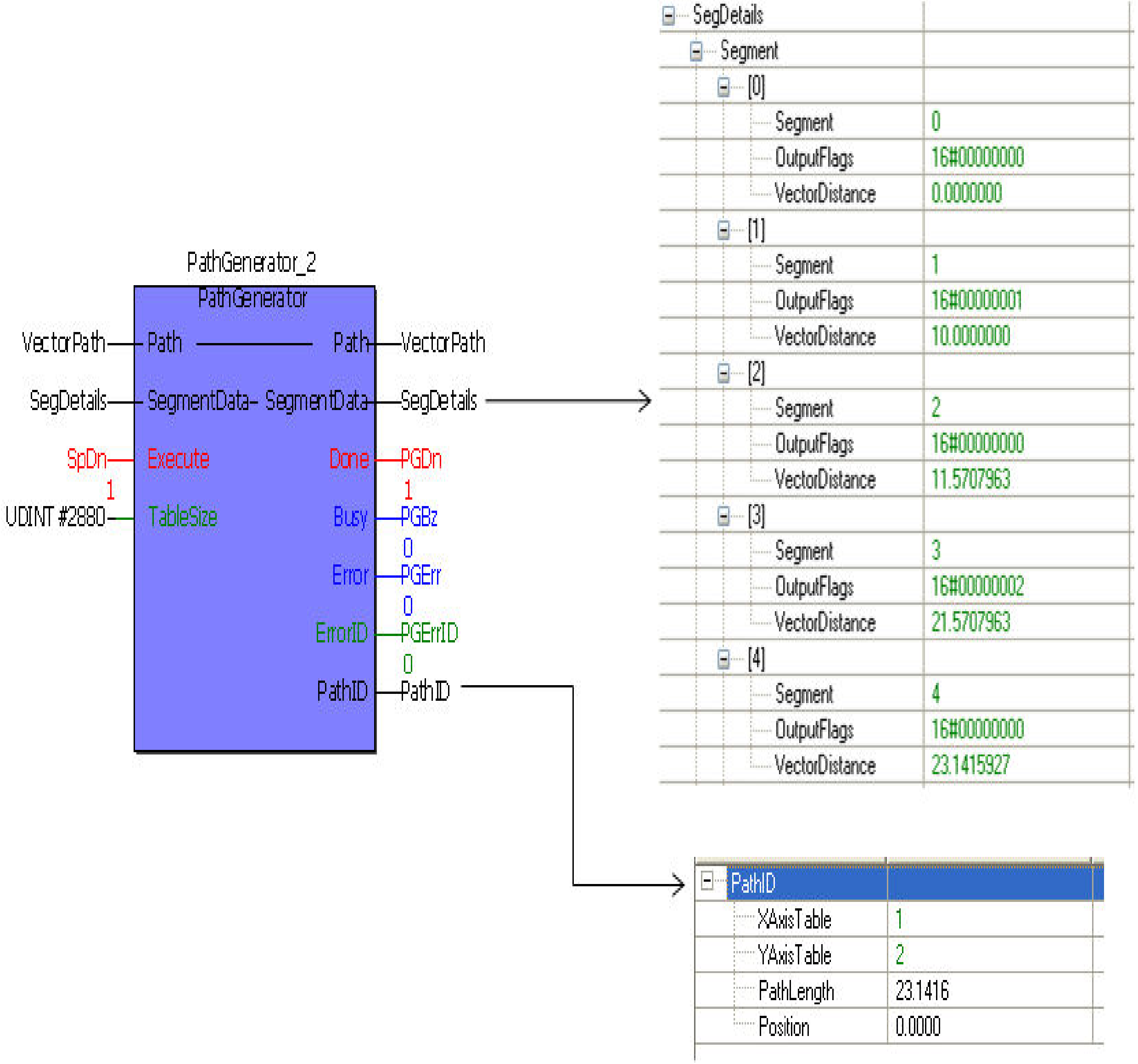
- The entire path must be specified and processed by the PathGenerator before motion can occur using the MovePath function block.
- If the PathGenerator will be executed multiple times because the application requires the ability to make a variable number of paths, you must use the PathIDManager function block to clear up the old cam memory in the motion engine layer.
- Do not use PathStruct element [0], start the path specification at element[1].

The inputs to the PathGenerator are shown below:



```
PathDetail:STRUCT
    SegmentType:INT;
    XCoord:LREAL;
    YCoord:LREAL;
    Radius:LREAL;
    StartAngle:LREAL;
    TraversedAngle:LREAL;
    Resolution:REAL;
    OutputFlags:DWORD;
    MasterEnd:LREAL;
END_STRUCT;

PathPointArray: ARRAY[0..100] OF PathDetail;

PathStruct: STRUCT
    Data:PathPointArray;
    Segments:INT;
END_STRUCT;


(*  ENUM Type for PathDetail's SegmentType  *)
TB_PatternType:
(
    na,
    StraightLine,
    Arc
```

The outputs from the PathGenerator are shown below:



SegDetails	
Segment	
[0]	
Segment	0
OutputFlags	16#00000000
VectorDistance	0.0000000
[1]	
Segment	1
OutputFlags	16#00000001
VectorDistance	10.0000000
[2]	
Segment	2
OutputFlags	16#00000000
VectorDistance	11.5707963
[3]	
Segment	3
OutputFlags	16#00000002
VectorDistance	21.5707963
[4]	
Segment	4
OutputFlags	16#00000000
VectorDistance	23.1415927

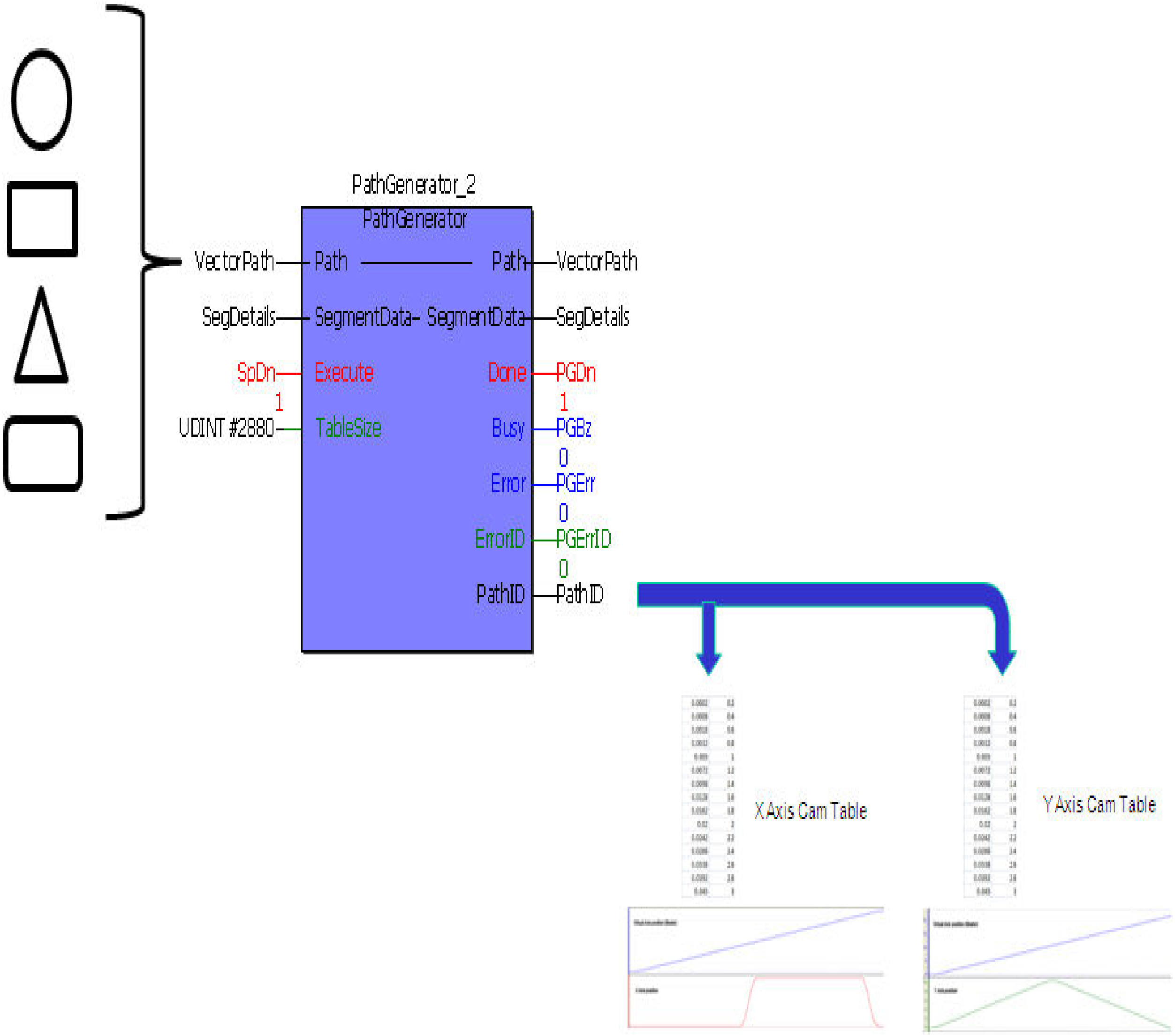
PathID	
XAxisTable	1
YAxisTable	2
PathLength	23.1416
Position	0.0000

- See Yaskawa's Youtube channel for more info, details, and examples.

Error Description

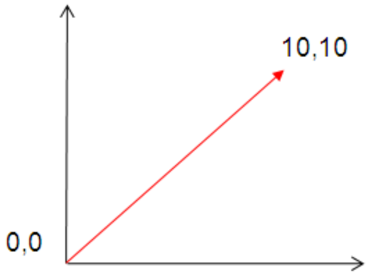
ErrorID	Meaning
0	No error.
10038	CamData.LastSegment must be greater than 0 and less than 400, or whatever value has been declared as the ARRAY size in the CTB_Types file.
10053	DataPoint Error.
10054	One of the segments in the path has an invalid Segment Type. Valid Segments Types are defined in Group Toolbox GroupTypes file as enumeration GTB_SegmentType.
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.

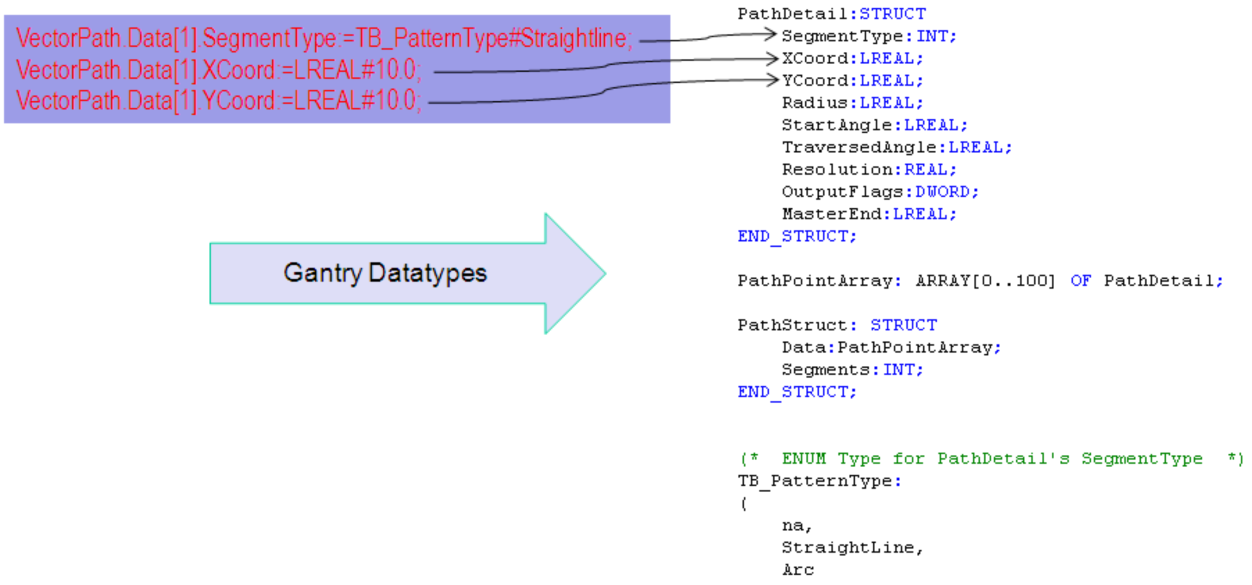
Usage Example



PathStruct Example 1

Straight Line Path Example

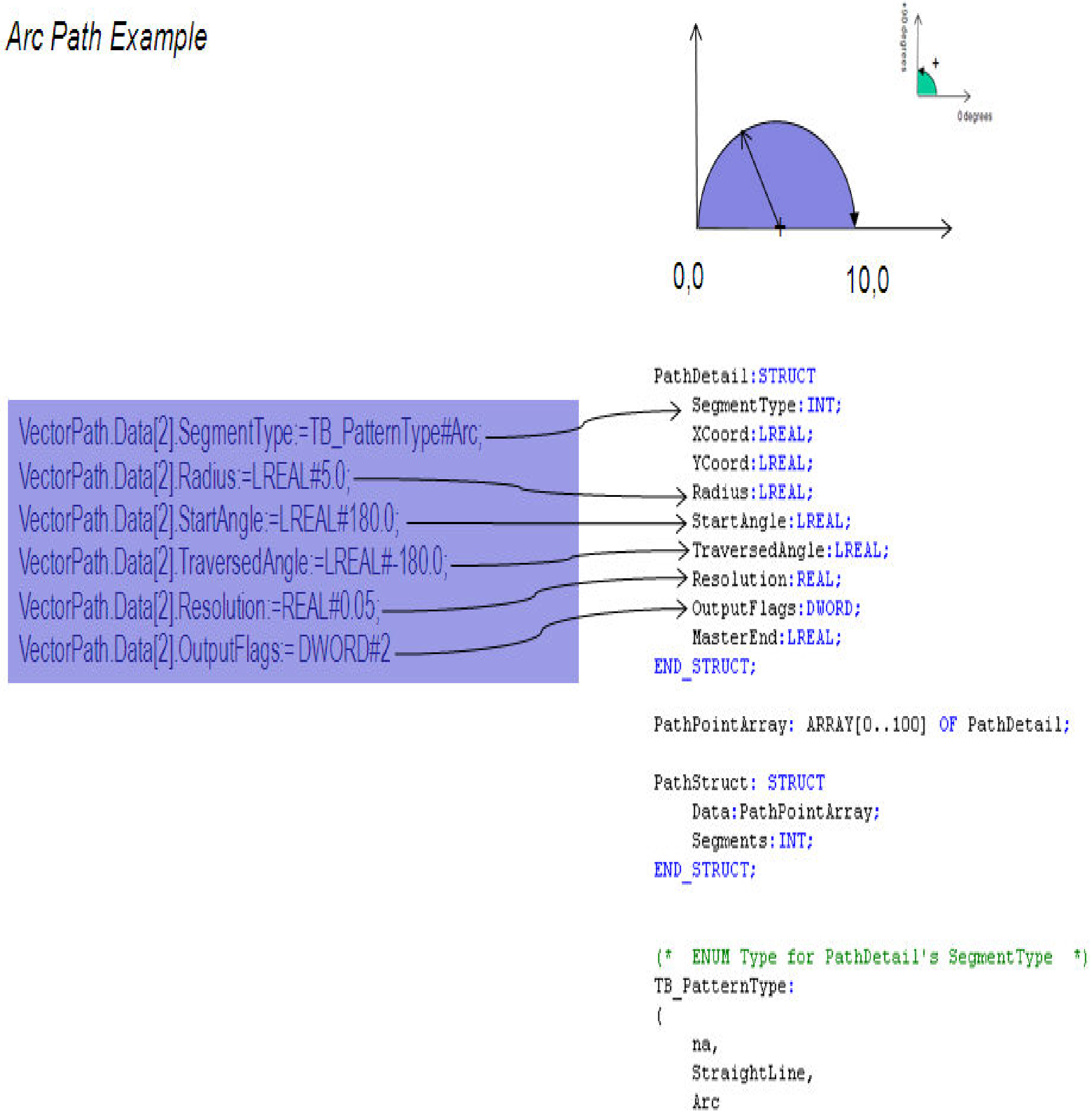




```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#10.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
```

```
PathDetail:STRUCT
    SegmentType:INT;
    XCoord:LREAL;
    YCoord:LREAL;
    Radius:LREAL;
    StartAngle:LREAL;
    TraversedAngle:LREAL;
    Resolution:REAL;
    OutputFlags:DWORD;
    MasterEnd:LREAL;
END_STRUCT;

PathPointArray: ARRAY[0..100] OF PathDetail;

PathStruct: STRUCT
    Data:PathPointArray;
    Segments:INT;
END_STRUCT;


(*  ENUM Type for PathDetail's SegmentType  *)
TB_PatternType:
(
    na,
    StraightLine,
    Arc
```

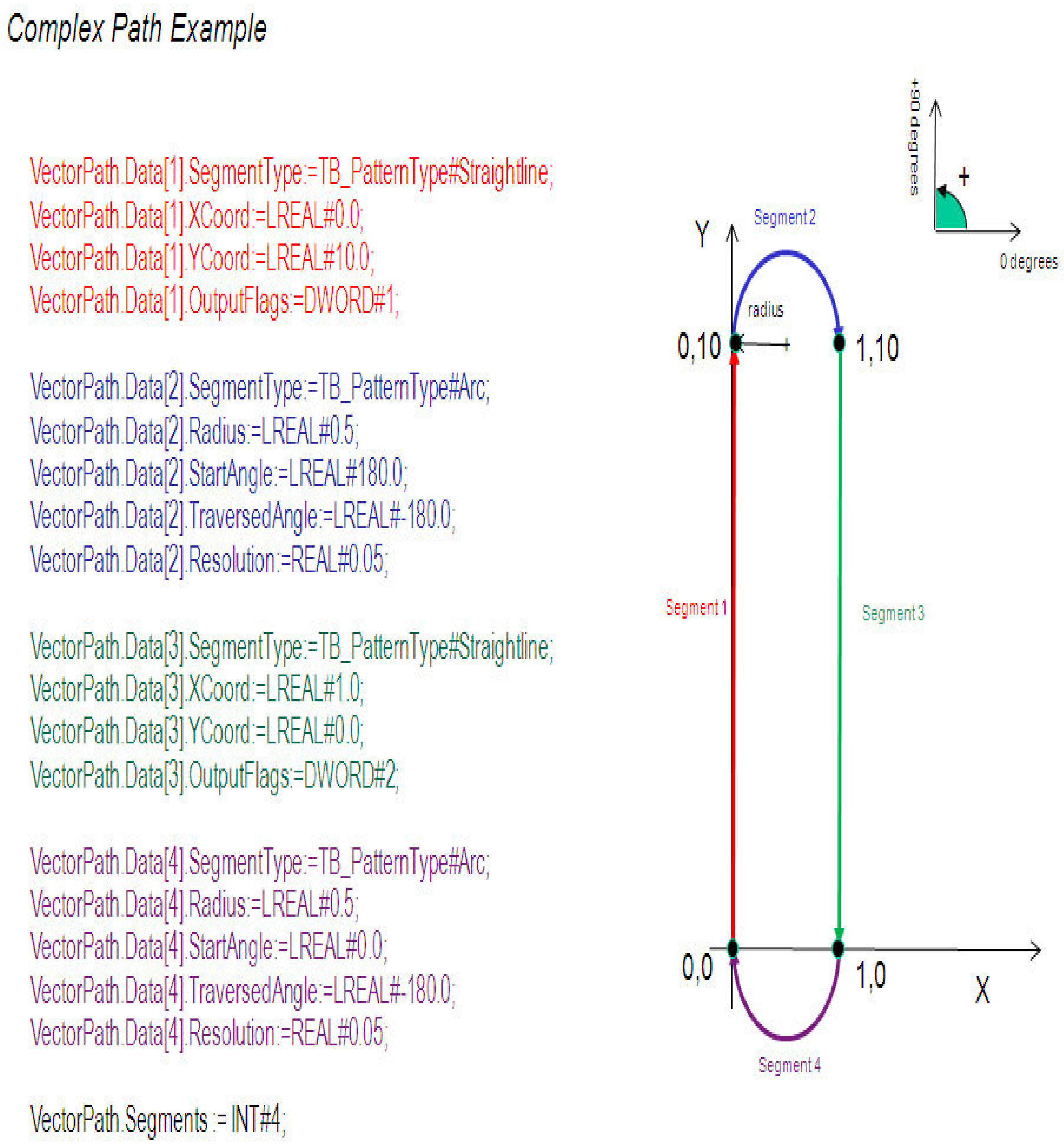
PathStruct Example 2



```
VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[2].Radius:=LREAL#5.0;
VectorPath.Data[2].StartAngle:=LREAL#180.0;
VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[2].Resolution:=REAL#0.05;
VectorPath.Data[2].OutputFlags:= DWORD#2
```

```
PathDetail:STRUCT
    SegmentType:INT;
    XCoord:LREAL;
    YCoord:LREAL;
    Radius:LREAL;
    StartAngle:LREAL;
    TraversedAngle:LREAL;
    Resolution:REAL;
    OutputFlags:DWORD;
    MasterEnd:LREAL;
END_STRUCT;

PathPointArray: ARRAY[0..100] OF PathDetail;

PathStruct: STRUCT
    Data:PathPointArray;
    Segments:INT;
END_STRUCT;


(*  ENUM Type for PathDetail's SegmentType  *)
TB_PatternType:
(
    na,
    StraightLine,
    Arc
```

PathStruct Example 3

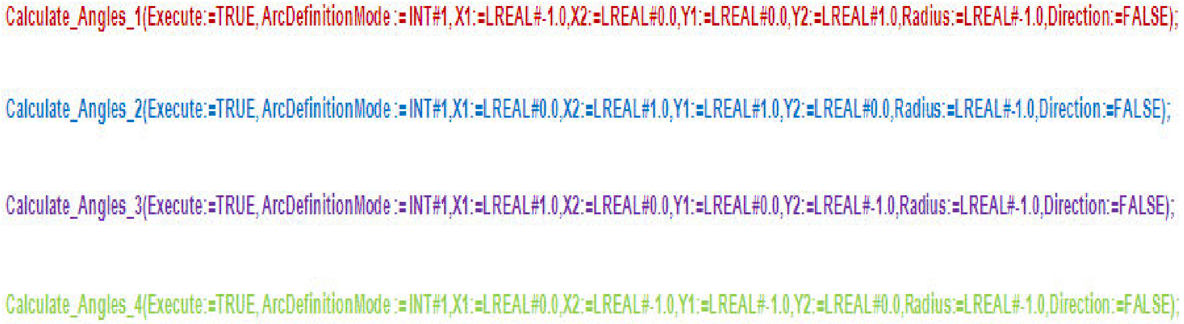


Complex Path Example

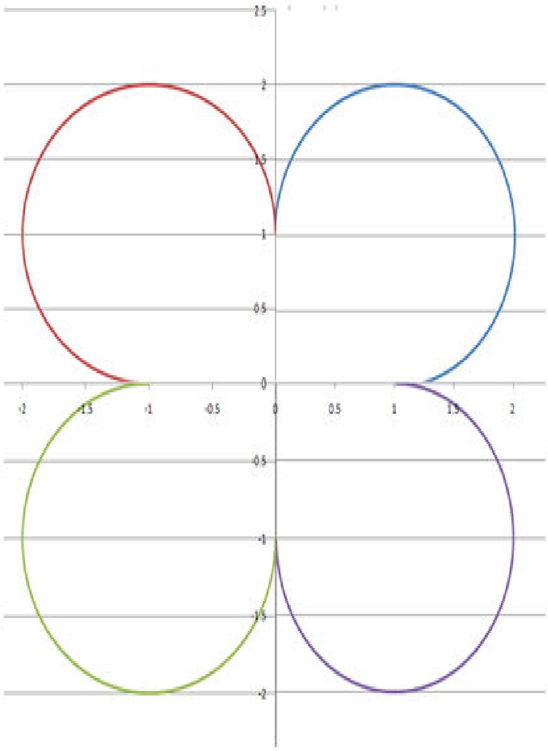
```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#0.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
VectorPath.Data[1].OutputFlags:=DWORD#1;

VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[2].Radius:=LREAL#0.5;
VectorPath.Data[2].StartAngle:=LREAL#180.0;
VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[2].Resolution:=REAL#0.05;

VectorPath.Data[3].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[3].XCoord:=LREAL#1.0;
VectorPath.Data[3].YCoord:=LREAL#0.0;
VectorPath.Data[3].OutputFlags:=DWORD#2;

VectorPath.Data[4].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[4].Radius:=LREAL#0.5;
VectorPath.Data[4].StartAngle:=LREAL#0.0;
VectorPath.Data[4].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[4].Resolution:=REAL#0.05;

VectorPath.Segments := INT#4;
```

Application example

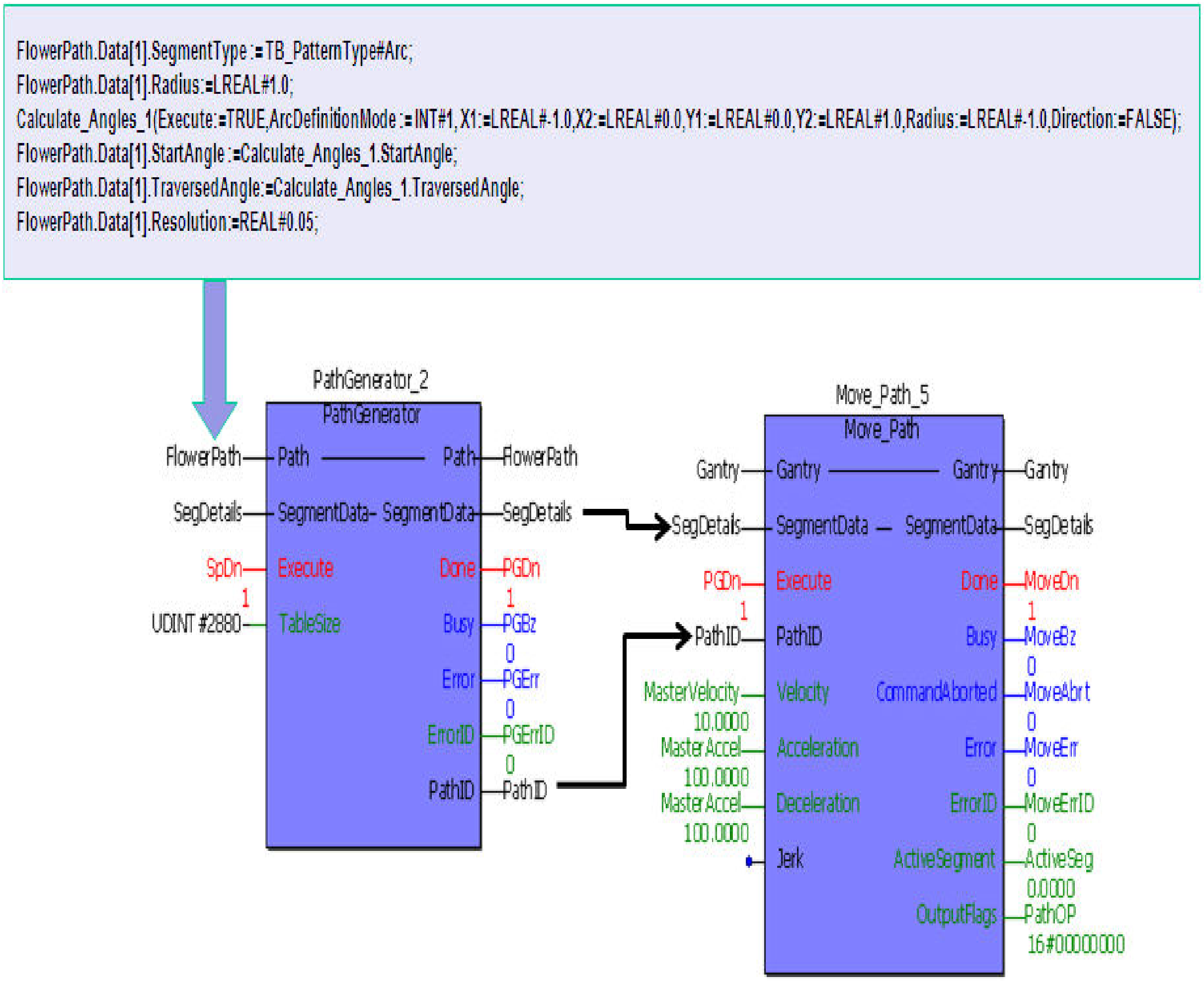
Step1: Using Calculate_Angles to calculate start and traverse angles for the flower path shown below



```
Calculate_Angles_1(Execute:=TRUE, ArcDefinitionMode := INT#1, X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_2(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#0.0,X2:=LREAL#1.0,Y1:=LREAL#1.0,Y2:=LREAL#0.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_3(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#-1.0,Radius:=LREAL#-1.0,Direction:=FALSE);

Calculate_Angles_4(Execute:=TRUE, ArcDefinitionMode := INT#1,X1:=LREAL#0.0,X2:=LREAL#-1.0,Y1:=LREAL#-1.0,Y2:=LREAL#0.0,Radius:=LREAL#-1.0,Direction:=FALSE);
```

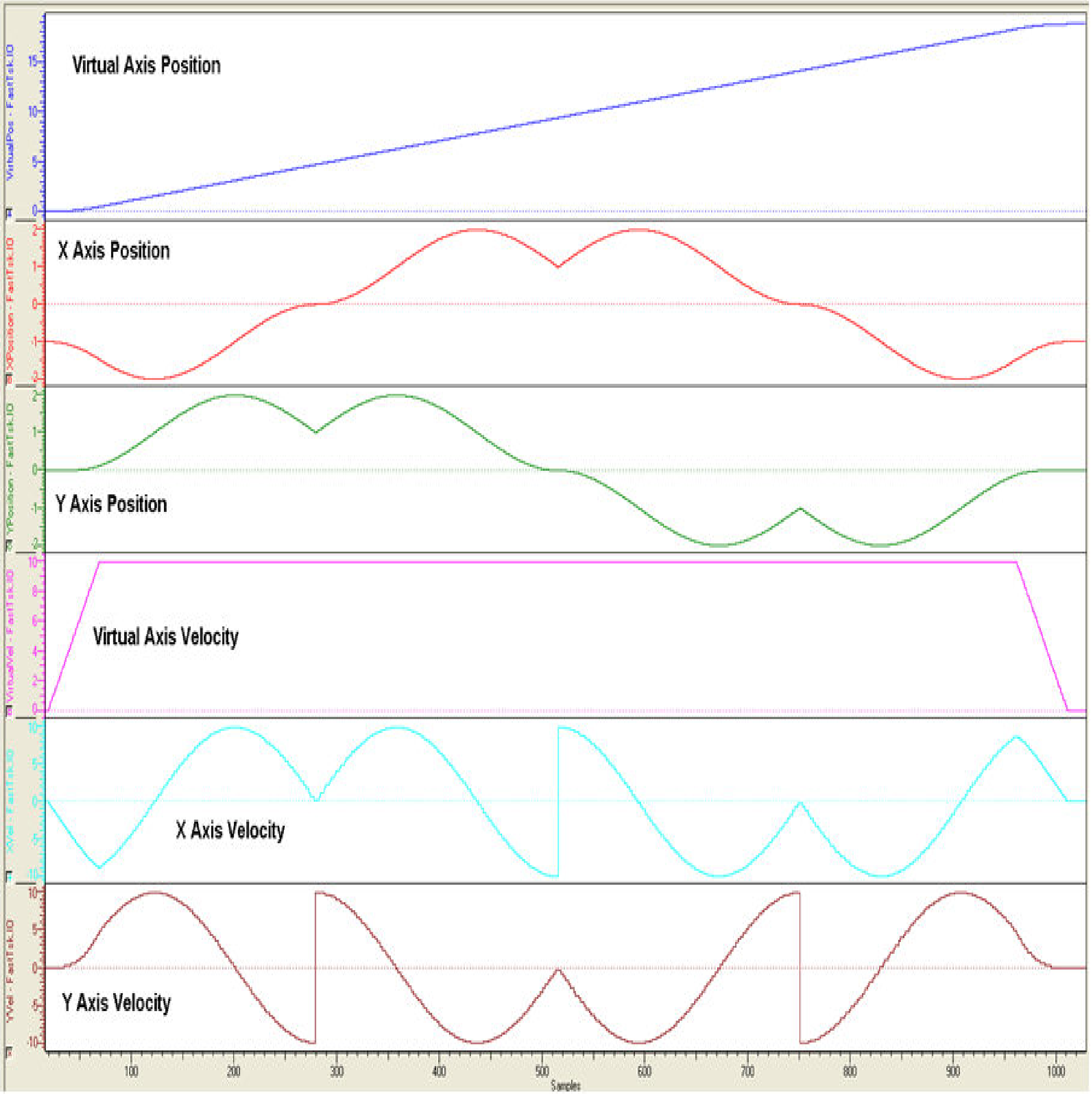


Step 2: Use PathGenerator create the path and Move_Path to implement XY motion



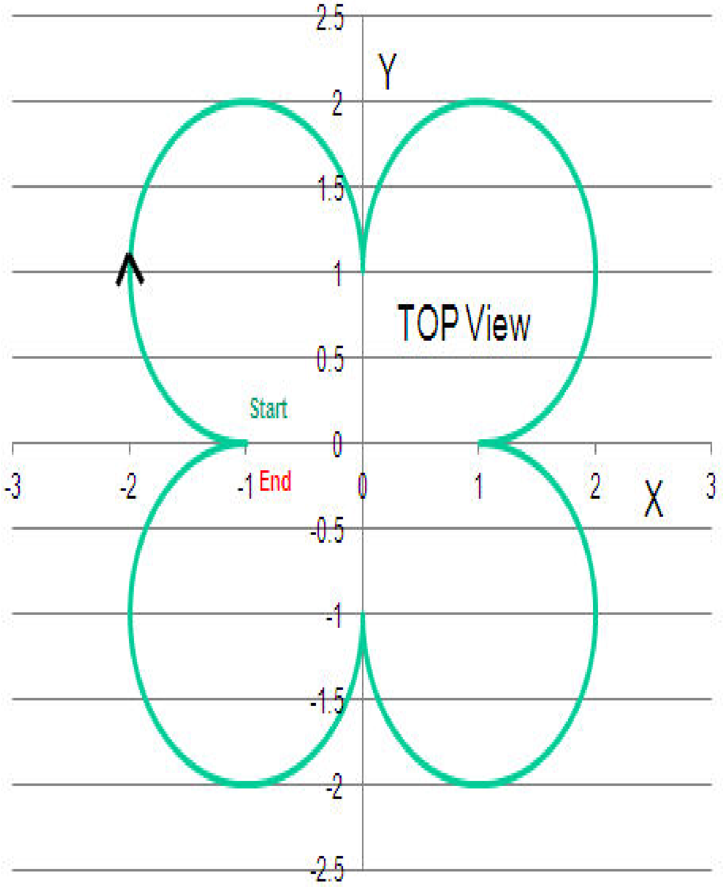
```
FlowerPath.Data[1].SegmentType:=TB_PatternType#Arc;
FlowerPath.Data[1].Radius:=LREAL#1.0;
Calculate_Angles_1(Execute:=TRUE,ArcDefinitionMode:=INT#1,X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);
FlowerPath.Data[1].StartAngle:=Calculate_Angles_1.StartAngle;
FlowerPath.Data[1].TraversedAngle:=Calculate_Angles_1.TraversedAngle;
FlowerPath.Data[1].Resolution:=REAL#0.05;
```

Step 3: Validation using logic analyzer



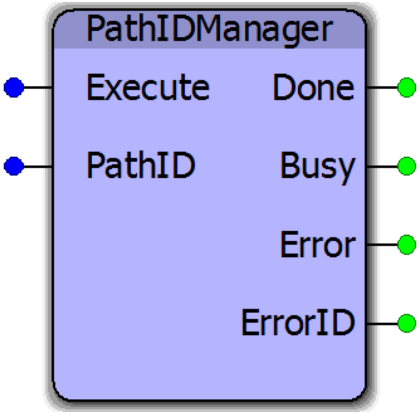
Step 4: Result on XY system

motioncontrols.com



2.5
Y
2
1.5
1
TOP View
0.5
Start
0
-3
-2
-1
End
0
1
2
X
3
-0.5
-1
-1.5
-2
-2.5

PathIDManager



This function block cleans up memory for applications which require many new paths during normal operation. This function block incorporates a FIFO buffer for PathIDs. It will delete memory used by the motion engine allocated to the oldest PathID by executing the Y_RemoveCamTable function block from the PLCopenPlus firmware library. A circular buffer of four PathIDs is maintained by the PathIDManager. When this function block is executed a fifth time, it releases the memory area of the oldest PathID. This function block does not affect the memory used in the IEC application layer, only the memory used at the motion engine layer for controller motion.

Library

Gantry 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
V	PathID	PathIDStruct	The most recent PathID created by the PathGenerator function block.	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	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

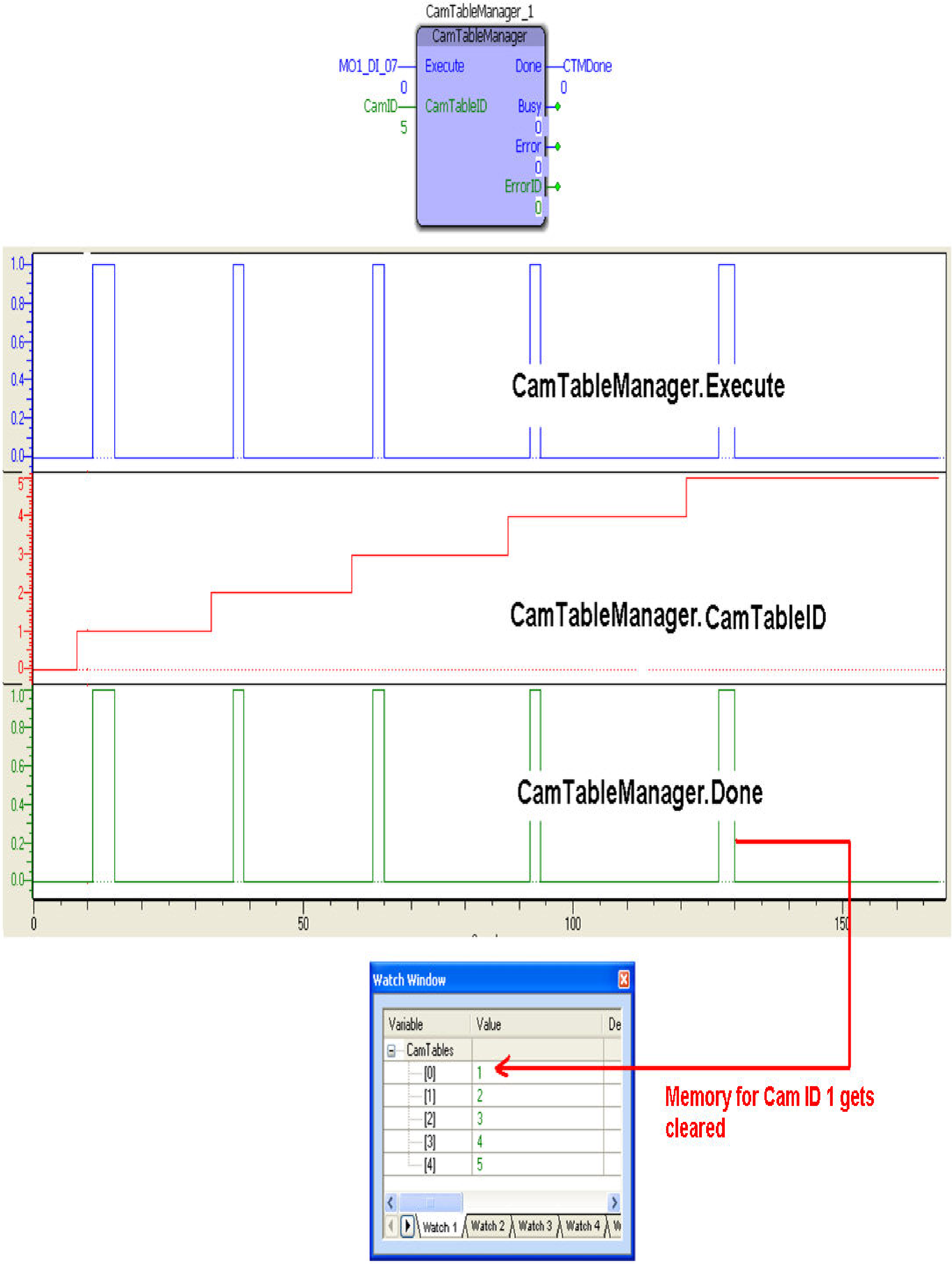
- Add the PathIDManager to the Done output of the PathGenerator. Feed it the PathID output from the PathGenerator. PathIDManager keeps a circular buffer of 4 PathIDs. When a PathID becomes the oldest in the buffer, the path is removed from the motion engine memory. This is the memory used by the controller to process motion, not the memory used the path data in the IEC application task.
- Even though the memory for cam tables has been released, the PathID values will continue to increase.
- If the PathGenerator is executed numerous times (dozens or hundreds) without cleaning up the motion engine memory, a "Memory Exhausted" controller alarm will result.
- This function block is unnecessary in applications which use a single, static PathID.

Error Description

ErrorID	Meaning
0	No error.
4887	CamTableID does not refer to a valid cam table.

Example 1

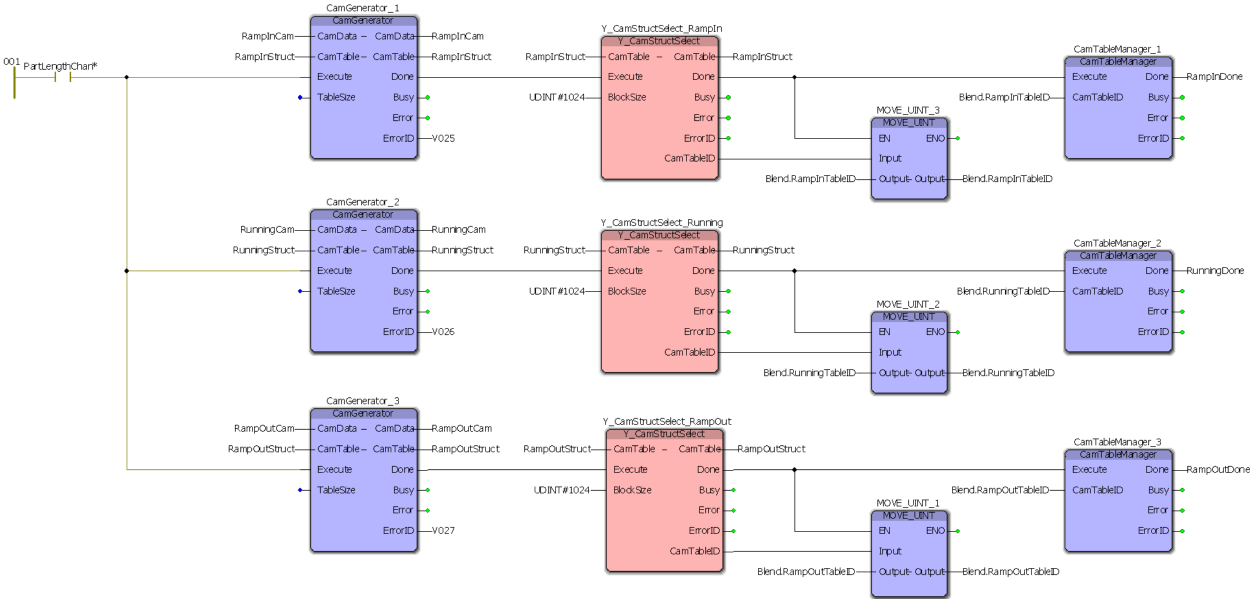
An example of using the CamTableManager is shown below; it operates very similarly to the PathIDManager function block. On the fifth execute of the PathIDManager block, the memory for the oldest Path ID gets released. In the example shown below, the memory for PathID 1 gets released. The next execution of the PathIDManager will release the memory for PathID 2.



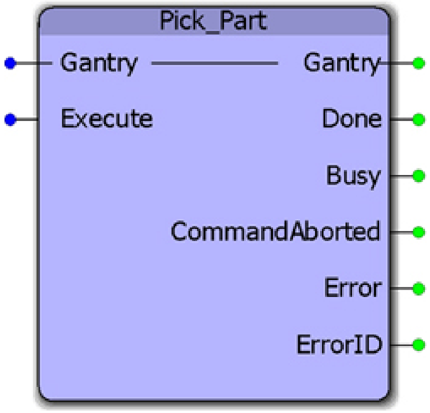
Variable	Value	De
CamTables		
[0]	1	
[1]	2	
[2]	3	
[3]	4	
[4]	5	

Memory for Cam ID 1 gets cleared

Application Example



Pick_Part



Assuming that a gripper actuator is empty and available to pick up a part in its mechanism, this function block initiates a series of actions that involves moving the XY axes to a specific location, opening a gripper actuator, moving the Z axis to a "Down" location, closing the gripper (to pick a part), and then finally moving the Z axis back to its "Up" position.

Library

Gantry Toolbox

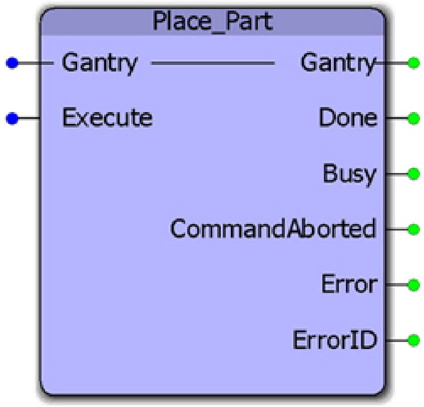
Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
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.	
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

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10034	Interpolation calculation error.
10035	Gripper Close Error (Timeout).
10036	Gripper Open Error (Timeout).
57617	Instance object is NULL.
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.

Place_Part



Given that a gripper actuator already has a part in its mechanism, this function block initiates a series of actions that involves moving the XY axes to a specific location, moving the Z axis to a "Down" location, opening the gripper (to place the part), and then finally moving the Z axis back to its "Up" position.

Parameters

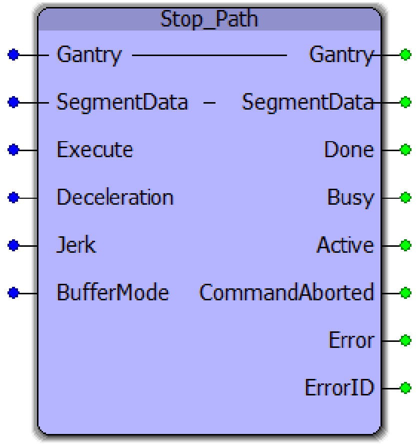
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and 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

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10034	Interpolation calculation error.
10035	Gripper Close Error (Timeout).
10036	Gripper Open Error (Timeout).
57617	Instance object is NULL.
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.

Stop_Path



This function block stops motion executed by the Move_Path function block.

Library

Gantry Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Structure containing information about all the axes in the Gantry system.	
V	SegmentData	SegmentStruct	Structure containing data about the path motion	
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	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.*	
B	BufferMode	MC_BufferMode	Defines the behavior of the axis - allowable modes are Aborting, Buffered, BlendingLow, BlendingPrevious, BlendingNext, and BlendingHigh.	MC_BufferMode#Aborting
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.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Notes

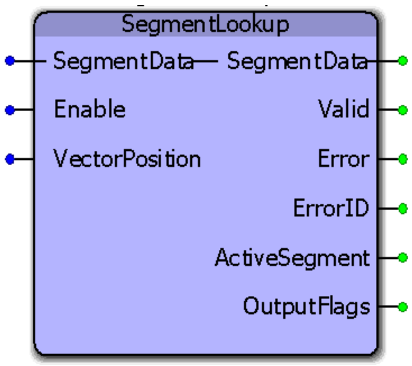
- This function block applies MC_Stop to the virtual axis of the Gantry specified.

Error Description

ErrorID	Meaning
0	No error.
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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4660	Deceleration is less than or equal to zero.
4983	The specified external axis may not be used. A physical axis is required.
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.

Example

SegmentLookup



This function block outputs the number of the segment currently active and also outputs the flags for the active segment.

Library

Gantry Toolbox

Parameters

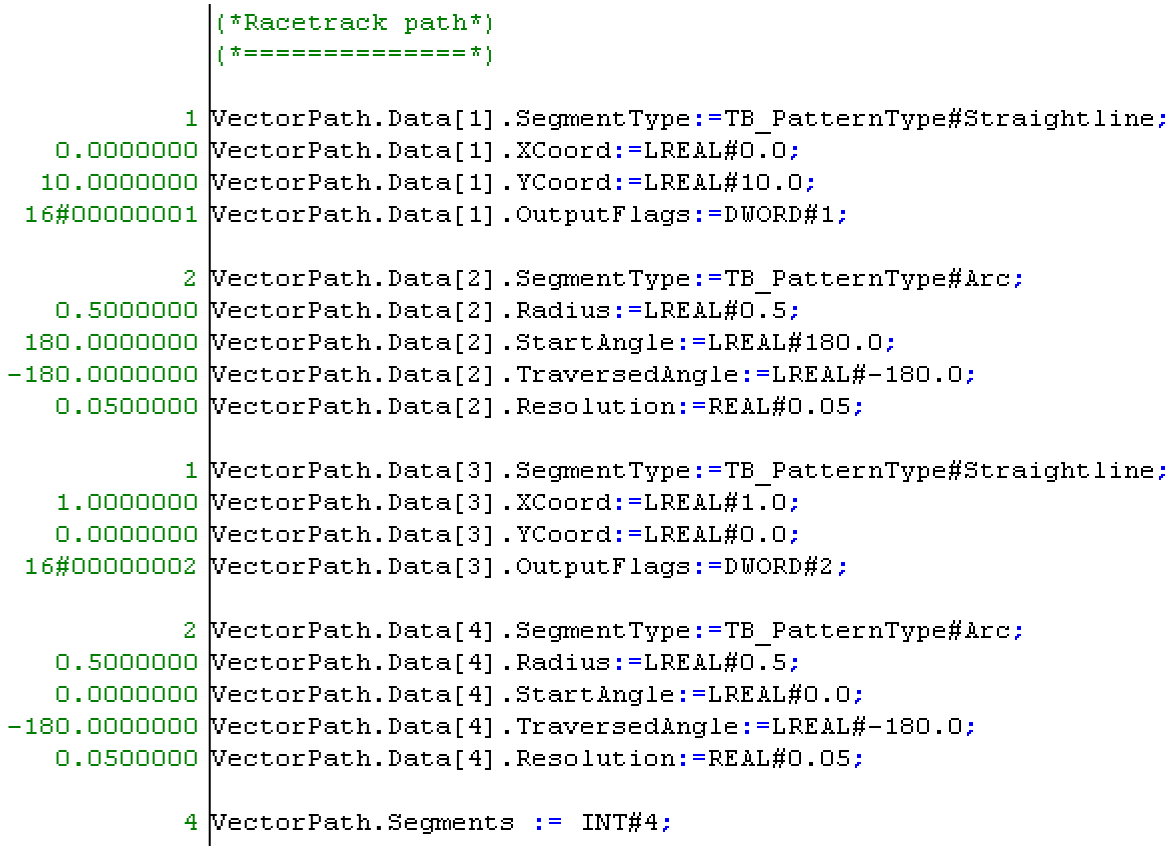
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Segmentdata	SegmentStruct	Structure of data that contains the segment number, output code, and tool path endpoint for each segment in the motion path.	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	VectorPosition	LREAL	Position of the vector along the path relative to the start of the Move_Path function block. This value can be obtained from the virtual axis of the Gantry.	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	ActiveSegment	INT	The segment that corresponds to the current position of the vector relative to the start of the Move_Path function block.	
V	OutputFlags	DWORD	Outputs as a DWORD that can be used to control digital output patterns unique for each segment.	

Error Description

ErrorID	Meaning
0	No error.
10140	Must be greater than zero and less than 20.

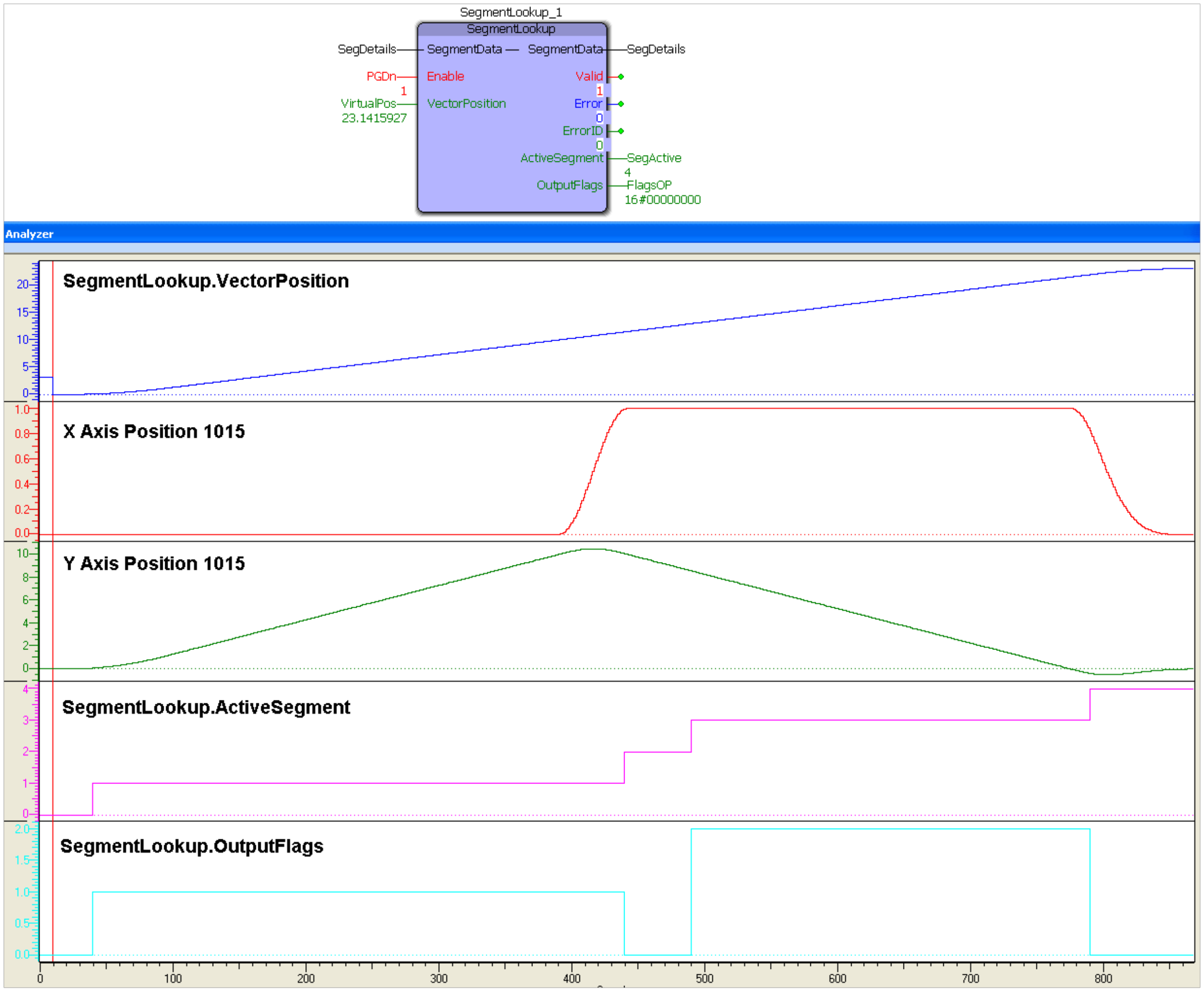
Example

Consider the profile shown below:

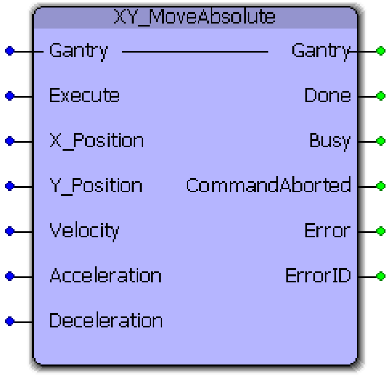


```
               (*Racetrack path*)
               (*===============*)

            1  VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
    0.0000000  VectorPath.Data[1].XCoord:=LREAL#0.0;
   10.0000000  VectorPath.Data[1].YCoord:=LREAL#10.0;
  16#00000001  VectorPath.Data[1].OutputFlags:=DWORD#1;

            2  VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
    0.5000000  VectorPath.Data[2].Radius:=LREAL#0.5;
  180.0000000  VectorPath.Data[2].StartAngle:=LREAL#180.0;
 -180.0000000  VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
    0.0500000  VectorPath.Data[2].Resolution:=REAL#0.05;

            1  VectorPath.Data[3].SegmentType:=TB_PatternType#Straightline;
    1.0000000  VectorPath.Data[3].XCoord:=LREAL#1.0;
    0.0000000  VectorPath.Data[3].YCoord:=LREAL#0.0;
  16#00000002  VectorPath.Data[3].OutputFlags:=DWORD#2;

            2  VectorPath.Data[4].SegmentType:=TB_PatternType#Arc;
    0.5000000  VectorPath.Data[4].Radius:=LREAL#0.5;
    0.0000000  VectorPath.Data[4].StartAngle:=LREAL#0.0;
 -180.0000000  VectorPath.Data[4].TraversedAngle:=LREAL#-180.0;
    0.0500000  VectorPath.Data[4].Resolution:=REAL#0.05;

            4  VectorPath.Segments := INT#4;
```

The output flags are set to DWORD#1 during segment 1 and set to DWORD#2 during segment 3. These can be seen in the logic analyzer plots from the SegmentLookup outputs.


SegmentLookup_1
SegmentLookup
SegDetails
SegmentData
SegmentData
SegDetails
PGDn
Enable
Valid
1
1
VirtualPos
23.1415927
VectorPosition
Error
0
ErrorID
0
ActiveSegment
SegActive
4
OutputFlags
FlagsOP
16#00000000
Analyzer
SegmentLookup.VectorPosition
20
15
10
5
0
X Axis Position 1015
1.0
0.8
0.6
0.4
0.2
0.0
Y Axis Position 1015
10
8
6
4
2
0
SegmentLookup.ActiveSegment
4
3
2
1
0
SegmentLookup.OutputFlags
2.0
1.5
1.0
0.5
0.0
0
100
200
300
400
500
600
700
800

XY_MoveAbsolute



This function block will perform an absolute move the X and Y axes to a specific location within the gantry coordinate system. The X and Y axes must be specified in GantryStruct before executing this function block. This block calculates the required acceleration, deceleration and velocity for each axis and then executes MC_MoveAbsolute function blocks simultaneously for each axis to create straight line motion at the tool point. This is not considered interpolated motion. If configured, no motion on the Z axis will occur.

Library

Gantry Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
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	X_Position	LREAL	Target X coordinate of the vector.	LREAL#0.0
B	Y_Position	LREAL	Target Y coordinate of the vector.	LREAL#0.0
B	Velocity	LREAL	Velocity of the vector.	LREAL#0.0
B	Acceleration	LREAL	Acceleration of the vector.	LREAL#0.0
B	Deceleration	LREAL	Deceleration of the vector.	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	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.	

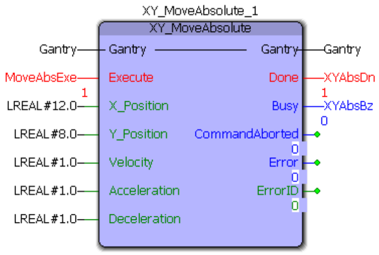
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

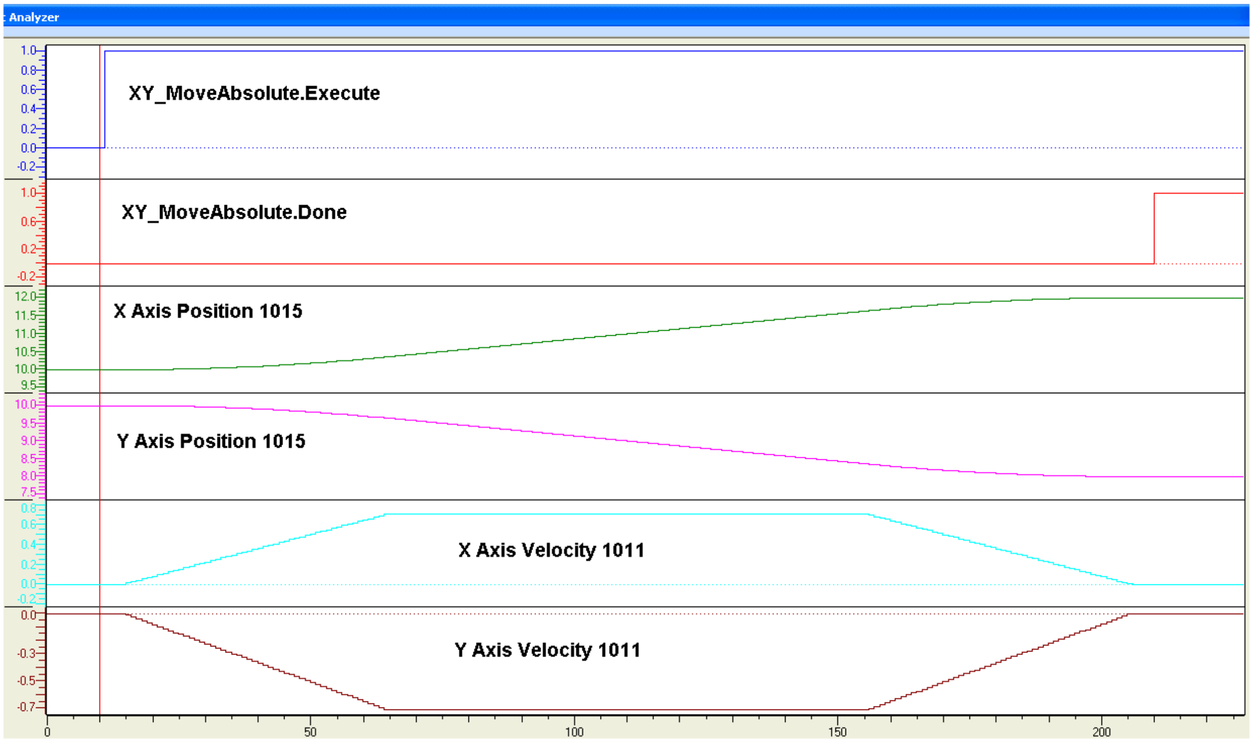
Error Description

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10034	Interpolation calculation 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.

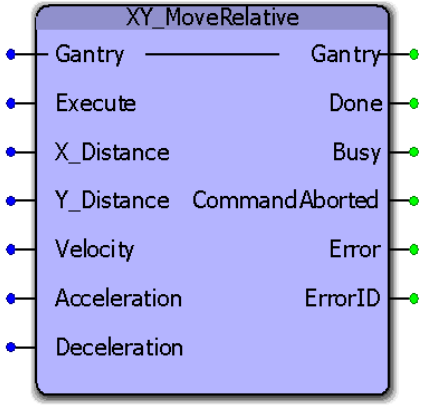
Example

In the example shown below, the XY gantry tooltip is at coordinate 10,10. The target coordinate is 12,8. On executing the XY_ MoveAbsolute function block, the X, Y axes move such that the tooltip's final position is 12, 8. The velocities, accelerations and decelerations of the two axes are calculated (in XY_MoveAbsolute) such that the individual axes start and stop at the same time instant.


XY_MoveAbsolute_1
XY_MoveAbsolute
Gantry
Gantry
Gantry
Gantry
MoveAbsExe
Execute
Done
XYAbsDn
1
1
LREAL#12.0
X_Position
Busy
XYAbsBz
0
LREAL#8.0
Y_Position
CommandAborted
0
LREAL#1.0
Velocity
Error
0
LREAL#1.0
Acceleration
ErrorID
0
LREAL#1.0
Deceleration


Analyzer
XY_MoveAbsolute.Execute
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
XY_MoveAbsolute.Done
1.0
0.6
0.2
-0.2
X Axis Position 1015
12.0
11.5
11.0
10.5
10.0
9.5
Y Axis Position 1015
10.0
9.5
9.0
8.5
8.0
7.5
X Axis Velocity 1011
0.8
0.6
0.4
0.2
0.0
-0.2
Y Axis Velocity 1011
0.0
-0.3
-0.5
-0.7
0
50
100
150
200

XY_MoveRelative



This function block will perform a relative move on the tooltip in a gantry coordinate system. The X and Y axes must be specified in GantryStruct before executing this function block. This block calculates the required acceleration, deceleration and velocity for each axis and then executes MC_MoveRelative function blocks simultaneously for each axis to create straight line motion at the tool point. This is not considered interpolated motion. If configured, no motion on the Z axis will occur.

Library

Gantry Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
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	X_Distance	LREAL	X coordinate distance to be moved	LREAL#0.0
V	Y_Distance	LREAL	Y coordinate distance to be moved	LREAL#0.0
B	Velocity	LREAL	Velocity of the tool tip	LREAL#0.0
B	Acceleration	LREAL	Acceleration of the tool tip	LREAL#0.0
B	Deceleration	LREAL	Deceleration of the tool tip	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	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This	

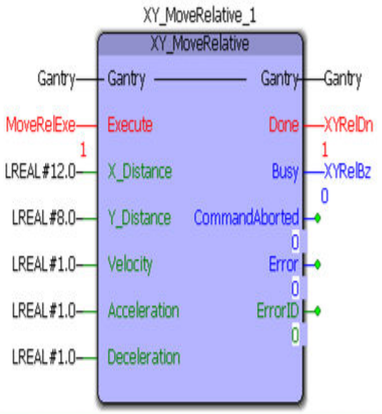
			output is cleared with the same behavior as the Done output.
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset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

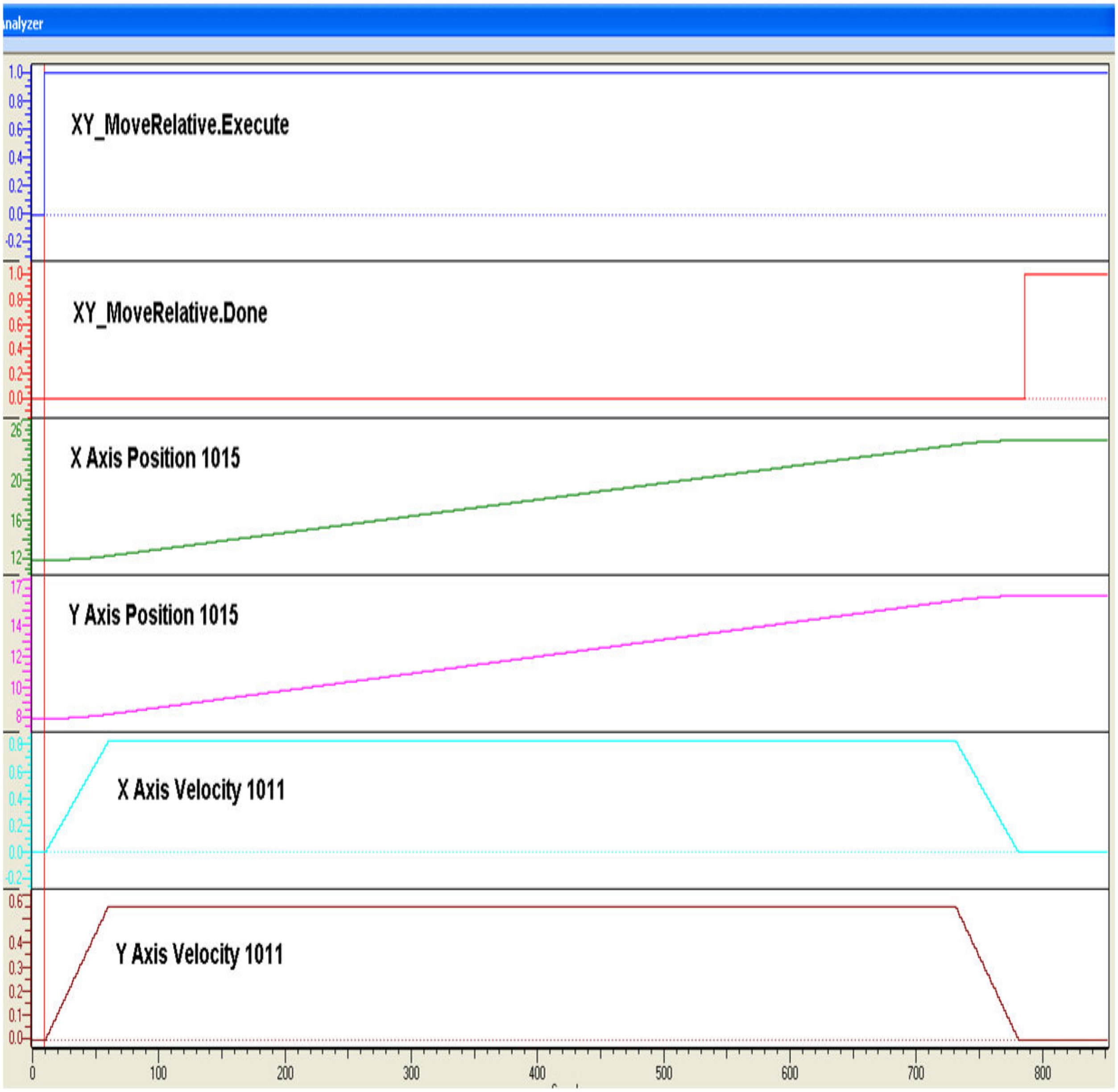
Error Description

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10034	Interpolation calculation 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.

Example

In the example shown below, the X Y coordinate of the tool tip is 12,8. On commanding an XY_MoveRelative move of 12, 8, the tool tip moves to coordinates 24, 16. The velocities, accelerations and decelerations of the two axes are calculated (in XY_MoveRelative) such that the individual axes start and stop at the same time instant.


XY_MoveRelative_1
XY_MoveRelative
Gantry
Gantry
Gantry
Gantry
MoveRelExe
Execute
Done
XYRelDn
1
1
LREAL#12.0
X_Distance
Busy
XYRelBz
0
LREAL#8.0
Y_Distance
CommandAborted
0
LREAL#1.0
Velocity
Error
0
LREAL#1.0
Acceleration
ErrorID
0
LREAL#1.0
Deceleration


nalyzer
XY_MoveRelative.Execute
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
XY_MoveRelative.Done
1.0
0.8
0.6
0.4
0.2
0.0
X Axis Position 1015
26
20
16
12
Y Axis Position 1015
17
14
12
10
8
X Axis Velocity 1011
0.9
0.6
0.4
0.2
0.0
-0.2
Y Axis Velocity 1011
0.6
0.4
0.3
0.2
0.1
0.0
0
100
200
300
400
500
600
700
800

Group_Toolbox

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Getting Started with Group Toolbox

The Group Toolbox is the successor to the Gantry Toolbox, which was created before the MPiec product officially supported interpolated motion for groups of axes. (PLCopen Part 4.) Group Toolbox is only supported on MP3000iec series controllers.

Differences from Gantry Toolbox

- Gantry Toolbox included functions such as GantryPower, GantryStop, and XYMoveAbsolute. These functions treated individual axes as a group at the IEC application layer, and did not provide true interpolated motion calculated by the firmware.
- For path functions via PathGenerator and MovePath, Gantry Toolbox converted the path motion required into cam tables and used a virtual master to drive the vector path. Group Toolbox uses firmware interpolation and group support to provide motion for gantries and other mechanisms such as T-Bot, H-Bot, Delta robots and MLX200 hosted robots.
- Gantry toolbox did not support gantries with dual motors operating one joint. Through firmware support for groups with more than one motor on a given axis, more complex gantries are supported using Group Toolbox techniques.
- Gantry Toolbox could only make paths of finite lengths which must be completely specified before motion starts. Group Toolbox can make infinite path lengths and start motion before all the data has been processed.

Similarities to Gantry Toolbox

- Both Toolboxes offer similar MovePath functions to build complex motion sequences, but Group toolbox takes advantage of the features of PLCopen Part 4, and provides support for blended paths, circles on any plane, and a potentially infinite path length.

Requirements for v330

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

Firmware libraries:

- YMotion
- YCoordinatedMotion
- If using the streaming method (Read_GCode_Stream), Y_DeviceComm

User libraries:

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

- DataTypes_Toolbox (v330 or higher)
- Math_Toolbox (v331 or higher. Note! This is one revision higher than the Math Toolbox released with MotionWorks IEC 3.3.0)
- Yaskawa Toolbox (v330 or higher)
- PLCopen_Toolbox (v330 or higher)
- PLCopenPart4 supporting library, such as Group_stub_v330 or MLX200_v214a. MotionWorks IEC version 3.3.0 and higher automatically manages this library when saving the Hardware Configuration.
- PLCopenPart4 (v330 or higher) MotionWorks IEC version 3.3.0 and higher automatically manages this library when saving the Hardware Configuration.
- If using the File method (Read_GCode_File), FileRW Toolbox (v302 or higher)
- If using the streaming method (Read_GCode_Stream), Comm Toolbox (v301 or higher)

Group Toolbox Revision History

Current Version:

(******************** 2017-01-07 v330 released ***********************)

This is the first release.

G Code Support

G Codes

Command	Description	Support	Comment
G00	Rapid positioning. (Max speed for each axis)	No	Requires firmware support for MC_MoveDirectAbsolute and MC_MoveDirectRelative. The G Code parser will interpret a G0 as a G1. Future support is planned.
G01	Linear interpolation	Yes	
G02	CCW circular interpolation	Yes	
G03	CW circular interpolation	Yes	
G04	Dwell	Yes	Time (in milliseconds) must be given in parameter P.
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.
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.
G08			
G09	Exact Stop, non-modal.	Yes	Modal version is G61.
G10	Programmable data input. Modifies the value of work coordinate and tool offsets.	Yes	
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	
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 will allow the G_Code parser to convert positions in the G Code file to the configured units of the machine.
G21	Programming in mm	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 will allow the G_Code parser to convert positions in the G Code file to the configured units of the machine.

G28	Return to the Home Position	No	Return to the machine's reference position, sometimes called the zero position. That zero return position is where most programs begin.
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	Turn off cutter radius compensation. Cancels G41 or G42.
G41	Tool radius compensation left	Yes	Creates a Left Tool Compensation along the XY axis. Tool Data can be found in the Tool structure.
G42	Tool radius compensation right	Yes	Creates a Right Tool Compensation along the XY axis. Tool Data can be found in the Tool structure.
G43	Tool height offset compensation negative	No	Uses the H register as the tool length offset. The value is negative because it will be added to the gauge line position.
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	No	
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 rather than program zero. Can be helpful for tool changes. Nonmodal and absolute only. Subsequent blocks are interpreted as "back to G54" even if it is not explicitly programmed.	No	
G54	Work Coordinate System 1	Yes	Set using G10, or write to MachineStruct.CoordSystem directly.
G55	Work Coordinate System 2	Yes	Set using G10, or write to MachineStruct.CoordSystem directly.
G56	Work Coordinate System 3	Yes	Set using G10, or write to MachineStruct.CoordSystem directly.
G57	Work Coordinate System 4	Yes	Set using G10, or write to MachineStruct.CoordSystem directly.
G58	Work Coordinate System 5	Yes	Set using G10, or write to MachineStruct.CoordSystem directly.
G59	Work Coordinate System 6	Yes	Set using G10, or write to MachineStruct.CoordSystem directly.
G61	Exact stop check, modal. Can be canceled with G64. Non-modal version is G09	No	
G62	Automatic Corner Override	No	
G64	Default cutting mode. Cancels G61	No	
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	No	

	cycle, for roughing (X-axis emphasis)		
G73	Fixed cycle, multiple repetitive cycle, for roughing, with pattern repetition	No	
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 external to the AxesGroup to the XY plane of a path. The external axis must be defined as shown in the example for MC_MovePath. Requires Software and firmware 3.3.0.
G79	Tangent Motion Disable	Yes	
G80	Simple Drilling Cycle	No	
G81	Drilling Cycle with Dwell	No	
G83	Peck Drilling cycle (full retraction from pecks)	No	
G84	Tapping cycle, righthand thread, M03 spindle direction	No	
G83	Peck Drilling cycle (full retraction from pecks)	No	
G84	Tapping cycle, righthand thread, M03 spindle direction	No	

M Codes

Command	Description	Support	Comment
M00	Non-optional Stop. Machine always stops here	No	
M01	Optional Stop. Only stops if Optional stop button has been pressed	No	
M02	End of Program	Yes	
M03	Spindle On (clockwise rotation)	Yes	Supported as OutputFlag.X3
M04	Spindle On (counterclockwise rotation)	Yes	Supported as OutputFlag.X4
M05	Spindle Stop	Yes	Supported as OutputFlag.X3 and X4
M06	Automatic Tool Change	No	
M07	Coolant On (mist)	Yes	Supported as OutputFlag.X0
M08	Coolant On (Flood)	Yes	Supported as OutputFlag.X1
M09	Coolant Off	Yes	Supported as OutputFlag.X0 and X1
M10	Pallet Clamp On	Yes	Supported as OutputFlag.X2
M11	Pallet Clamp Off	Yes	Supported as OutputFlag.X2
M13	Spindle on (clockwise rotation) and coolant on (flood)	Yes	Supported as OutputFlags X1 and X3
M19	Spindle Orientation	No	
M21	Mirror, X-Axis	No	
M22	Mirror, Y-Axis	No	
M23	Mirror Off	No	
M24	Thread gradual pullout off	No	
M30	End of program, with return to program top	No	
M41	Gear select 1	No	
M42	Gear select 2	No	
M43	Gear select 3	No	
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	
M98	Subprogram call	No	
M99	Subprogram end	No	

Parameters

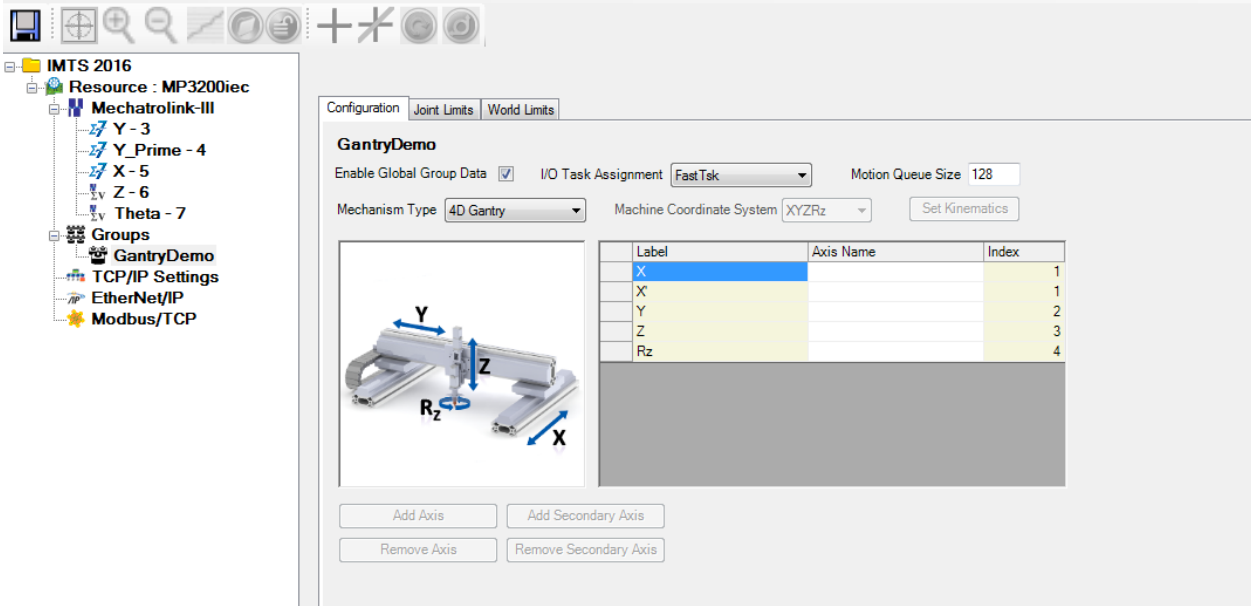
Command	Description	Support	Comment
A	Absolute or incremental position of A axis (rotational axis around X axis)	Yes	This is a Rotational position. (Rx) Actual implementation handled in firmware based on Hardware Configuration support for selected mechanisms.
B	Absolute or incremental position of B axis (rotational axis around Y axis)	Yes	This is a Rotational position. (Ry) Actual implementation handled in firmware based on Hardware Configuration support for selected mechanisms.
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	Precision feedrate for threading on lathes	No	
E	Extruder axis position for 3D Printing	Yes	
F	Feed rate	Yes	Specify in units / minute.
H	Tool length offset	No	
I	Arc Center in X axis for G02 or G03	Yes	
J	Arc Center in Y axis for G02 or G03	Yes	
K	Arc Center in Z axis for G02 or G03	Yes	
L	Fixed cycle loop count	No	
N	Line number (optional)	Yes	
O	Program Name	No	
P	Dwell time (G04), parameter for some canned cycles, and for jumps	Yes	Support For G04 only
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 speed or surface speed depending on mode	No	
T	Tool Selection	Yes	
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.

3D Printing

Some special configuration is required to enable 3D printing support.

Configure the Group in Hardware Configuration

Configure a 4D gantry. Consider the user units of the theta axis.



- IMTS 2016
  - Resource : MP3200iec
    - Mechatrolink-III
      - Y - 3
      - Y_Prime - 4
      - X - 5
      - Z - 6
      - Theta - 7
    - Groups
      - GantryDemo
    - TCP/IP Settings
    - EtherNet/IP
    - Modbus/TCP

Configuration | Joint Limits | World Limits

GantryDemo

Enable Global Group Data ☑ I/O Task Assignment: FastTsk Motion Queue Size: 128

Mechanism Type: 4D Gantry Machine Coordinate System: XYZRz

Label	Axis Name	Index
X		1
X'		1
Y		2
Z		3
Rz		4

Configure the Extruder in MachineStruct

If the G Code data will contain extruder position / feed information as "E" axis data, use the following initialization code:

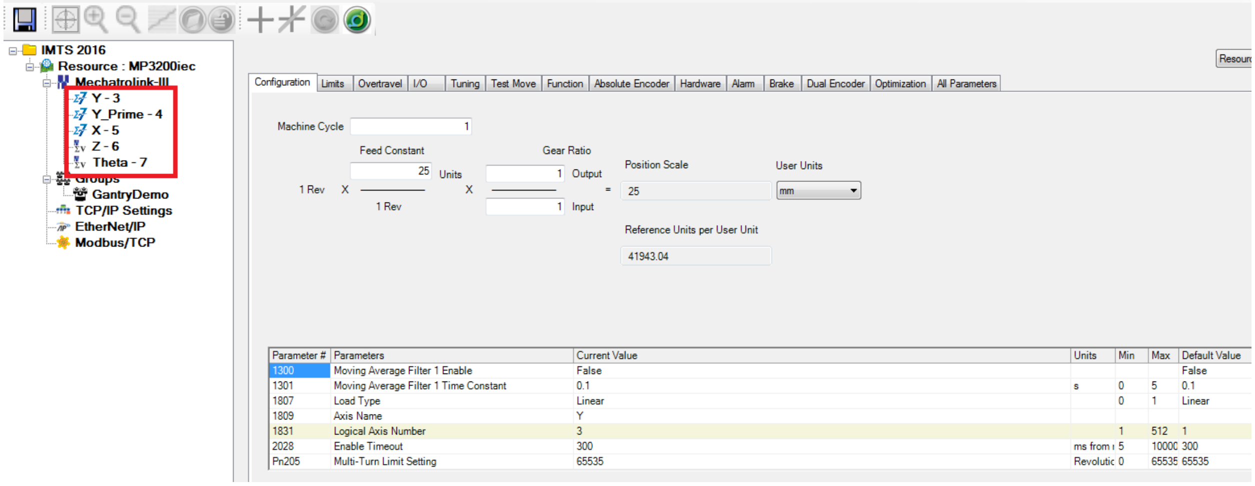
```
MyMachine.Extruder.Axis.AxisNum:=INT#7; (* The value is the AXIS_REF for the Rz axis. *)
```

Tangent Mode

Tangent mode operates a theta axis tangent to the path and can be activated / deactivated using G78 and G79.

Configuring a System for Tangent operation

1) Configure all the axes using the Hardware Configuration.



Machine Cycle 1

Feed Constant: 1 Rev X 25 Units / 1 Rev

Gear Ratio: X 1 Output / 1 Input

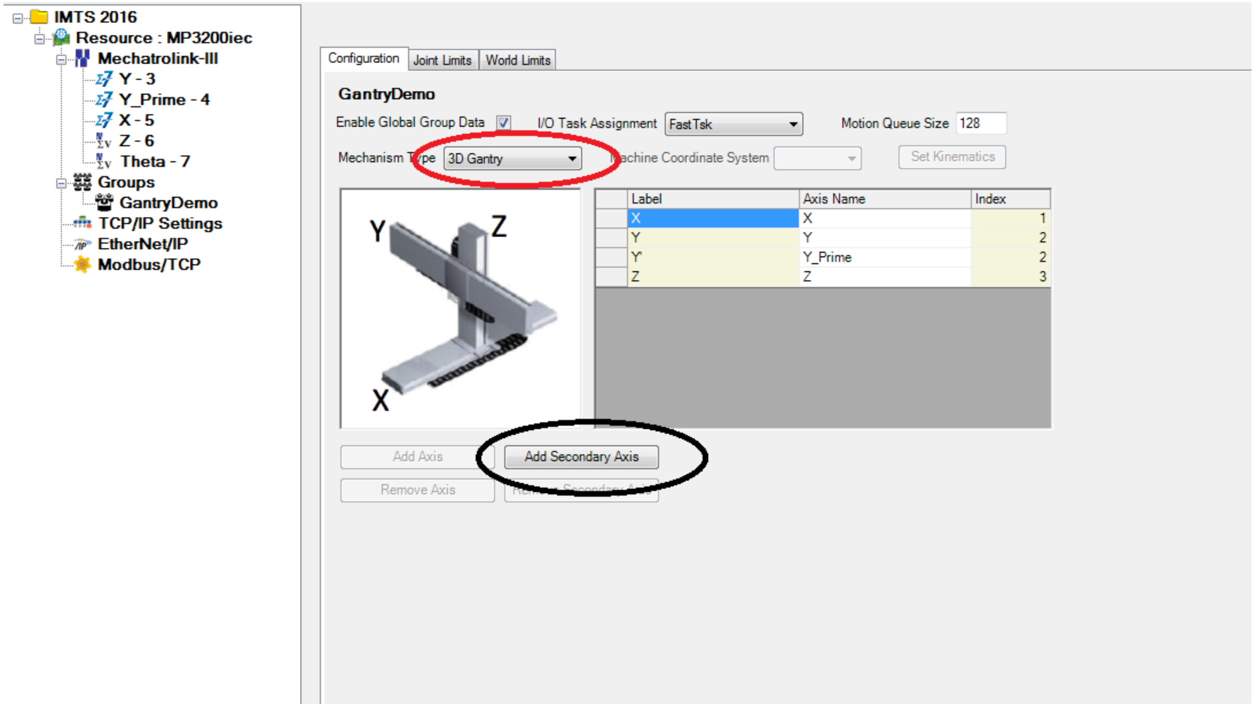
Position Scale = 25

User Units mm

Reference Units per User Unit 41943.04

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 ı	5	10000	300
Pn205	Multi-Turn Limit Setting	65535	Revolutic	0	65535	65535

2) 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.



- IMTS 2016
  - Resource : MP3200iec
    - Mechatrolink-III
      - Y - 3
      - Y_Prime - 4
      - X - 5
      - Z - 6
      - Theta - 7
    - Groups
      - GantryDemo
    - TCP/IP Settings
    - EtherNet/IP
    - Modbus/TCP

GantryDemo

Enable Global Group Data ☑ I/O Task Assignment FastTsk Motion Queue Size 128

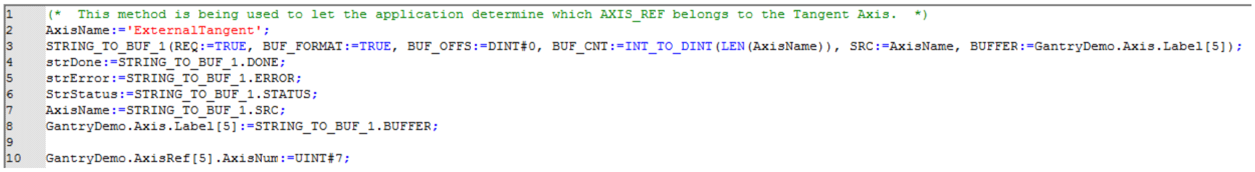
Mechanism Type 3D Gantry Machine Coordinate System

	Label	Axis Name	Index
	X	X	1
	Y	Y	2
	Y'	Y_Prime	2
	Z	Z	3

3) Create an Initialization POU.

Although the Theta axis is not officially part of the Group, manually add Theta's AXIS_REF to the AxesGroup.AxisRef structure as shown below on line 10. By this method, the Y_GroupPower / GroupControl function blocks will also operate the theta axis.

The initialization example below also shows how to add the AxisLabel 'ExternalTangent' to the AxesGroup structure. This is essential for the MC_MovePath function block to operate the Tangent axis correctly. It looks for the Axis Label spelled exactly as shown on line two below.



```
1   (*  This method is being used to let the application determine which AXIS_REF belongs to the Tangent Axis.  *)
2   AxisName:='ExternalTangent';
3   STRING_TO_BUF_1(REQ:=TRUE, BUF_FORMAT:=TRUE, BUF_OFFS:=DINT#0, BUF_CNT:=INT_TO_DINT(LEN(AxisName)), SRC:=AxisName, BUFFER:=GantryDemo.Axis.Label[5]);
4   strDone:=STRING_TO_BUF_1.DONE;
5   strError:=STRING_TO_BUF_1.ERROR;
6   StrStatus:=STRING_TO_BUF_1.STATUS;
7   AxisName:=STRING_TO_BUF_1.SRC;
8   GantryDemo.Axis.Label[5]:=STRING_TO_BUF_1.BUFFER;
9
10  GantryDemo.AxisRef[5].AxisNum:=UINT#7;
```

Group_DataTypes

Toolbox Help Documentation

Help version created 1/25/2017

Data Type: BufferStruct

This is a sub structure of MC_PATH_DATA_REF which contains data about the circular buffer of Segments.

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 function for MyPathStruct.Segment.	MyPathStruct.Buffer.Size
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
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
C	MotionAvailable	DINT	Populated in MC_MovePath for the StreamStruct if the Read_GCode_Stream method is used. This pertains to the firmware level motion buffer and is identical to AxesGroup.Status.FreeMotionSegments.	MyPathStruct.Buffer-.MotionAvailable
C	MotionPercent	REAL	Populated in MC_MovePath for the StreamStruct if the Read_GCode_Stream method is used. This pertains to the firmware level motion buffer and is identical to AxesGroup.Status.FreeMotionSegments.	MyPathStruct.Buffer-.MotionPercent

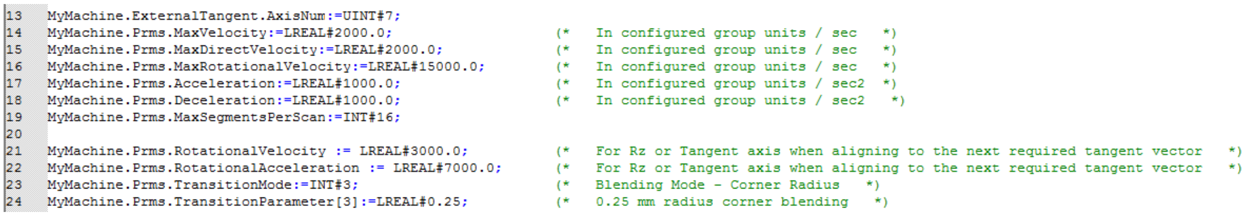
Data Type: MachineStruct

Holds configuration data used by MC_MovePath, Read_GCode_File, and Read_GCode_Stream.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyMachine			
U	Origin	MC_CARTESIAN_REF	The home position of the machine. Used in conjunction with G Code 28.	MyMachine.Origin[x]
U	CoordinateSystem	OffsetStruct	Holds coordinate offsets for up to 9 offsets, corresponding to G54 through G59.3	MyMachine.CoordinateSYstem.Offset[x]
U	Prms	MotionParameters	User configurable parameters such as max velocity and max acceleration.	MyMachine.Prms.MaxVelocity
U	Extruder	ExtruderStruct	Data for a 3D Printer extruder axis. This data is typically just collected from the G Code stream and populated here for access by the program, which must be customized to use it as required for the application.	MyMachine.Extruder.Axis.AxisNum
U	ExternalTangent	AXIS_REF	AXIS_REF of axis configured for tangent motion. This axis is not part of the group configuration.	MyMachine.ExternalTangent.AxisNum

Example Initialization



```
MyMachine.ExternalTangent.AxisNum:=UINT#7;
MyMachine.Prms.MaxVelocity:=LREAL#2000.0;                   (*   In configured group units / sec    *)
MyMachine.Prms.MaxDirectVelocity:=LREAL#2000.0;             (*   In configured group units / sec    *)
MyMachine.Prms.MaxRotationalVelocity:=LREAL#15000.0;        (*   In configured group units / sec    *)
MyMachine.Prms.Acceleration:=LREAL#1000.0;                  (*   In configured group units / sec2   *)
MyMachine.Prms.Deceleration:=LREAL#1000.0;                  (*   In configured group units / sec2    *)
MyMachine.Prms.MaxSegmentsPerScan:=INT#16;

MyMachine.Prms.RotationalVelocity := LREAL#3000.0;          (*   For Rz or Tangent axis when aligning to the next required tangent vector    *)
MyMachine.Prms.RotationalAcceleration := LREAL#7000.0;      (*   For Rz or Tangent axis when aligning to the next required tangent vector    *)
MyMachine.Prms.TransitionMode:=INT#3;                       (*   Blending Mode - Corner Radius    *)
MyMachine.Prms.TransitionParameter[3]:=LREAL#0.25;          (*   0.25 mm radius corner blending    *)
```

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

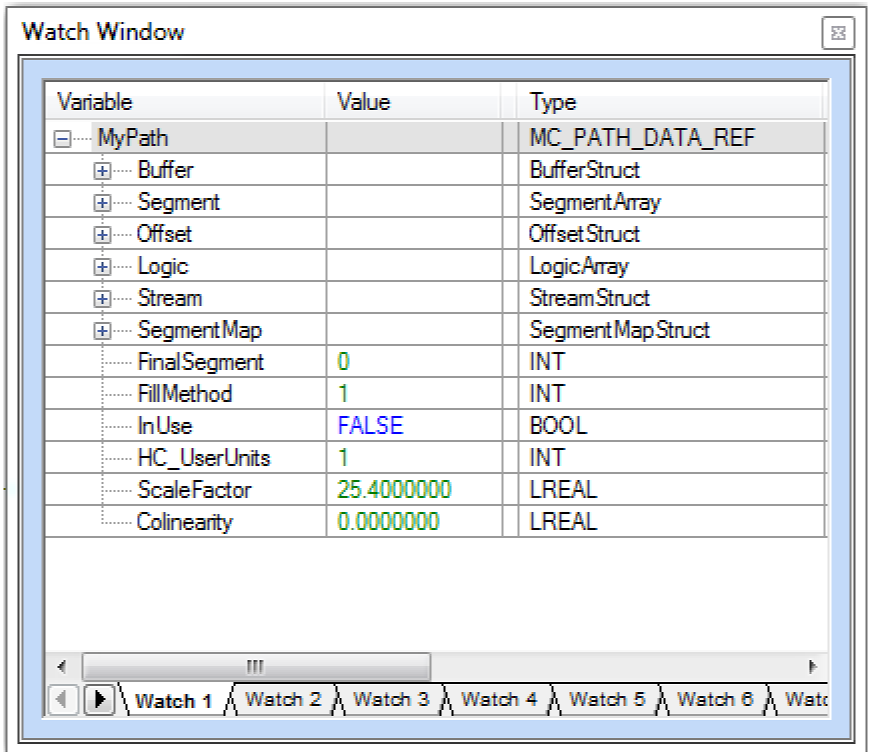
The column that indicates whether the 'U'ser of the 'C'ontroller write the data can be misleading for this data type, as there are function blocks available such as Read_GCode_File which act on behalf of the user to populate many of these values. In all cases, 'C' indicates that the user should not write 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
	MyPathStruct	**MC_PATH_DATA_REF**		
C/-U	Buffer	BufferStruct	Sub structure containing information for managing a 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.	MyPathStruct.Buffer.StorePointer
U	Segment	SegmentArray	Contains all details required for each of the Segment in the path.	MyPathStruct.Segment[0].X
U	Logic	LogicArray	For support of SegmentType LoopDecision. Up to 32 loops or jumps can be configured in PathData. Limitation: 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 .	MyPathStruct.Logic[n].DecisionIndex
C	Stream	StreamStruct	For the Read_GCode_Stream function block. This customizable structure contains all relevant information required by the source sending path data via Ethernet socket.	MyPathStruct.Stream.PathStatus.PathSegment

C	SegmentMap	SegmentMapStruct	MC_MovePath uses this for tracking the relationship between MyPathStruct.Segment[n] and the unique motion segment ID used 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.	MyPathStruct.SegmentMap.Map[0][12].PathIndex
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.	MyPathStruct.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.	MyPathStruct.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 and error if upon the rising edge it detects that MC_MovePath is still using the data.	MyPathStruct.InUse
U	HC_UserUnits	INT	Stores the value /code of the user units set in the Hardware Configuration.	MyPathStruct.HC_UserUnits
U	ScaleFactor	LREAL	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.	MyPathStruct.ScaleFactor
U	Colinearity	LREAL	Specify an angle which must invoke an 'ExactStopCheck'. For example, when shape cutting, if multiple segments are nearly co linear (less than x 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	MyPathStruct.Colinearity

			degrees), a co linearity value of say 85 degrees can be detected, and cause motion to stop before raising the tool.	

Watch Window Views

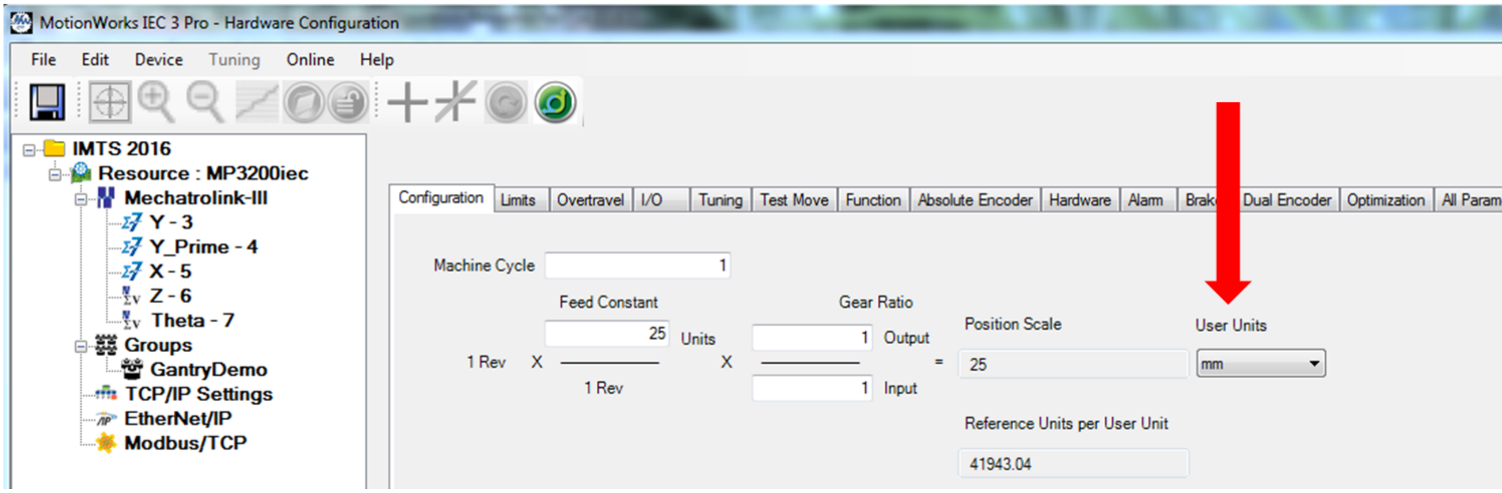


Variable	Value	Type
MyPath		MC_PATH_DATA_REF
Buffer		BufferStruct
Segment		SegmentArray
Offset		OffsetStruct
Logic		LogicArray
Stream		StreamStruct
SegmentMap		SegmentMapStruct
FinalSegment	0	INT
FillMethod	1	INT
InUse	FALSE	BOOL
HC_UserUnits	1	INT
ScaleFactor	25.4000000	LREAL
Colinearity	0.0000000	LREAL

Example 1 - Basic motion

Example 2 - Determining Hardware Configuration User Units.

This example shows Axis as the second axis in GantryDemo, which



Machine Cycle: 1

Feed Constant: 25 Units / 1 Rev

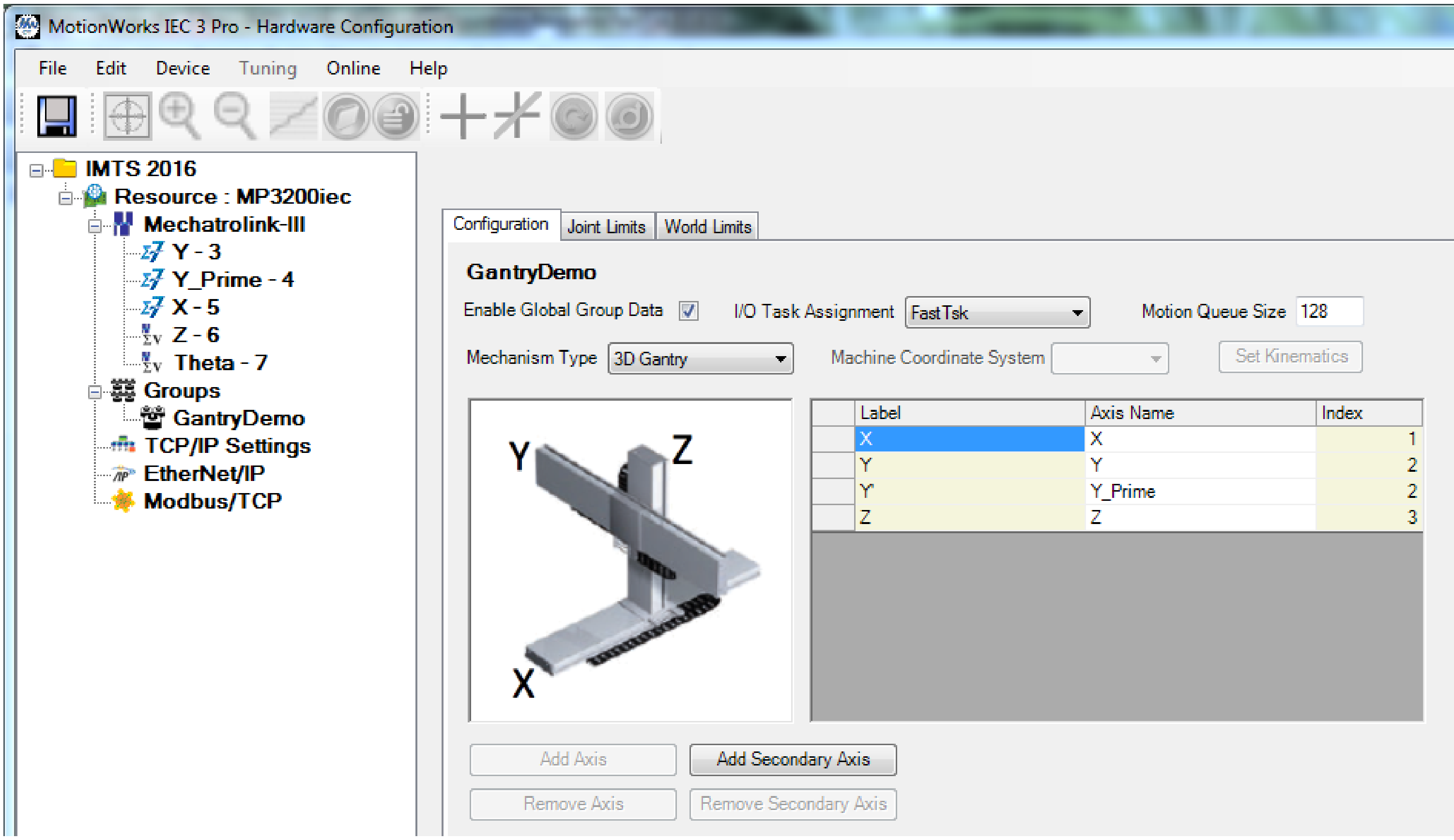
1 Rev X Feed Constant X Gear Ratio = Position Scale

Gear Ratio: Output 1 / Input 1

Position Scale: 25

User Units: mm

Reference Units per User Unit: 41943.04



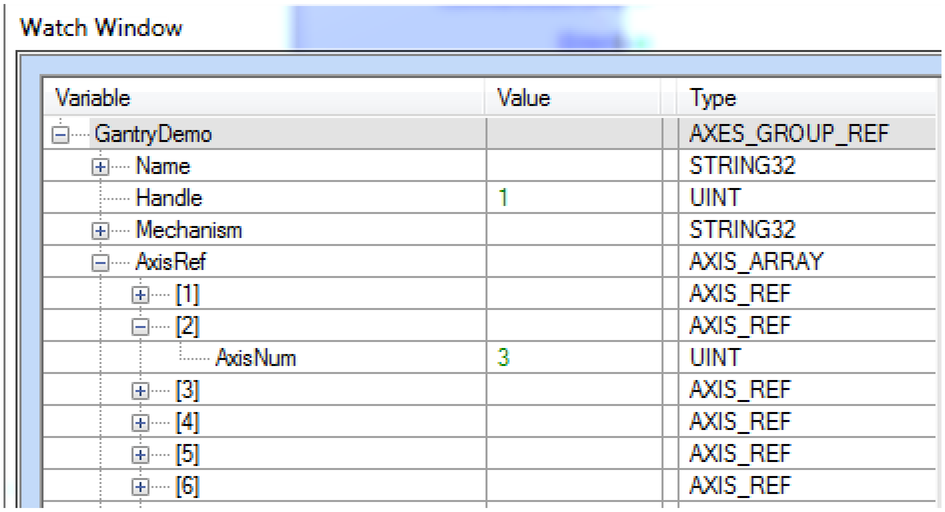
GantryDemo

Enable Global Group Data ☑ I/O Task Assignment: FastTsk Motion Queue Size: 128

Mechanism Type: 3D Gantry Machine Coordinate System:

Label	Axis Name	Index
X	X	1
Y	Y	2
Y'	Y_Prime	2
Z	Z	3

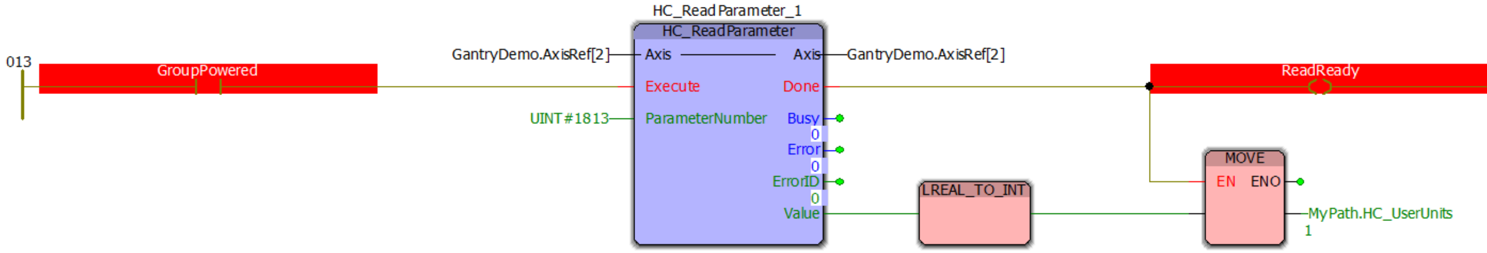
According to the Hardware Configuration and confirmed by the Watch Window, GantryDemo.AxisRef[2] will report the user units of the Y axis. We assume the units for X and Z are the same.

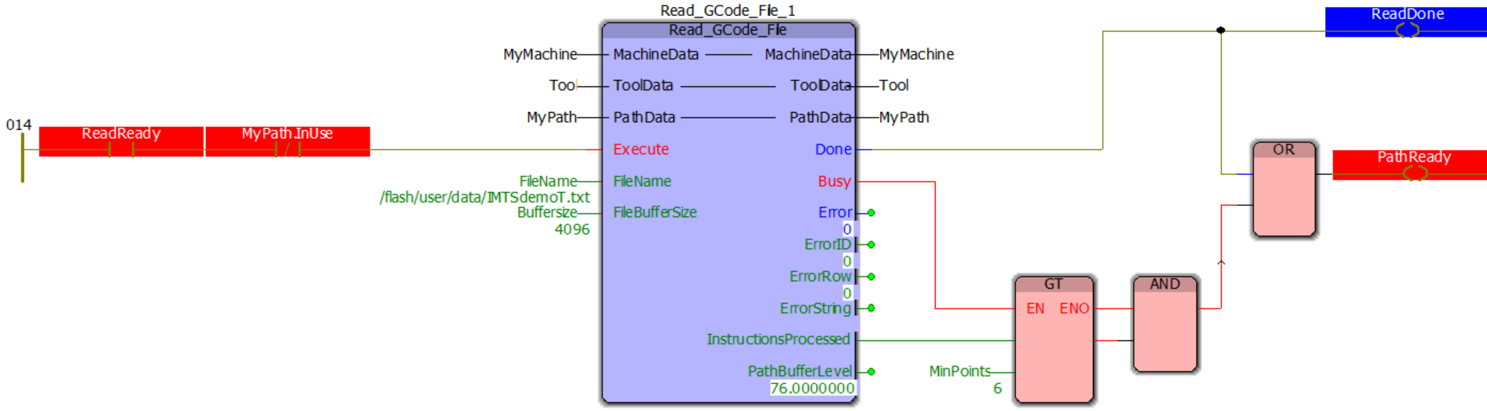


Watch Window

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

HC_ReadParameter is located in the FileRW_Toolbox.





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 form the MachineStruct.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MachineStruct.Prms	**MotionParameter**		
U	MaxVelocity	LREAL	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.	MachineStruct.Prms.MaxVeloccity
U	MaxDirectVelocity	LREAL	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.	MachineStruct.Prms.MaxDirectVeloccity
U	MaxRotationalVelocity	LREAL	Max velocity in rotational units/sec when a motion segment contains a rotational change but no Cartesian motion.	MachineStruct.Prms.MaxRotationalVeloccity
U	Acceleration	LREAL	Acceleration in Cartesian units/sec2 applied to linear (G1) and direct Cartesian motion.	MachineStruct.Prms.Acceleration
U	Deceleration	LREAL	Deceleration in Cartesian units/sec2 applied to linear (G1) and direct Cartesian motion.	MachineStruct.Prms.Deceleration
U	Jerk	LREAL	Not supported.	
U	DirectAcceleration	LREAL	Acceleration in Cartesian units/sec2 applied dir-	MachineStruct.Prms.DirectAcceleration

			ect (G0) Cartesian motion.	
U	DirectDeceleration	LREAL	Deceleration in Cartesian units/sec2 applied to direct (G0) Cartesian motion.	MachineStruct.Prms.DirectDeceleration
U	DirectJerk	LREAL	Not Supported.	
U	RotationalVelocity	LREAL	Velocity used 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
U	MaxSegmentsPerScan	INT	The maximum number of similar motion functions that will be executed in the same scan. 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 they are being consumed at a fast rate. For most applications however, especially those with fewer, longer segments, setting MaxSegmetsPerScan to 1 is adequate. See the "Motion Queue Size" in the hardware Configuration under the Group item in the configuration tree.	MachineStruct.Prms.MaxSegmentsPerScan
U	TransitionMode	INT	This will be the default Transition mode used as the input to PLCopen Part 4 motion blocks. See Trans-	MachineStruct.Prms.TransitionMode

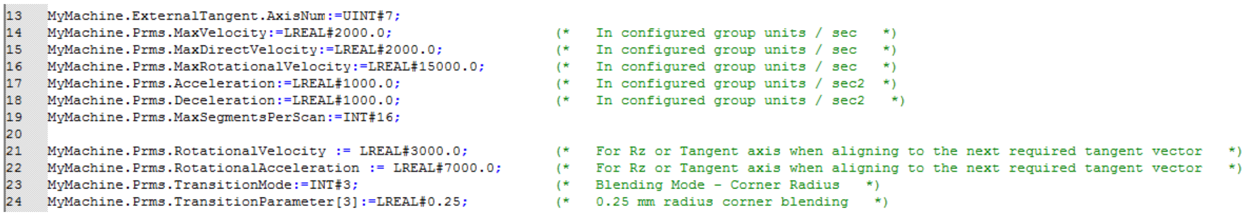
			ition Mode in the PLCopen Motion Function Blocks help manual.	
U	TransitionParameter	VECTOR	This will be the default TransitionParameter used as the input to PLCopen Part 4 motion blocks. See Transition Mode in the PLCopen Motion Function Blocks help manual.	MachineStruct.Prms.TransitionParameter

Special Notes for G Code Applications:

User units for all values in the MachineStruct must be those of the AxesGroup, even though support for alternate user units which may be contained in the data is supported. 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/sec^2 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.



```
MyMachine.ExternalTangent.AxisNum:=UINT#7;
MyMachine.Prms.MaxVelocity:=LREAL#2000.0;                 (*   In configured group units / sec    *)
MyMachine.Prms.MaxDirectVelocity:=LREAL#2000.0;           (*   In configured group units / sec    *)
MyMachine.Prms.MaxRotationalVelocity:=LREAL#15000.0;      (*   In configured group units / sec    *)
MyMachine.Prms.Acceleration:=LREAL#1000.0;                (*   In configured group units / sec2   *)
MyMachine.Prms.Deceleration:=LREAL#1000.0;                (*   In configured group units / sec2    *)
MyMachine.Prms.MaxSegmentsPerScan:=INT#16;

MyMachine.Prms.RotationalVelocity := LREAL#3000.0;        (*   For Rz or Tangent axis when aligning to the next required tangent vector   *)
MyMachine.Prms.RotationalAcceleration := LREAL#7000.0;    (*   For Rz or Tangent axis when aligning to the next required tangent vector   *)
MyMachine.Prms.TransitionMode:=INT#3;                     (*   Blending Mode - Corner Radius   *)
MyMachine.Prms.TransitionParameter[3]:=LREAL#0.25;        (*   0.25 mm radius corner blending    *)
```

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Data Type: OffsetStruct

Sub structure of the MachineStruct which holds data related to the work coordinate offsets. When a G10 command is detected in a G Code file, 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 the G Code data contains a G54 through G59.3 command, the functions will apply the selected offset values accordingly.

This data can be populated by the G Code G10 or from another source such as an HMI.

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

Example

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

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
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_STRING8	For debugging purposes, short descriptive labels can be added. When MC_MovePath is executing, it will display the value as a STRING(8) at the VAR_OUTPUT 'SegmentLabel.'	MyPathStruct.Segment[n].Label
U	X	LREAL	If SegmentType is StraightLine, Arc,	MyPathStruct.Segment[n].X

			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.	
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
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	DWORD	For use with G Code applications using G92 to Set Position. These flags identify which axes the command applies to.	MyPathStruct.Segment [n].AxesFlags
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,	MyPathStruct.Segment [n].CoordSystem

					this data is ignored.	

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	LREAL	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	Resolution	REAL	Reserved for future use.	MyPathStruct.Segment[n].Resolution
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
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 'CircMode' 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	InputConditions	DWORD	Used only if SegmentType is 'WaitForInputs', 'LoopDecision', 'BranchDecision', 'NonBlockingInputCheck', 'NonBlockingLoopDecision', or 'NonBlockingBranchDecision'. Set the InputConditions required to advance to the next Segment.	MyPathStruct.Segment[n].InputConditions
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	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.	MyPathStruct.Segment[n].TangentActive

			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.	
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	LogicIndex	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

Example 1

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Data Type: SegmentMapStruct

This structure is used by MC_MovePath to keep track of 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

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Data Type: StreamStruct

Holds data for reporting to a host application via Ethernet. This structure will likely require customization when developing for a host PC application.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyStream	**StreamStruct**		
C	Position	LREALArray6	Array of 6 world space positions	MyStream.Position[[3]
C	Velocity	LREALArray6	Array of 6 world space velocities	MyStream.Velocity[1]
C	Torque	LREALArray6	Array of 6 world space torques	MyStream.Torque[5]
C	PathStatus	PathStatusStruct	Data pertaining to the path such as segments processed, current path segment.	MyStream.PathStatus.PathSegment
C	Buffer	BufferStatusStruct	Data pertaining to the three buffers involved with processing data: ByteBuffer, PathBuffer, and MotionBufer.	MyStream.Buffer.MotionAvailable

Data Type:ToolStruct

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 must be provided in the user units of the machine. There is support for G Code data to be provided in alternate user units (G20 = inches, G21 = mm) but the data will be scaled to that of the user unit configuration of the group before Tool Compensation is applied.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyTools	**ToolStruct**		
U	Radius	LREAL	Radius of a tool in the groups Cartesian user units from the Hardware Configuration	MyTools [2].Radius
U	Length	LREAL	Length of a tool in the groups Cartesian user units from the Hardware Configuration	MyTools [4].Length

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).

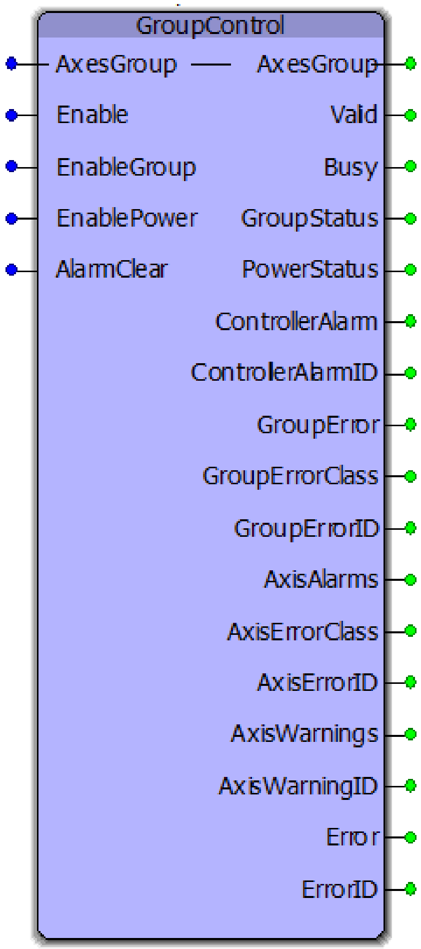
Enumerated Types Declaration

Enumerated Type	#INT Value	Enum Value	Description
TB_PatternType	**ENUM Type for PathDetails' SegmentType**		
	0	n/a	Not a valid PatternType
	1	StraightLine	Straight line motion between two machine coordinate locations.
	2	Arc	Arc or circular path.
	3	StandStill	Pause between segments.

	4	WaitForInputs	Path processing will wait until the specified input conditions are met.
	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 machine coordinate locations.
	7	LoopDecision	Jump to another non sequential Segment based on InputConditions.
	8	BranchDecision	End the path early based on InputConditions.
	9	NonBlockingInputCheck	Don’t wait to process Segments after this input check 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 InputConditions are checked. As soon as they are satisfied, processing of the segments after type 9 resumes.
	10	InputLookAhead	Reserved for future use.
	11	MoveToOrgin	Reserved for future use.
	12	SetPosition	Set the position of an extruder axis. (3D Printing support)
	13	NonBlockingLoopDecision	Don’t wait to process this segment type until previously buffed motion has completed.
	14	NonBlockingBranchDecision	Don’t wait to process this segment type until previously buffed motion has completed.

Group_FBs

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 substructures 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	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:

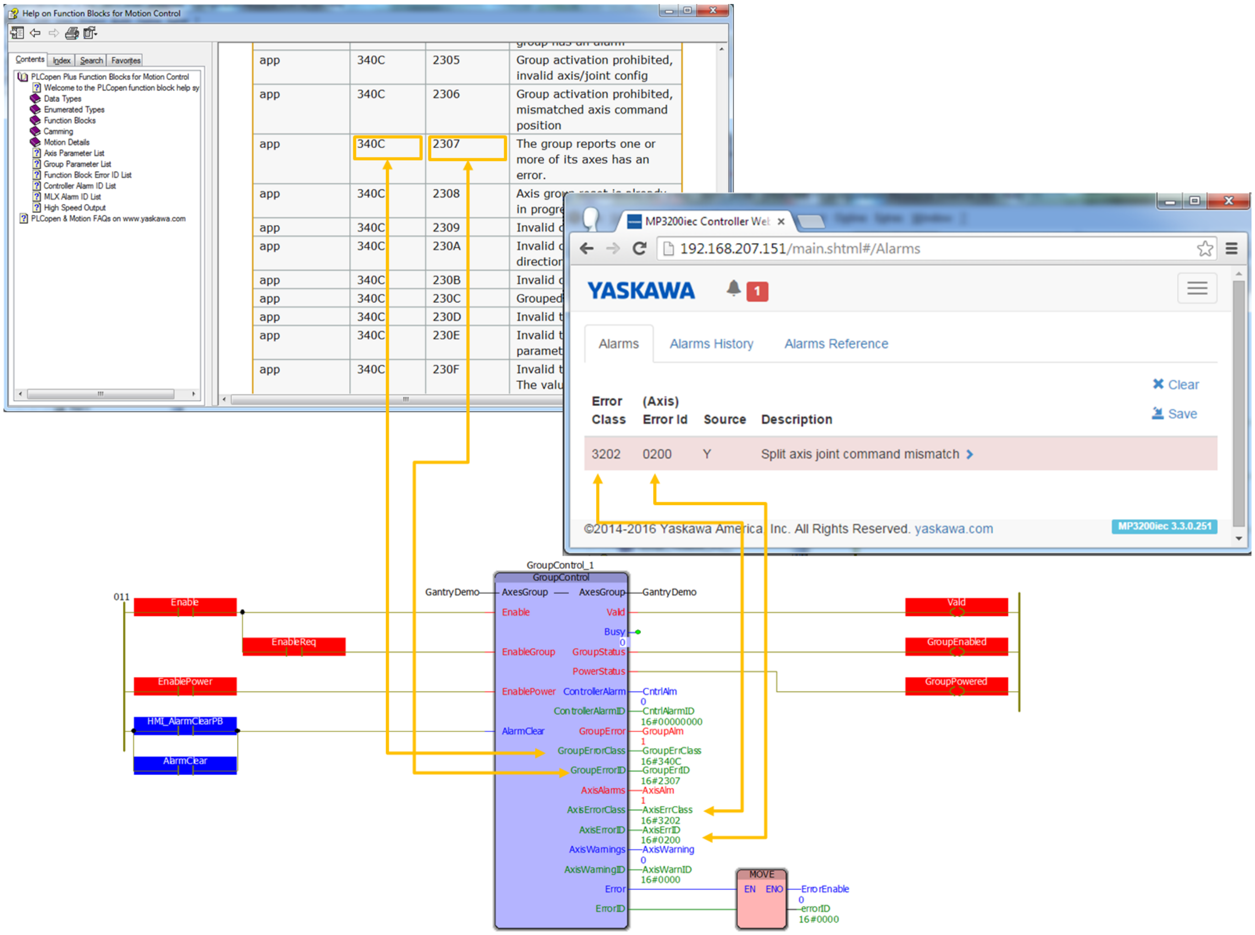
- Sigma-7 Mechatrolink-III: SIEPS8000128, see Section 12.2
- Sigma-5 Mechatrolink-III with rotary motor: SIEPS8000064, see Section 9.1
- Sigma-5 Mechatrolink-III with linear motor: SIEPS8000065, see Section 8.1
- Sigma-5 Mechatrolink-II with rotary motor: SIEPS8000046, see Section 9.1
- Sigma-5 Mechatrolink-II 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

ErrorID	Meaning
4412	Parameter not supported for the specified axis or group.
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 POUs.
8960	Invalid axes group. Confirm that the AxesGroup variable has the correct %M address as automatically assigned by the Hardware Configuration.
8961	An axis is already owned by another group.
8962	Group activation is blocked. Ownership can not be changed while Mechatrolink reset is in progress.
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.
8968	Axis group reset is already in progress.
9216	Invalid Host_ID. Supported Host_IDs are (0 = MECHATROLINK group, 1 = MLX hosted 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.
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.
45332	Sending clear alarms command to servo drive failed.
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.
60909	This function block is not supported for the AxesGroup specified.

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.



Help on Function Blocks for Motion Control

			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 progre
app	340C	2309	Invalid c
app	340C	230A	Invalid c direction
app	340C	230B	Invalid c
app	340C	230C	Grouped
app	340C	230D	Invalid t
app	340C	230E	Invalid t paramet
app	340C	230F	Invalid t The valu

MP3200iec Controller Wel

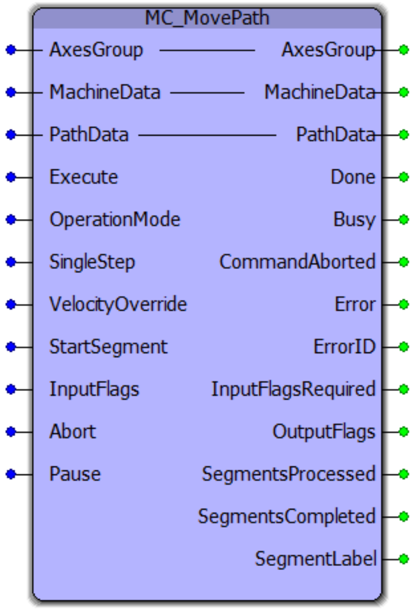
192.168.207.151/main.shtml#/Alarms

Error Class	(Axis) Error Id	Source	Description
3202	0200	Y	Split axis joint command mismatch

©2014-2016 Yaskawa America, Inc. All Rights Reserved. yaskawa.com

MP3200iec 3.3.0.251

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 the following function blocks: Read_GCode_File, Read_GCode_Stream, and CP_Generator.

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	MachineData	MachineStruct	Contains machine parameters including accel and velocity maximums, and support for additional axes such as extruder and tangent axes if required.
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, or CP_Generator but simple sequences could also be populated as a static script in ST code.
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: GT_OperationMode#Normal GT_OperationMode#SingleStepFwd GT_OperationMode#SingleStepBack (Future support) GT_OperationMode#InfiniteRepeat See Notes below for details.	INT#0
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	VelocityOverride	REAL	The 'FeedRate' element of each Segement is multiplied by the VelocityOverride Input. Changes only affect new motion segments executed after a change to VelocityOverride, previous segments already in the motion buffer will be executed at the Feedrate and VelocityOverride specified at the time they were processed.	REAL#100.0
V	StartSegment	INT	The first Segment of the PathData to use upon the rising edge of the Execute input.	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: GT_SegmentType#WaitForInputs GT_SegmentType#LoopDecision GT_SegmentType#BranchDesicion GT_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	Upon the rising edge of Pause, motion will be stopped using the MC_GroupInterrupt function block and no Segments will be processed. Upon the falling edge, the MC_GroupContinue function will be executed and segments will be processed once again.	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	SegmentsProcessed	UDINT	Running count of the number of PathData.Segments[] processed since the rising edge of the Execute input.
V	SegmentsCompleted	UDINT	Running count of the number of PathData.Segments[] completed since the rising edge of Execute. The difference between the SegmentsProcessed and SegmentsCompleted is the number of motion blocks still in the motion buffer.
V	SegmentLabel	YTB_STRING8	For use as a debugging tool. If PathData.Segment[].Label was populated by the source, this Output reports a descriptive name of the Segment being processed.

Notes

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#SingleStepBack is not supported, it is for future use.
- 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.FinalSegment) 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 PathData 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 offsets are stored in the MachineStruct.CoordinateSystem and processed by Read_GCode_File or Read_GCode_Stream. The offsets can also be updated by the application program via an HMI or PC for example.
  - 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.
  - Because work offsets and tool compensation are processed in other function blocks, changes will not occur unless the Read_GCode* block re reads the G Code. If the data is large and being continuously buffered, changes will take effect immediately.

Error Description

The number of possible Errors is large due to the number of sub function blocks within MC_MovePath. Check the master list of Function Bock ErrorID's if the ErrorID is omitted from the following list.

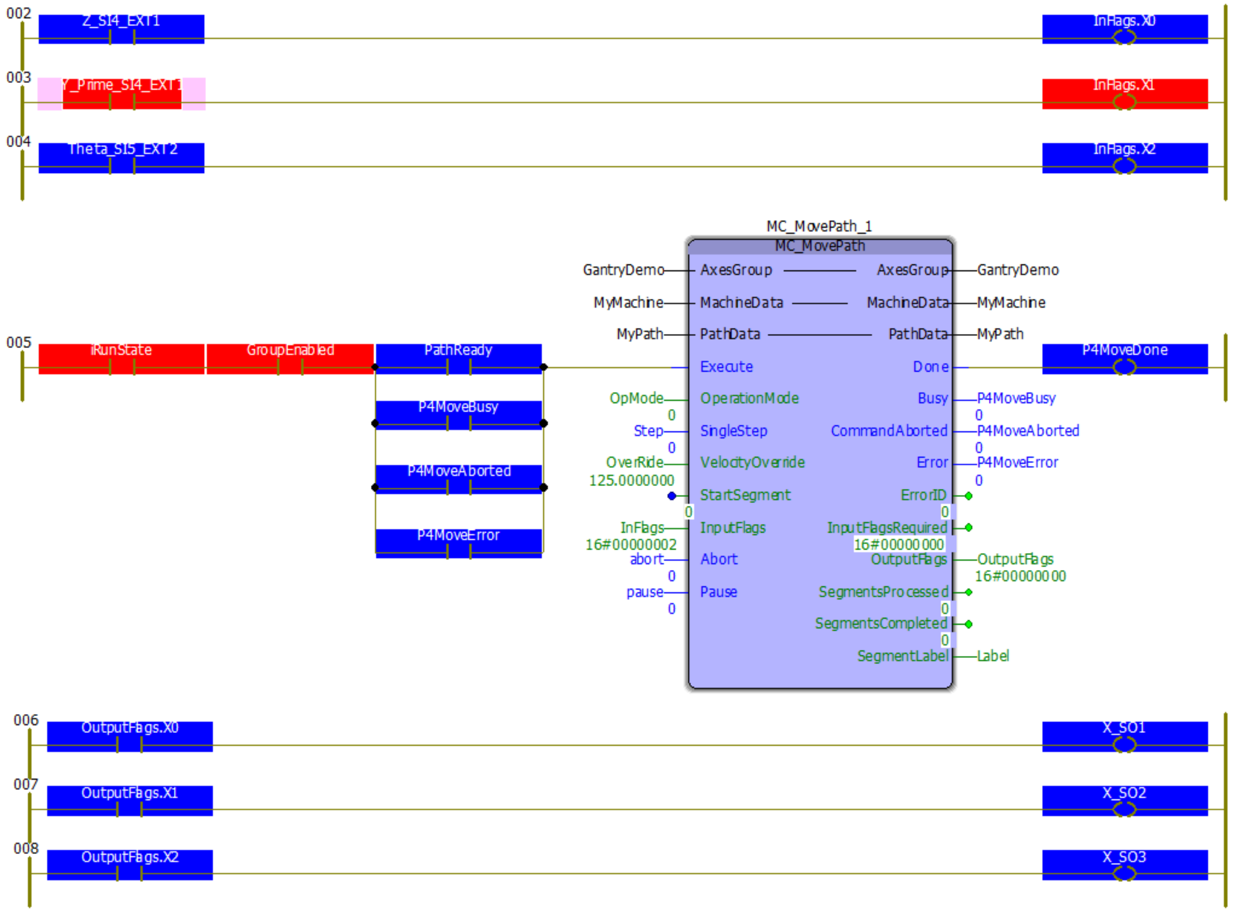
ErrorID	Meaning

0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
4412	Parameter not supported for the specified axis or group.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4654	Invalid Speed Unit setting in Y_MoveOptions.ProfileUnit. Select 0 for Absolute units, or 1 for % of maximum.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4667	Jerk is less than or equal to zero.
4680	Invalid acceleration filter type entered.
4982	Default drive parameter info is not available for this parameter. Use the DataType Override input to specify the parameter size.
4983	The specified external axis may not be used. A physical axis is required.
8960	Invalid axes group. Confirm that the AxesGroup variable has the correct %M address as automatically assigned by the Hardware Configuration.
8963	Invalid coordinate system.
8964	Move prohibited because group has an alarm.
8969	Invalid circular path method.
8970	Invalid circular path direction.
8971	Invalid circle geometry.
8973	Invalid transition mode.
8974	Invalid transition parameter.
9216	Invalid Host_ID. Supported Host_IDs are (0 = MECHATROLINK group, 1 = MLX hosted 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.
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. A command for an invalid state transition was issued.
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.
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 liveman switch is preventing manual mode operation.
9253	The Group's Safety circuit is preventing motion.
9420	Calculation leads to a singularity.
10022	Internal error in the StoreMotionSegment function block. It tried to overwrite the circular buffer data. Call Yaskawa to report the issue.

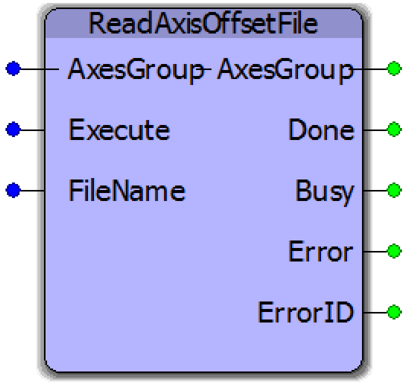
10054	One of the segments in the path has an invalid Segment Type. Valid Segments Types are defined in Group Toolbox GroupTypes file as enumeration GTB_SegmentType.
10136	Loop Error. The ability to repeat sections of the path requires that the entire path is contained in the PataData structure at once. A LoopDecision or NonBlockingLoopDecision segment type detected that PathData.Buffer.Overwritten = TRUE.
10137	Path Dirty Error - The path has been overwritten (circular buffer) so it cannot be re executed from the beginning. Re read teh file or re start the stream.
10607	Segment Error. A function inside MC_MovePath could not find a SegmentID for the current motion that matches one assiged when the motion function block was executed.
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 struture.
57617	Instance object is NULL.
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.
60909	This function block is not supported for the AxesGroup specified.
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. Tangent axis configured as Load Type = linear in Hardware Configuration?

Example

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.



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 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:

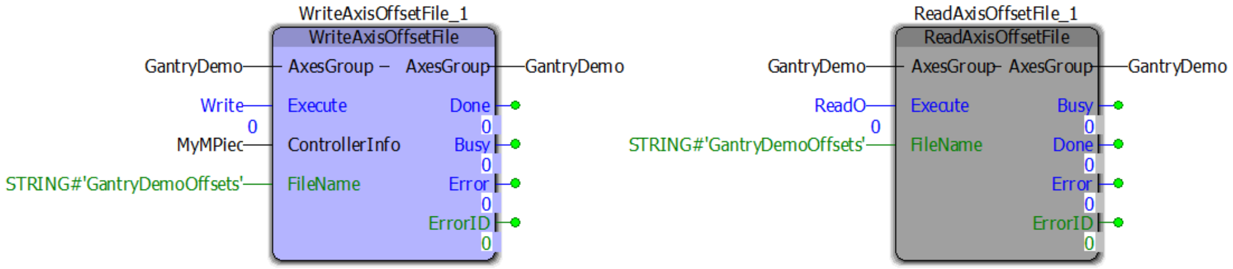
- Firmware version 3.3.0 is required to use this function. It relies on MC_ReadParameter 1838, which was added for version 3.3.0.
- The Flash/Local directory was also added for firmware 3.3.0. It is not deleted when a project archive is deleted or the controller is restored to factory defaults.
- Once 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

ErrorID	Meaning
2	Maximum number of files are already open.
4	File is already opened.
5	File is write protected or access is denied.
6	File name not defined.
4378	The function block is not applicable for the external axis specified.
4392	The function block can not be used with an inverter axis.
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 POUs.
4648	The parameter number does not exist for the specified axis.
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.
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.

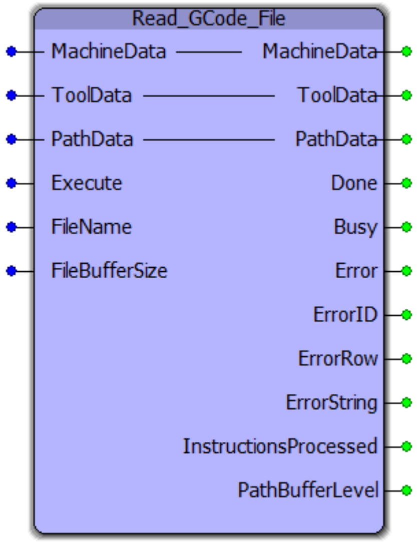
10124	Unsupported Case condition.
10125	Conversion Error. Check the ErrorRow and ErrorCol / ErrorString outputs for details.
10126	NoDataError - The End Of File was reached, but the record count is zero. Verify the file is not corrupted.
10127	TooManyRecords - DataType is not large enough.
10128	MaxNotDefined - The user must set the maximum number of records that can be added to the structure.
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.
10166	File Not Found
10168	Buffer Size Error.
10619	Invalid file name. File names must only contain alphanumeric characters. The first character must not be numeric.

Example



`

Read_GCode_File



This function block reads a file containing G Codes from the MPiec controller's flash or ramdisk file system and stores the data in a structure 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	PathData	MC_PATH_DATA_REF	Data structure containing the details parsed from the G Code source and used with the MC_MovePath function block.	
V	MachineData	MachineStruct	Holds data such as motion parameters, origin, and external tangent configuration.	
V	ToolData	ToolStruct	Structure holding 21 ToolData structures.	
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	Name of file to be read or written. Example STRING#'/-	STRING#''

			flash/user/data/myFile.csv' FileName can include any extension. Max characters for the total FileName is 24.	
V	FileBufferSize	UDINT	The number of bytes to read from a file 'per cycle'. The maximum is 16384. See notes below for more details.	UDINT#2048
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 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. 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. MC_PATH_DATA_REF has a default size allocation of 250 Segments. Once the buffer becomes full, Read_GCode_File will wait until MC_MovePath has executed 50% of the segments, then read from the file buffer again. 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.

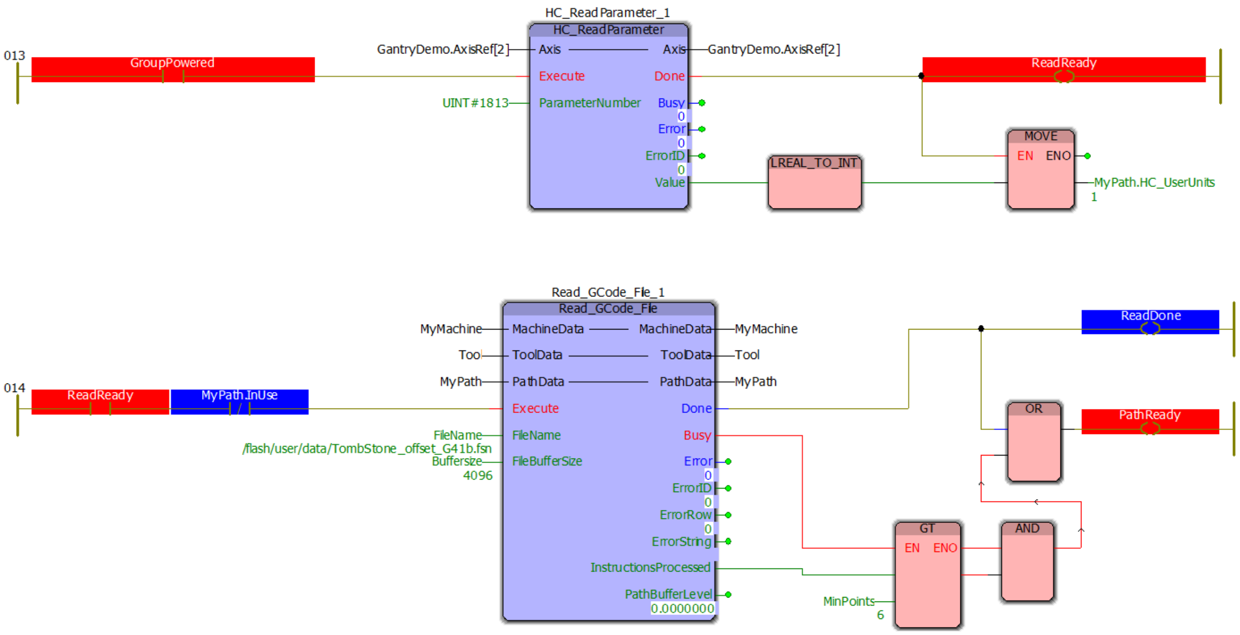
Error Description

ErrorID	Meaning
0	No error.
10054	One of the segments in the path has an invalid Segment Type. Valid Segments Types are defined in Group Toolbox GroupTypes file as enumeration GTB_SegmentType.
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.
10120	File could not be opened. Check for accurate directory path and use of "/"
10122	Row Error. The data is out of sync with the expected row / column arrangement expected.
10123	Column Start Error. The data is corrupted.
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.
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.
10164	Invalid character position input.
10168	Buffer Size Error.
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.
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 GCode ypes file.
10615	G10 Error. The 'L' register must be 1 or 2.

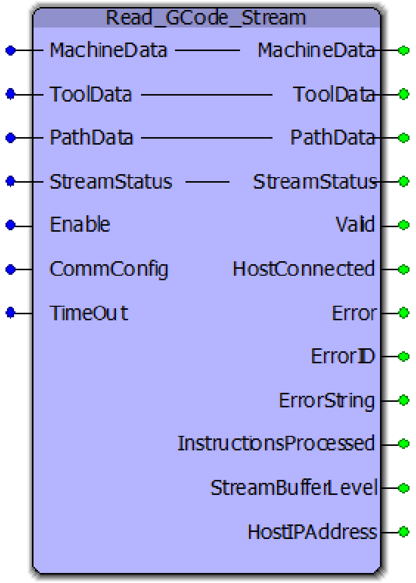
Example

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

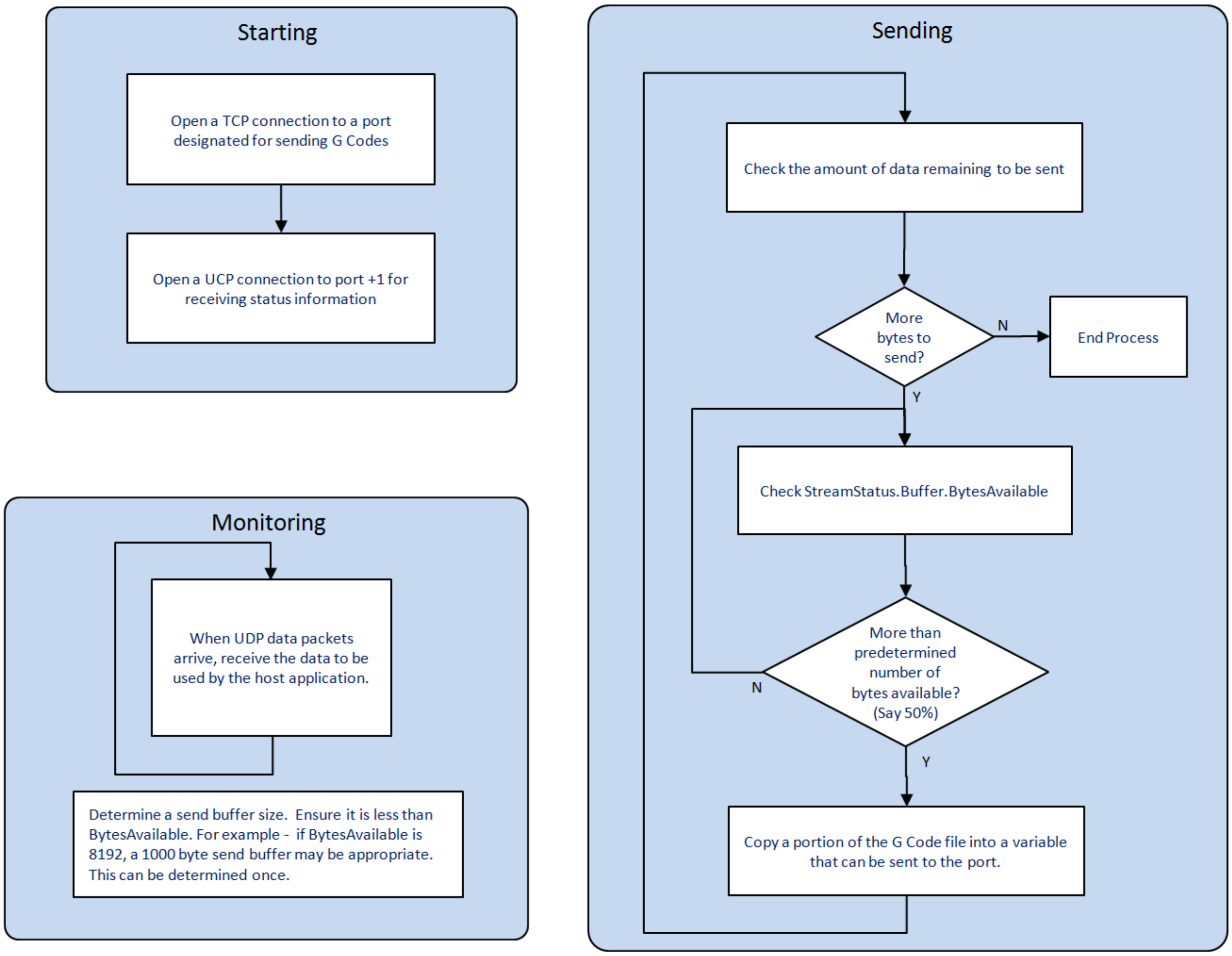
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	MachineData	MachineStruct	Contains data such as motion parameters, origin, and external tangent configuration.	
V	ToolData	ToolStruct	Contains radius and length data for tools that may be selected.	
V	PathData	MC_PATH_DATA_REF	Data structure used with MC_MovePath function block. This structure contains the details parsed from the G Code source.	
V	StreamStatus	StreamStruct	Contains information about the data stream and MPiec buffers. This structure is send back to the host device via UDP packets at the cyclic task interval of the task where this function block executes. The host device should use this information to manage the stream to avoid overloading or starving conditions.	
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	CommConfig	CommStruct	For use with CommunicationChannel function block. Contains information about the communication interface used. See the example below.	All zeros in structure
V	TimeOut	TIME	Set this value if the controller should close the connection and stop waiting for commands.	T#0s (No Timeout)
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 the 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		
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 = 250. 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 (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.	
V	HostIPAddress	STRING	The IP address of the device which initiated a connection request to the MPiec controller for G Code streaming.	

Notes

The streaming approach will likely require some customization to the StreamStatus structure. This is data which is sent back to the host device via UDP packets to inform the host about the status of the bytebuffer, pathbuffer, motionbuffer, and axes positions, as well as other customizable items. Each application may require additional information.

The host application must read the StreamStatus to determine the number of bytes available to send. The host application and the Read_GCode_Stream function block will work together to transmit / receive the byte stream of G Code information.

Basic Flowchart of Host PC application


Starting
Open a TCP connection to a port designated for sending G Codes
Open a UCP connection to port +1 for receiving status information
Monitoring
When UDP data packets arrive, receive the data to be used by the host application.
Determine a send buffer size. Ensure it is less than BytesAvailable. For example - if BytesAvailable is 8192, a 1000 byte send buffer may be appropriate. This can be determined once.
Sending
Check the amount of data remaining to be sent
More bytes to send?
N
End Process
Y
Check StreamStatus.Buffer.BytesAvailable
More than predetermined number of bytes available? (Say 50%)
N
Y
Copy a portion of the G Code file into a variable that can be sent to the port.

Error Description

ErrorID	Meaning
0	No error.
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?)

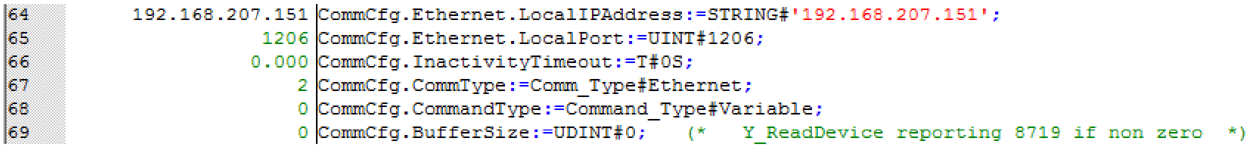
8711	Unable to accept new socket/IO device handle connection.
8712	Unable to bind to the specified address.
8713	The socket/IO device handle type argument was invalid.
8714	The local address or port was not valid.
8715	Connecting to the socket/IO device handle failed.
8716	The remote IP address is unreachable. 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.
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.
10023	Buffer size too small / cannot be zero.
10054	One of the segments in the path has an invalid Segment Type. Valid Segments Types are defined in Group Toolbox GroupTypes file as enumeration GTB_SegmentType.
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.
10120	File could not be opened. Check for accurate directory path and use of "/"
10122	Row Error. The data is out of sync with the expected row / column arrangement expected.
10123	Column Start Error. The data is corrupted.
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.
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.
10164	Invalid character position input.
10168	Buffer Size Error.
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.
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 GCode ypes file.
10615	G10 Error. The 'L' register must be 1 or 2.

Example

The following structured text shows the initialization of CommCfg.

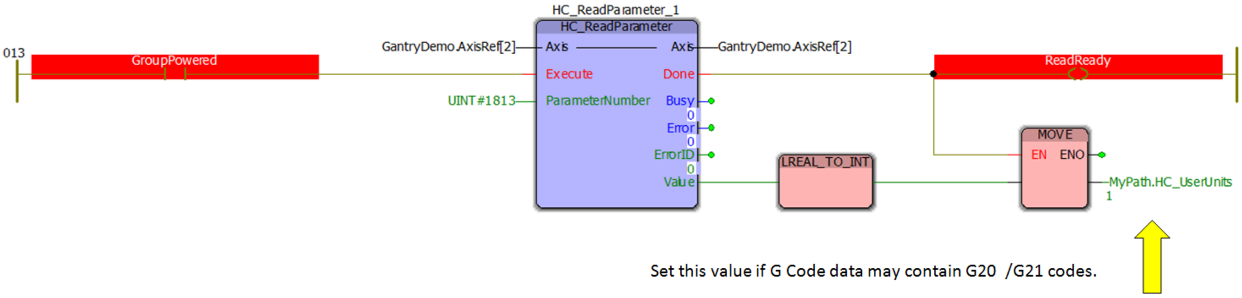
- The LocalIPAddress is that of the MPiec controller. This is necessary because the controller may have more than one IP address and more than one network. It allows the function to listen for G Code data on the correct network. This information can be automatically obtained by referencing the ControllerInfo 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.

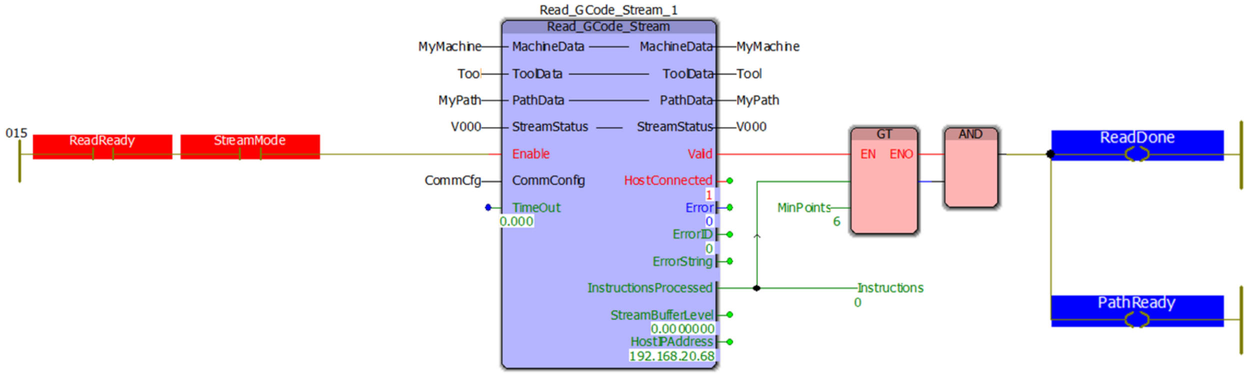


```
192.168.207.151 CommCfg.Ethernet.LocalIPAddress:=STRING#'192.168.207.151';
           1206 CommCfg.Ethernet.LocalPort:=UINT#1206;
          0.000 CommCfg.InactivityTimeout:=T#0S;
              2 CommCfg.CommType:=Comm_Type#Ethernet;
              0 CommCfg.CommandType:=Command_Type#Variable;
              0 CommCfg.BufferSize:=UDINT#0;    (*   Y_ReadDevice reporting 8719 if non zero  *)
```

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 makes Cartesian movement in the group.) Alternatively, if you know the user units of the groups 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 (G20 / G21).

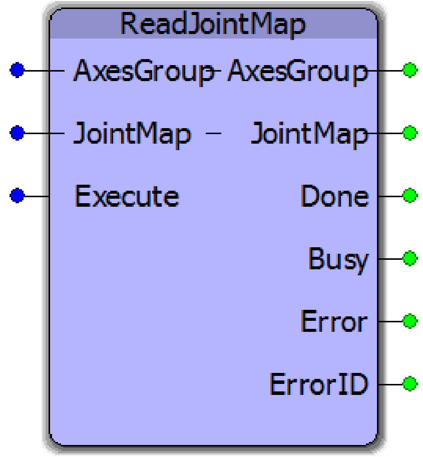


Set this value if G Code data may contain G20 /G21 codes.



In the example above, a minimum number of points must be loaded into MC_PATH_DATA_REF.Segment[] before the PathReady flag is set. This flag is used in the other task which executed the MC_MovePath function block. This allows the move to be buffered and sent to the motion engine more quickly to avoid data starvation when the motions starts.

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.

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

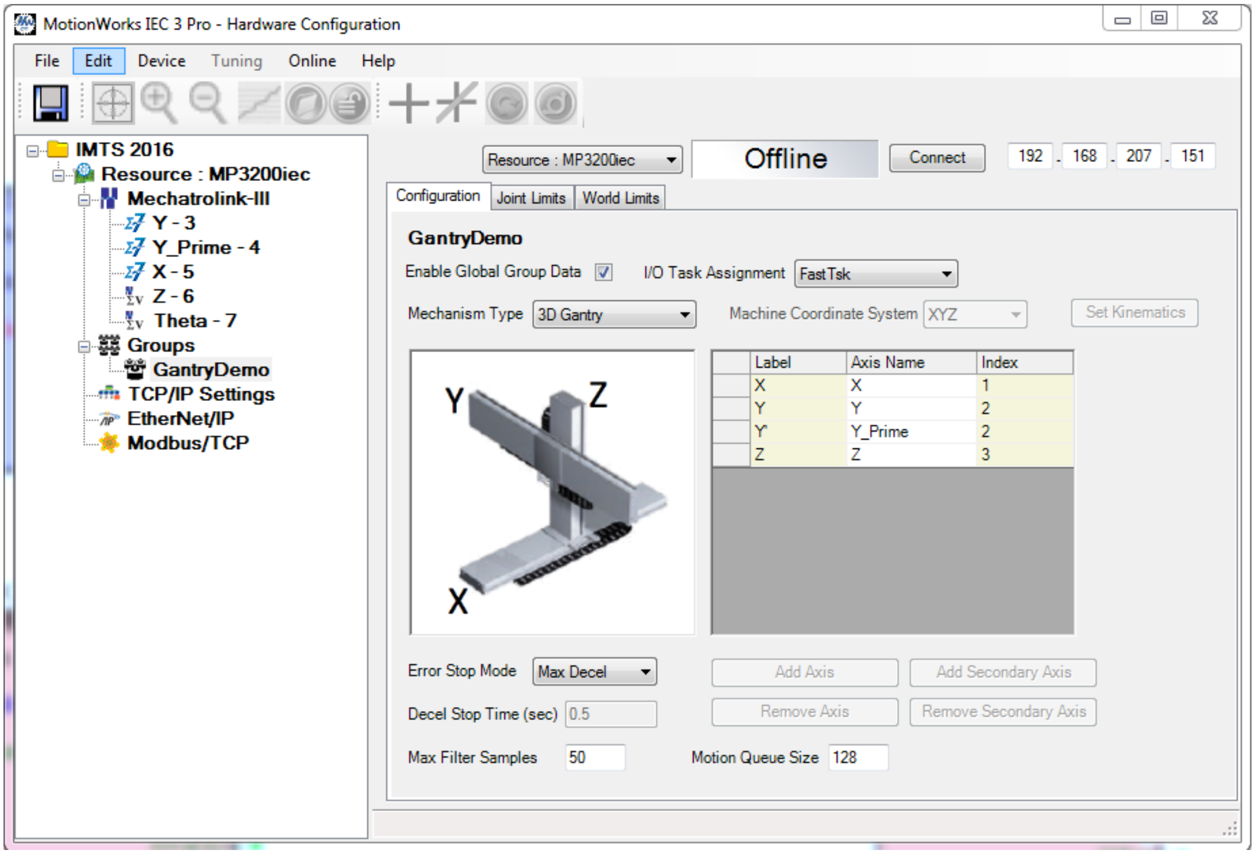
Some errors listed below are internal errors that should never occur, such as 10115 and 10128. Please report these errors to Yaskawa America Motion Applications if they occur.

ErrorID	Meaning
0	No error.
2	Maximum number of files are already open.
4	File is already opened.
5	File is write protected or access is denied.
6	File name not defined.
10115	XML Tag not found. Possibly the file is corrupt or the schema is not compatible with this function block.
10120	File could not be opened. Check for accurate directory path and use of "/"
10122	Row Error. The data is out of sync with the expected row / column arrangement expected.
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.
10125	Conversion Error. Check the ErrorRow and ErrorCol / ErrorString outputs for details.
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.
10165	CommandString length is too long or command delimiter not found.
10168	Buffer Size Error.

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.



GantryDemo

Enable Global Group Data ☑ I/O Task Assignment: FastTsk

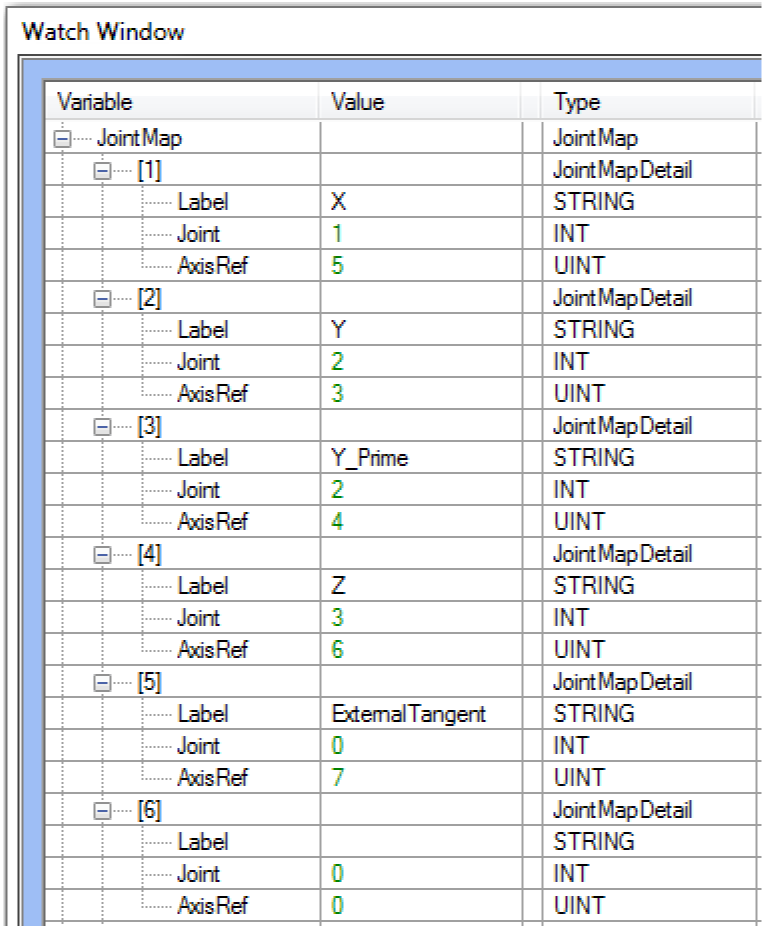
Mechanism Type: 3D Gantry Machine Coordinate System: XYZ

Label	Axis Name	Index
X	X	1
Y	Y	2
Y'	Y_Prime	2
Z	Z	3

Error Stop Mode: Max Decel

Decel Stop Time (sec): 0.5

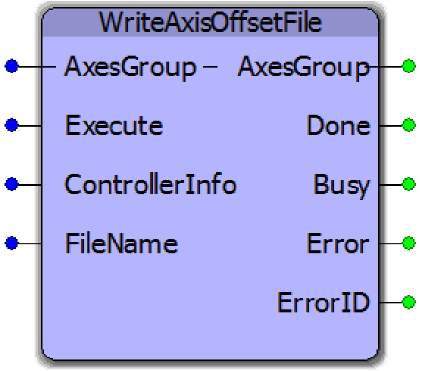
Max Filter Samples: 50 Motion Queue Size: 128



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

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 substructures 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	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.	All zeros in structure
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.	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:

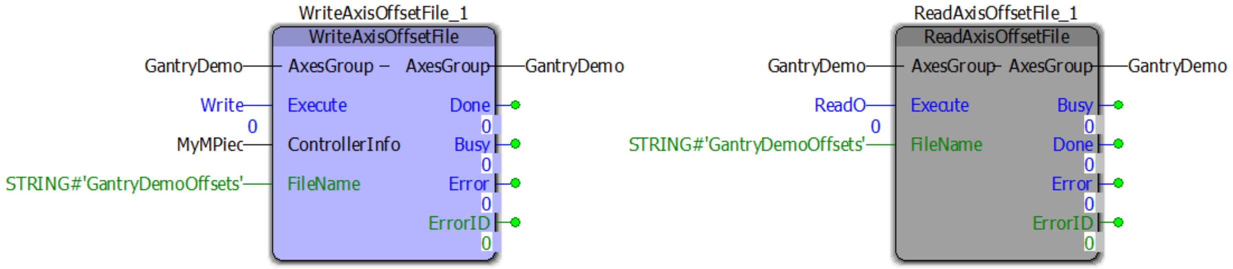
- Firmware version 3.3.0 is required to use this function. It relies on MC_ReadParameter 1838, which was added for version 3.3.0.
- The Flash/Local directory was also added for firmware 3.3.0. It is not deleted when a project archive is deleted or the controller is restored to factory defaults.
- It 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

ErrorID	Meaning
2	Maximum number of files are already open.
4	File is already opened.
5	File is write protected or access is denied.
6	File name not defined.
4378	The function block is not applicable for the external axis specified.
4392	The function block can not be used with an inverter axis.
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 POUs.
4648	The parameter number does not exist for the specified axis.
8960	Invalid axes group. Confirm that the AxesGroup variable has the correct %M address as automatically assigned by the Hardware Configuration.
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.
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.
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.
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.
10166	File Not Found
10168	Buffer Size Error.
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
10619	Invalid file name. File names must only contain alphanumeric characters. The first character must not be numeric.

Example



Math Toolbox

Math Revision History

Current Version:

(********* 2017-01-15 v331 released with Toolbox Installer - Jan 2017 Collection *************)**

RECT_TO_POLAR - New FB added. Support for Tool Compensation in Group Toolbox.

Previous Versions:

(************** 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 *****************************)**

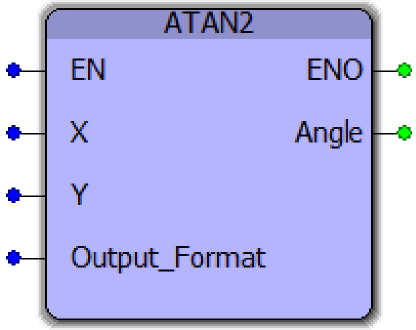
Upgraded to version 2.0 Project for MotionWorks IEC. Built from Math Toolbox v004beta .

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*,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

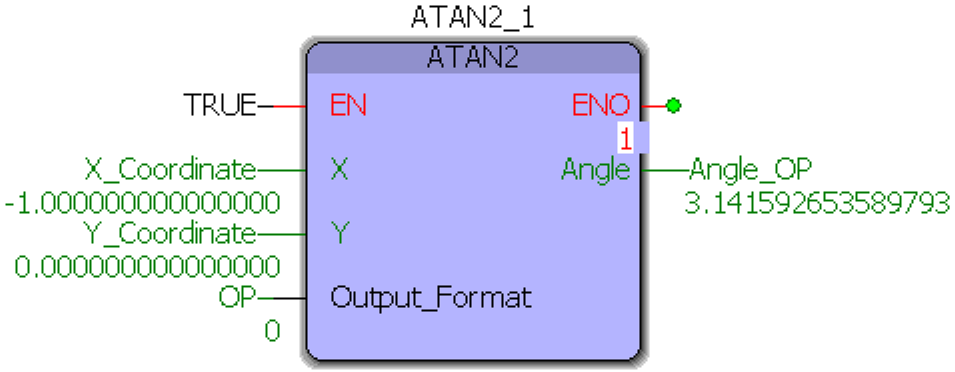
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	EN	BOOL	This function will continue to calculate the ATAN2 result while EN is held high.	FALSE
V	X	LREAL	X coordinate	LREAL#0.0
V	Y	LREAL	Y coordinate	LREAL#0.0
V	Output_ Format	INT	Format of the output value. 0: radians (-pi, pi] 1: radians [0, 2*pi) 2: degrees [0°, 360°)	INT#0
VAR_OUTPUT				
B	ENO	BOOL	High if the function is executing normally.	
V	Angle	LREAL	The result of the ATAN2 calculation.	

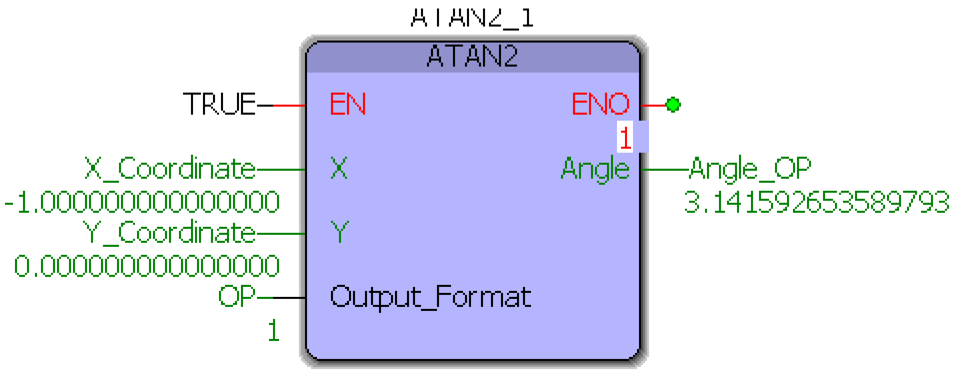
Notes

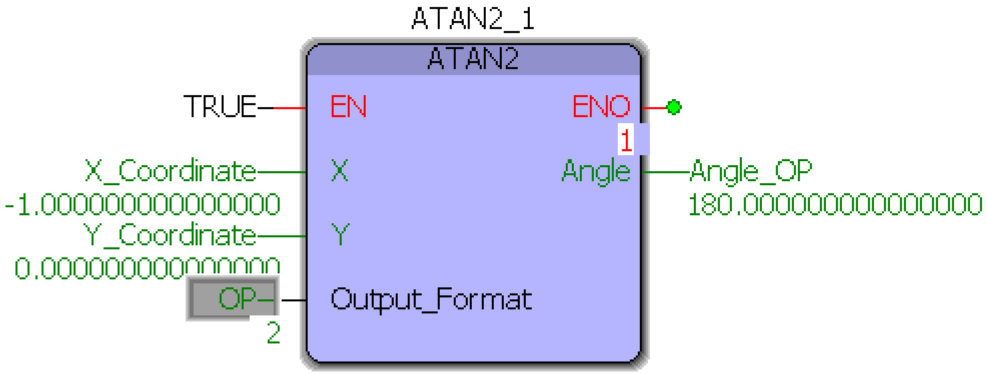
This is a function, not a function block and only provides one output. If ENO is not high when EN is high, this function cannot calculate the Angle.

Example

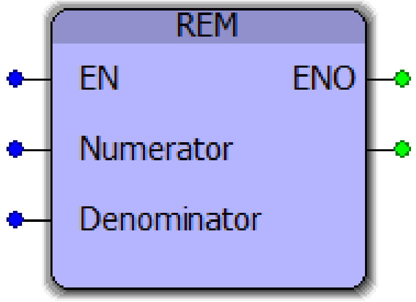
ATAN2 used with various output formats:







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

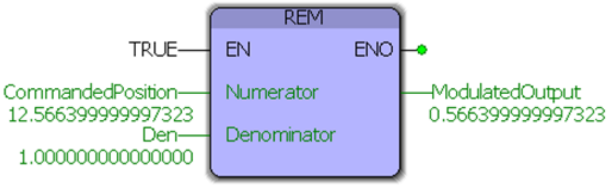
This is a function, not a function block and only provides one output. If ENO is not high when EN is high, this function cannot calculate the remainder. Verify that the Denominator is not zero.

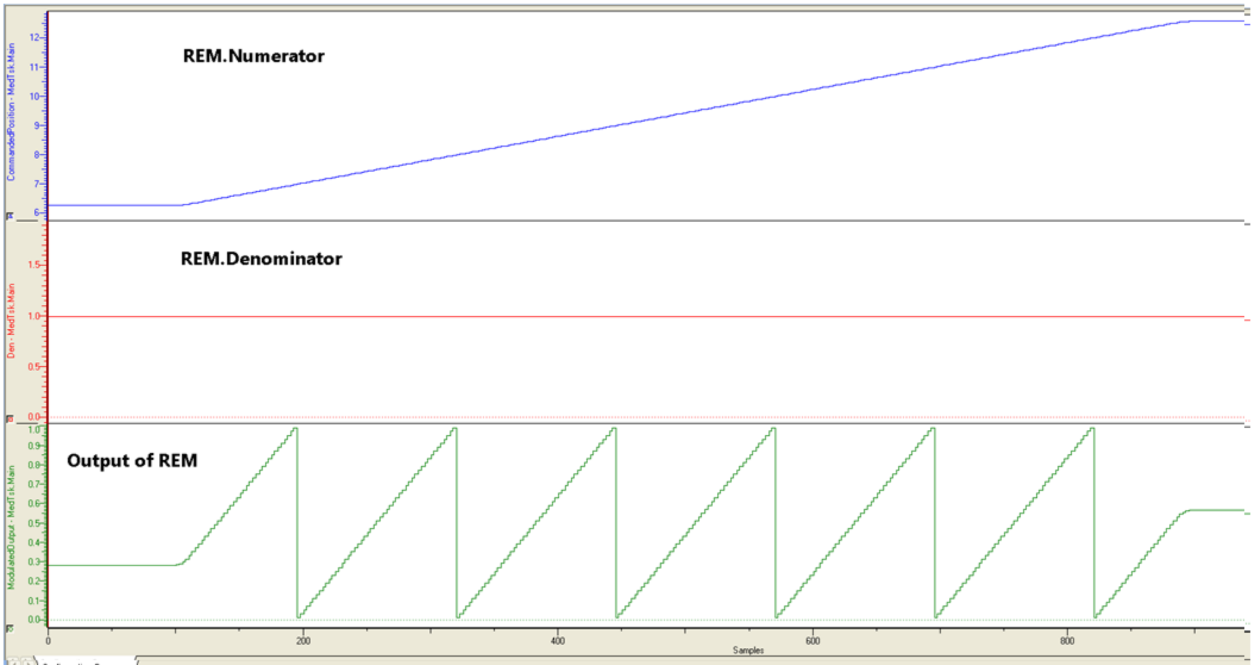
Example 1 - Structured Text

```
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);
```

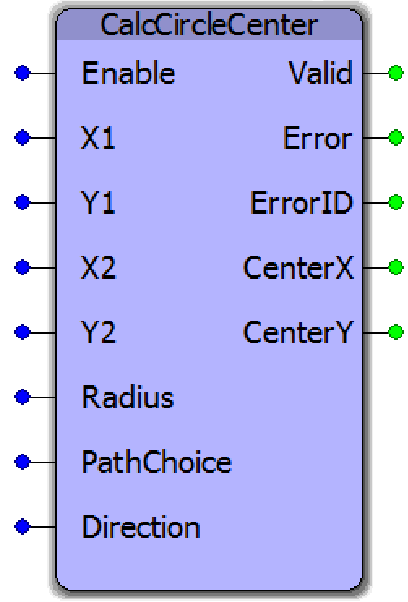
```
    Duration:=RegistrationData.SensorDistance - ((ControlData.EndSyncPosition - ControlData.StartSyncPosition) / LREAL#2.0) - (ActualPositionNonCyclic - RegistrationData.BufferNonCyclic[TempUsePointer]);
ELSE
    (*  These calculations are designed for reciprocating cam profiles (Slave net change = 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

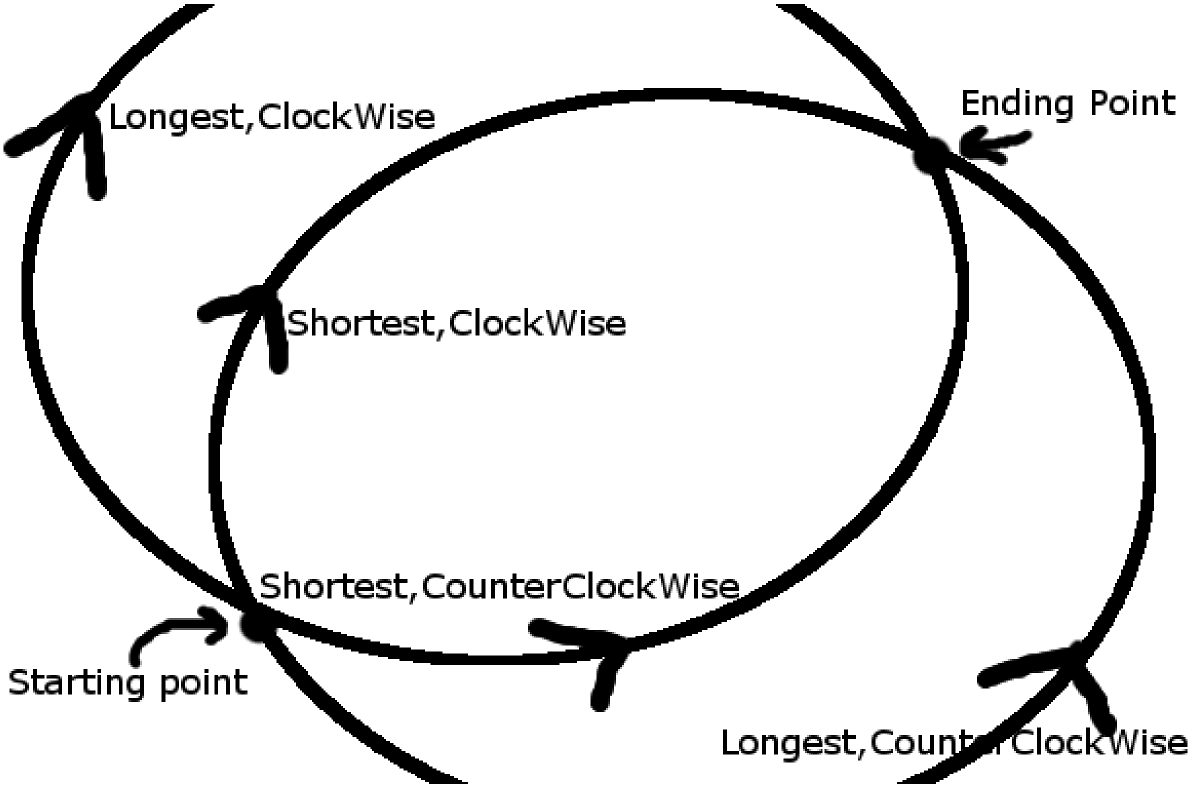
Math Toolbox

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
U	ENABLE	BOOL	This function will continue to calculate the center of the circle while this is enabled	FALSE
U	X1	LREAL	X coordinate of starting point	LREAL#0.0
U	Y1	LREAL	Y coordinate of starting point	LREAL#0.0
U	X1	LREAL	X coordinate of ending point	LREAL#0.0
U	Y1	LREAL	Y coordinate of ending point	LREAL#0.0
U	Radius	LREAL	Radius of the circle	LREAL#0.0
U	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
U	Direction	MC_CirclePathChoice	If the circle path travels clockwise use MC_CirclePathChoice#Clockwise. Otherwise if	MC_CirclePathChoice#ClockWise

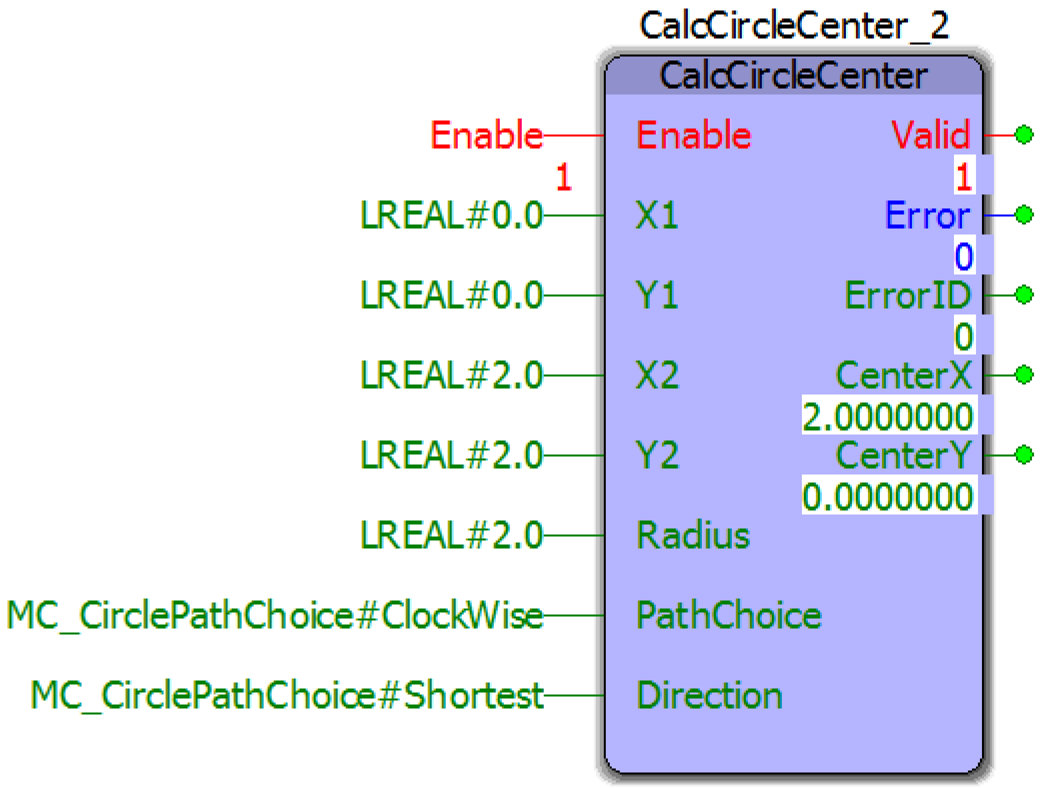
			the circle path travels counterclockwise use MC_CirclePathChoice#CounterClockWise	
VAR_OUTPUT				
C	Valid	BOOL	Displays if inputs are valid	
C	Error	LREAL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
C	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
C	CenterX	LREAL	X coordinate of the center of the circle	
C	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


CalcCircleCenter_2
CalcCircleCenter
Enable
1
LREAL#0.0
LREAL#0.0
LREAL#2.0
LREAL#2.0
LREAL#2.0
MC_CirclePathChoice#ClockWise
MC_CirclePathChoice#Shortest
Enable
X1
Y1
X2
Y2
Radius
PathChoice
Direction
Valid
1
Error
0
ErrorID
0
CenterX
2.0000000
CenterY
0.0000000

PackML Toolbox

Requirements for v302

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:

-
- DataTypes Toolbox (v300 or higher)
- Math_Toolbox (v300 or higher)
- 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:

*************************************** **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 supressed 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.

Previous Versions:

*************************************** **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 aad 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

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.

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

[[[Undefined variable Primary.Asterisk_DT]]]	Element	Data Type	Description	Usage
	MyEquipmentModule_STRUCT	**EquipmentModule_STRUCT**		
U	EnabledCMs	[[[Undefined variable Primary.DataType_WORD]]]	Number of enabled Control Modules contained in the Equipment Module	MyEquipmentModule_STRUCT.EnabledCMs
U	CMs_Active	[[[Undefined variable Primary.DataType_WORD]]]	Every bit in this word indicates if a control module is active	MyEquipmentModule_STRUCT.CMs_Active
U	CMs_NotDone	[[[Undefined variable Primary.DataType_WORD]]]	Every bit in this word indicates if a control module is done	MyEquipmentModule_STRUCT.CMs_NotDone
U	CM_InactiveMask	[[[Undefined variable Primary.DataType_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	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Reset the machine	MyEquipmentModule_STRUCT.Cmd_Reset
U	Sts_Resetting_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the resetting state	MyEquipmentModule_STRUCT.Sts_Resetting_SC
U	Cmd_Start	[[[Undefined variable	Command to Start the	MyEquipmentModule_

		Primary.DataType_BOOL]]]	machine	STRUCT.Cmd_Start
U	Sts_Starting_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Starting state	MyEquipmentModule_STRUCT.Sts_Starting_SC
U	Cmd_Stop	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Stop the machine	MyEquipmentModule_STRUCT.Cmd_Stop
U	Sts_Stopping_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Stopping state	MyEquipmentModule_STRUCT.Sts_Stopping_SC
U	Cmd_Hold	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Hold the machine	MyEquipmentModule_STRUCT.Cmd_Hold
U	Sts_Holding_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Holding state	MyEquipmentModule_STRUCT.Sts_Holding_SC
U	Cmd_UnHold	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Unhold the machine	MyEquipmentModule_STRUCT.Cmd_UnHold
U	Sts_UnHolding_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the UnHolding state	MyEquipmentModule_STRUCT.Sts_UnHolding_SC
U	Cmd_Suspend	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Suspend the machine	MyEquipmentModule_STRUCT.Cmd_Suspend
U	Sts_Suspending_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Suspending state	MyEquipmentModule_STRUCT.Sts_Suspending_SC
U	Cmd_UnSuspend	[[[Undefined variable Primary.DataType_BOOL]]]	Command to UnSuspend the machine	MyEquipmentModule_STRUCT.Cmd_UnSuspend
U	Sts_UnSuspending_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the UnSuspending state	MyEquipmentModule_STRUCT.Sts_UnSuspending_SC
U	Cmd_Abort	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Abort the machine	MyEquipmentModule_STRUCT.Cmd_Abort
U	Sts_Aborting_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Aborting state	MyEquipmentModule_STRUCT.Sts_Aborting_SC
U	Cmd_Clear	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Clear the machine	MyEquipmentModule_STRUCT.Cmd_Clear
U	Sts_Clearing_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Clearing state	MyEquipmentModule_STRUCT.Sts_Clearing_SC
U	Sts_Executing_SC	[[[Undefined vari-	When set,	MyEquip-

		able Primary.DataType_BOOL]]]	the machine is in the Executing state	mentModule_STRUCT.Sts_Executing_SC
U	Cmd_StateComplete	[[[Undefined variable Primary.DataType_BOOL]]]	Command to enter the Completing State	MyEquipmentModule_STRUCT.Cmd_StateComplete
U	Sts_Completing_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Completing state	MyEquipmentModule_STRUCT.Sts_Completing_SC
U	ModuleActive	[[[Undefined variable Primary.DataType_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

[[[Undefined variable Primary.Asterisk_DT]]]	Element	Data Type	Description	Usage
	MyIngredient_STRUCT	**Ingredient_STRUCT**		
U	ID	[[[Undefined variable Primary.DataType_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

[[[Undefined variable Primary.Asterisk_DT]]]	Element	Data Type	Description	Usage
	MyLimit_STRUCT	**Limit_STRUCT**		
U	ID	[[[Undefined variable Primary.DataType_INT]]]	User defined ID for the limit, 0000 reserved for no limit assigned	MyLimit_STRUCT.ID
U	Name	[[[Undefined variable Primary.DataType_STRING]]]	Literal name for the limit	MyLimit_STRUCT.Name
U	Unit	STRING_5	Unit of the limit value	MyLimit_STRUCT.Unit
U	Value	[[[Undefined variable Primary.DataType_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

[[[Undefined variable Primary.Asterisk_DT]]]	Element	Data Type	Description	Usage
	MyNode_ STRUCT	**Node_STRUCT**		
U	Number	[[[Undefined variable Primary.DataType_INT]]]	A chosen unique number of the Upstream/Downstream PackML machine	MyNode_STRUCT.Number
U	ControlCmdNumber	[[[Undefined variable Primary.DataType_INT]]]	User defined command to be sent from one node on the network to another	MyNode_STRUCT.ControlCmdNumber
U	CmdValue	[[[Undefined variable Primary.DataType_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

[[[Undefined variable Primary.Asterisk_DT]]]	Element	Data Type	Description	Usage
	MyPackML_Commands_STRUCT	**PackML_Commands_STRUCT**		
U	Mode	[[[Undefined variable Primary.DataType_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	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Reset the Machine	MyPackML_Commands_STRUCT.Reset
U	Start	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Start the Machine	MyPackML_Commands_STRUCT.Start
U	Stop	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Stop the Machine	MyPackML_Commands_STRUCT.Stop
U	Hold	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Hold the Machine	MyPackML_Commands_STRUCT.Hold
U	UnHold	[[[Undefined variable Primary.DataType_BOOL]]]	Command to UnHold the Machine	MyPackML_Commands_STRUCT.UnHold
U	Suspend	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Suspend the Machine	MyPackML_Commands_STRUCT.Suspend
U	UnSuspend	[[[Undefined variable Primary.DataType_BOOL]]]	Command to UnSuspend the Machine	MyPackML_Commands_STRUCT.UnSuspend
U	Abort	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Abort the Machine	MyPackML_Commands_STRUCT.Abort

U	Clear	[[[Undefined variable Primary.DataType_BOOL]]]	Command to Clear the Machine	MyPackML_Commands_STRUCT.Clear
U	StateComplete	[[[Undefined variable Primary.DataType_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.	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.	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.	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.	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.	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.	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.	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.	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.	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.	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 module is active to receive commands.	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	MyPackTags_Admin_STRUCT.ProdProcessed

			the machine’s counters and timers were reset.	
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 reworkable 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	UnitModeChangeInProcess	BOOL	[1 = Requested unit mode change in process]	MyPackTags_Status_STRUCT.UnitModeChangeInProcess
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	ProcModeChangeInProcess	BOOL	[1 = Requested procedure mode change in process]	MyPackTags_Status_STRUCT.ProcModeChangeInProcess
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	StateChangeInProcess	BOOL	[1 = Requested state change in process]	MyPackTags_Status_STRUCT.StateChangeInProcess
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	MyPackTags_Status_STRUCT.Clean

			after a cleaning cycle and reset once production begins again	
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

[[[Undefined variable Primary.Asterisk_DT]]]	Element	Data Type	Description	Usage
	MyUNitmachine_STRUCT	**UNitmachine_STRUCT**		
U	PackML_StateControlReady	[[[Undefined variable Primary.DataType_BOOL]]]	Indicates when the PackML_State_Diagram function block is ready to control the machine	MyUNitmachine_STRUCT.PackML_StateControlReady
U	EnabledEMs	[[[Undefined variable Primary.DataType_INT]]]	Number of enabled equipment modules in the machine	MyUNitmachine_STRUCT.EnabledEMs
U	EMs_Active	[[[Undefined variable Primary.DataType_WORD]]]	Every bit in this word indicates which equipment modules are Active	MyUNitmachine_STRUCT.EMs_Active
U	EMs_NotDone	[[[Undefined variable Primary.DataType_WORD]]]	Every bit in this word indicates which equipment modules are Not Done	MyUNitmachine_STRUCT.EMs_NotDone
U	EM_InactiveMask	[[[Undefined variable Primary.DataType_WORD]]]	Every bit in this word indicates which equipment modules are Inactive	MyUNitmachine_STRUCT.EM_InactiveMask
U	EM	[[[Undefined variable Primary.DataType_EquipmentModule_Array]]]	Array containing the Commands, Status and Active bits for the 16 Equipment Modules contained in the Machine	MyUNitmachine_STRUCT.EM
U	Sts_Resetting_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the resetting state	MyUNitmachine_STRUCT.Sts_Resetting_SC
U	Sts_Starting_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Starting state	MyUNitmachine_STRUCT.Sts_Starting_SC
U	Sts_Stopping_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Stopping state	MyUNitmachine_STRUCT.Sts_Stopping_SC

U	Sts_Holding_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Holding state	MyUNitmachine_STRUCT.Sts_Holding_SC
U	Sts_UnHolding_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the UnHolding state	MyUNitmachine_STRUCT.Sts_UnHolding_SC
U	Sts_Suspending_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Suspending state	MyUNitmachine_STRUCT.Sts_Suspending_SC
U	Sts_UnSuspending_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the UnSuspending state	MyUNitmachine_STRUCT.Sts_UnSuspending_SC
U	Sts_Aborting_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Aborting state	MyUNitmachine_STRUCT.Sts_Aborting_SC
U	Sts_Clearing_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Clearing state	MyUNitmachine_STRUCT.Sts_Clearing_SC
U	Sts_Executing_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Executing state	MyUNitmachine_STRUCT.Sts_Executing_SC
U	Sts_Completing_SC	[[[Undefined variable Primary.DataType_BOOL]]]	When set, the machine is in the Completing state	MyUNitmachine_STRUCT.Sts_Completing_SC

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Enumerated Types for PackML Toolbox

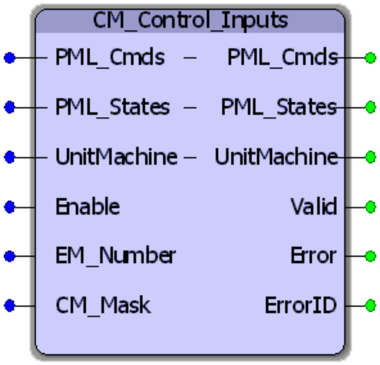
Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to zero-based integers (INT). Therefore, the first value equates to zero, the second to 1, etc. The format for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

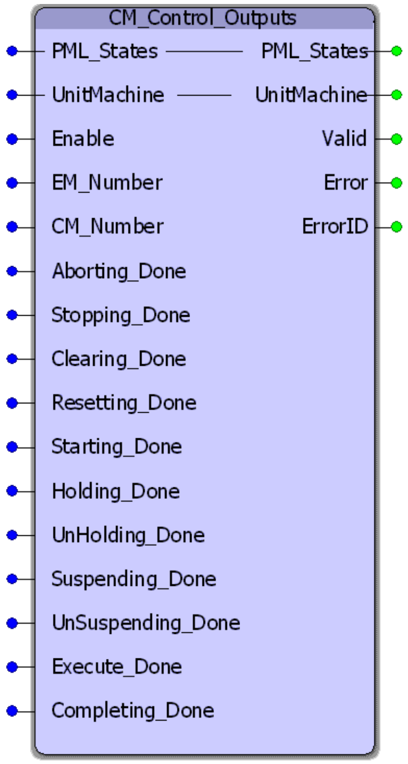
Notes

- See the PackML template documentation for further details on recommended usage.

Error Description

ErrorID	Meaning
0	No Error
12560	Invalid Equipment Module number.
12561	Equipment Module not enable in the system.
12562	Invalid number of enabled Control Modules in selected Equipment Module.

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.	

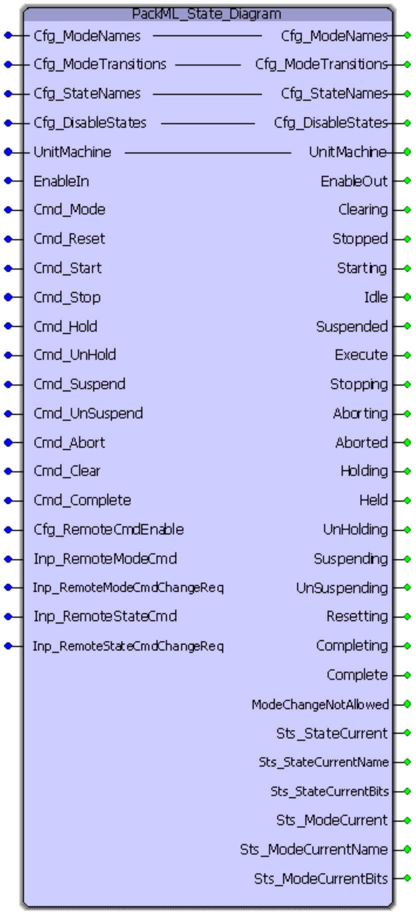
Notes

- See template documentation for further details on recommended usage.

Error Description

ErrorID	Meaning
0	No Error
12560	Invalid Equipment Module number.
12561	Equipment Module not enable in the system.
12562	Invalid number of enabled Control Modules in selected Equipment Module.

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				**Default**
B	EnableIn	BOOL	The function will continue to execute while the enable is held high	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	FALSE

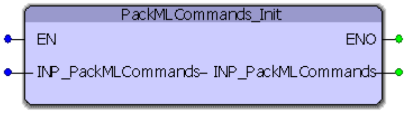
			command will be ignored	
V	Cmd_Clear	BOOL	Setting this bit sends the ‘Clear’ command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
V	Cmd_Complete	BOOL	Setting this bit sends the ‘Complete’ command to all enabled and 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

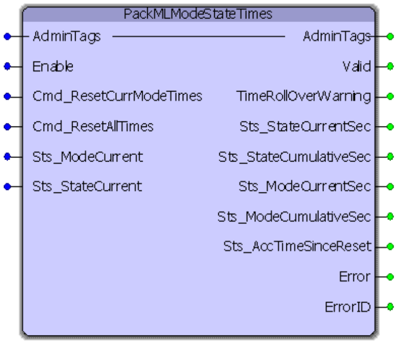
Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	INP_PackMLCommands	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_StateCu-	StateCumulativeArray	An array containing the times spent operating in different modes	

	mulativeSec		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.

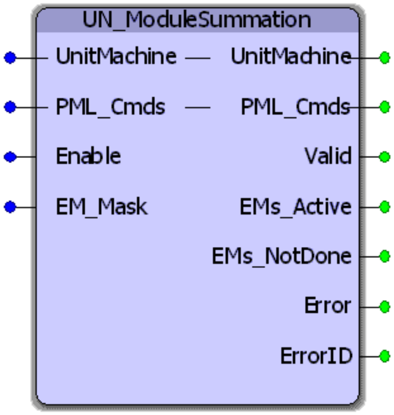
Notes

- See the PackML template documentation for further details on recommended usage.

Error Description

ErrorID	Meaning
0	No Error
12563	Time rollover warning.

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_Commands_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

ErrorID	Meaning
0	No Error

PLCopen Toolbox

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Getting Started with PLCopen Toolbox

Requirements for v330

To use the PLCopen Toolbox, your 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 (v300 or higher)
- Math_Toolbox (v300 or higher)
  - Only required if using the Jog_To_Position or ProductBuffer function block.
- Yaskawa Toolbox (v300 or higher)
  - Only required if using the Full_Closed_Control 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 form 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

New for PLCopen v205 and higher – 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 PLCopen Toolbox, you must do the following:
 1) Include the DataTypes Toolbox in your project.
 2) Remove any other Yaskawa supplied datatype files with firmware library definitions such as
   a. ControllInfoTypes
   b. YDeviceCommTypes

Current Version:

(****************************** 2016-10-31 v330 released ***********************************)**

1) ReadMotorSpeed - New FB added. - Known issue: This function block only works for Sigma 5 motors.

Previous Versions:

(****************************** 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 simultanious 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 MP2000 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 outptus 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: 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/sec2 as defined in the Hardware Configuration.	MyAxisStruct.Acceleration
U	Deceleration	LREAL	In user units/sec2 as defined in the Hardware Configuration.	MyAxisStruct.Deceleration
U	Jerk	LREAL	In user units/sec3 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. Note: For the Home_TouchProbe function block, this element is not used. Set a positive or negative ApproachDistanceLimit.	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 dis-	MyHomeStruct.CreepDistanceLimit

			tance the axis will travel in search of the C channel before aborting the homing operation and issuing an error.	
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/sec2 as defined in the Hardware Configuration.	MyMoveStruct.Acceleration
U	Deceleration	LREAL	In user units/sec2 as defined in the Hardware Configuration.	MyMoveStruct.Deceleration
U	Jerk	LREAL	In user units/sec3 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 is 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, thenext 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 function block.

Data Type Declaration

*	Element	Data Type	Description	Usage
	MyProductBufferStruct	**ProductBufferStruct**		
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	MyProductBufferStruct.SensorOffset

			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.	
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	MyProductBufferStruct.LastAction

			basic StorePointer and UsePointer method.	
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]

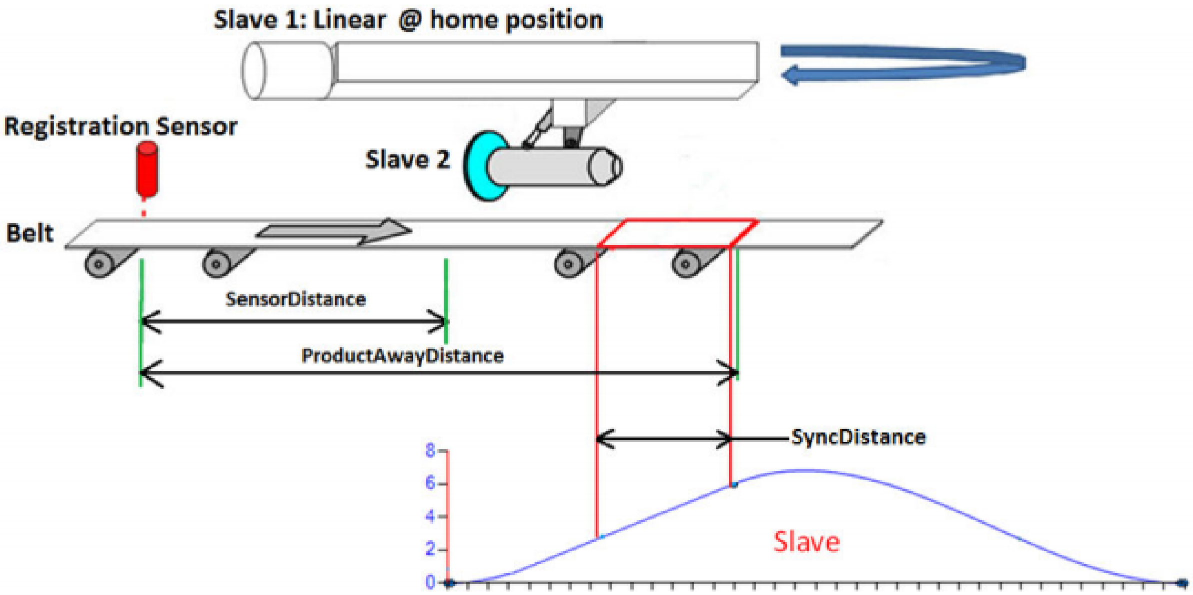
Notes:

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

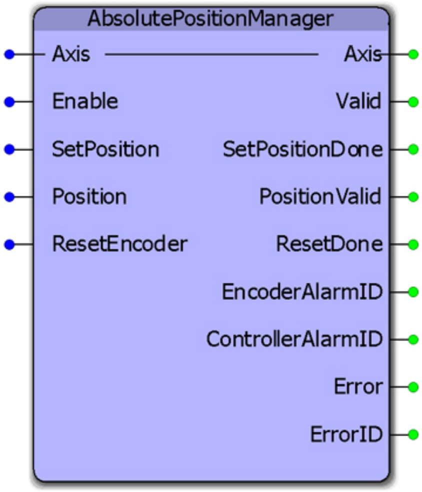
Some blocks accept an enumerated type (ENUM), which is a keyword (or constant) representing a value which will configure the operation of the function block. Enumerated types are equivalent to zero-based integers (INT). Therefore, the first value equates to zero, the second to 1, etc. The format for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

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

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.	

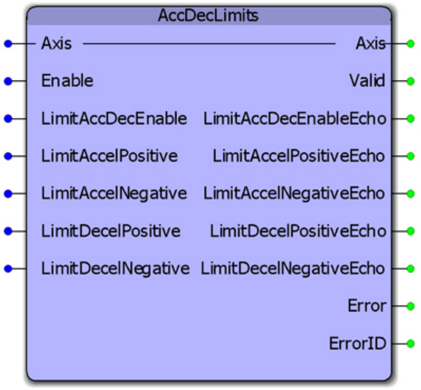
Notes

- To clear the absolute encoder alarm from a servopack, the user must use the 'ResetEncoder' input, then reset the servopack by either cycling power 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

ErrorID	Meaning
4378	The function block is not applicable for the external axis specified.
4380	MC_SetPosition cannot be executed while the axis is already moving.
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.
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.
4401	The controller cannot communicate with the drive. It may be disconnected from the MECHATROLINK network.
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 POUs.
4646	Mode does not correspond to a valid enumeration value.
45335	Failed to initialize absolute encoder.
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.

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

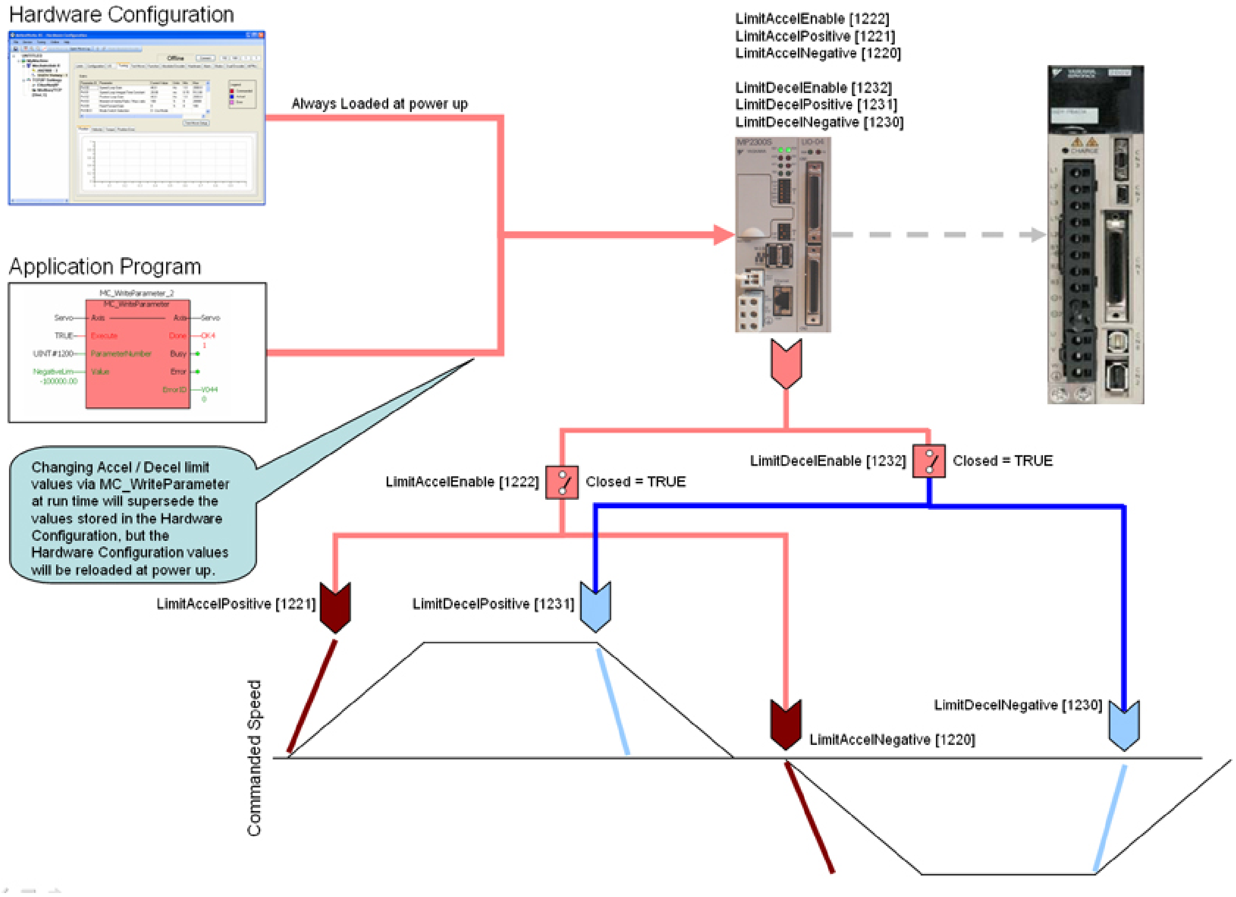
Parameters

*	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				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	LimitAccDecEnable	BOOL	Enables or Disables the Limit Accel Decel function. Parameter 1222 and 1232 are combined	FALSE
V	LimitAccelPositive	LREAL	Parameter 1221	LREAL#0.0
V	LimitAccelNegative	LREAL	Parameter 1220	LREAL#0.0
V	LimitDecelPositive	LREAL	Parameter 1231	LREAL#0.0
V	LimitDecelNegative	LREAL	Parameter 1230	LREAL#0.0
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
Application Program
MC_WriteParameter_2
MC_WriteParameter
Servo
Axis
Axis
Servo
TRUE
Execute
Done
OK4
1
UINT#1200
ParameterNumber
Busy
NegativeLim
-100000.00
Value
Error
ErrorID
V044
0
Always Loaded at power up
LimitAccelEnable [1222]
LimitAccelPositive [1221]
LimitAccelNegative [1220]
LimitDecelEnable [1232]
LimitDecelPositive [1231]
LimitDecelNegative [1230]
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.
LimitDecelEnable [1232]
Closed = TRUE
LimitAccelEnable [1222]
Closed = TRUE
LimitAccelPositive [1221]
LimitDecelPositive [1231]
LimitDecelNegative [1230]
LimitAccelNegative [1220]
Commanded Speed

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

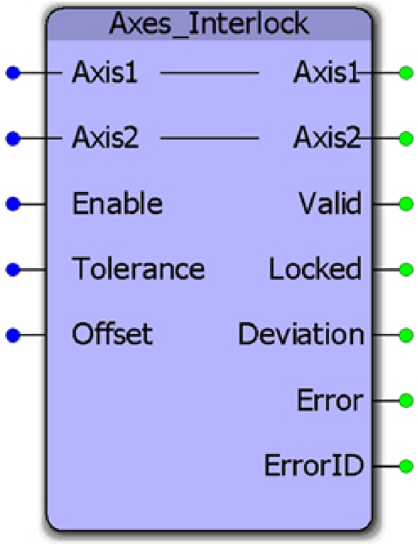
- Deceleration is defined by decreasing velocity towards zero.

- 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

ErrorID	Meaning
0	No error.
4378	The function block is not applicable for the external axis specified.
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 POUs.
4648	The parameter number does not exist for the specified axis.
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.
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.

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.

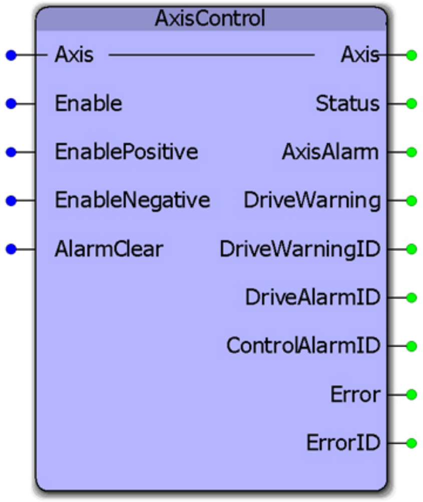
Notes

- 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

ErrorID	Meaning
0	No error
4378	The function block is not applicable for the external axis specified
4625	Axis ID 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 AXIS_REF is properly declared as a VAR or VAR_GLOBAL in all relevant POUs. Troubleshooting info on our Youtube channel.
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.

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				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
E	EnablePositive	BOOL	Not Supported	FALSE
E	EnableNegative	BOOL	Not Supported	FALSE
V	AlarmClear	BOOL	Clears axis related alarms using MC_Reset.	FALSE
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.
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

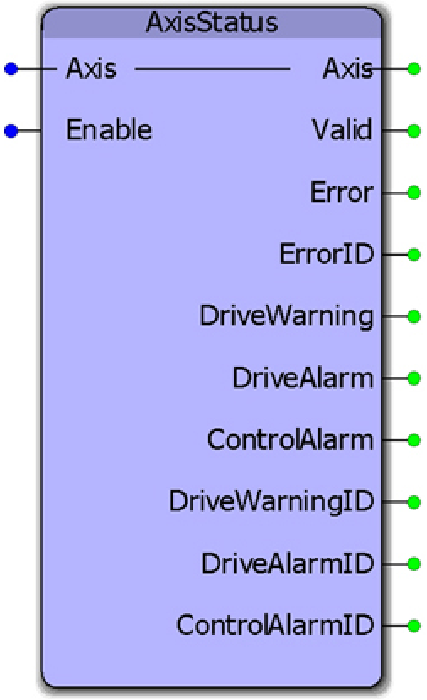
- When attempting to clear an alarm, the enable input must be FALSE or the alarm reset function will be blocked from executing.
- We recommend viewing the alarm and warning output ID’s in Hex, because all Yaskawa ServoPack documentation lists the amplifier alarm codes in Hex. This simplifies alarm identification. Note that MotionWorks IEC may show the value at the output in decimal. For example, a DriveAlarmID 0f 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

ErrorID	Meaning
0	No error.
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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
4414	MECHATROLINK communications to the drive was disrupted. Execute MC_Reset to restore the connection.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
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.
45332	Sending clear alarms command to servo drive failed.
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.
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.

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.

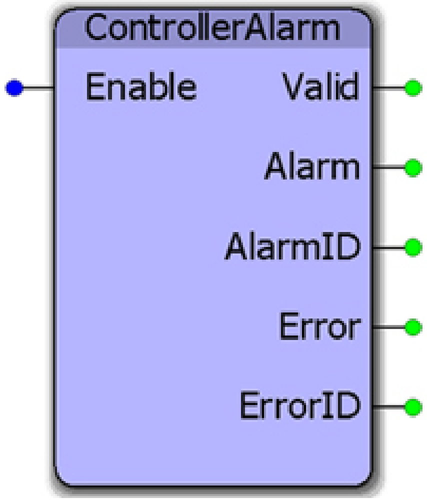
Notes

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

ErrorID	Meaning
0	No error.
4378	The function block is not applicable for the external axis specified.
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 POUs.
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.

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	
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	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.	

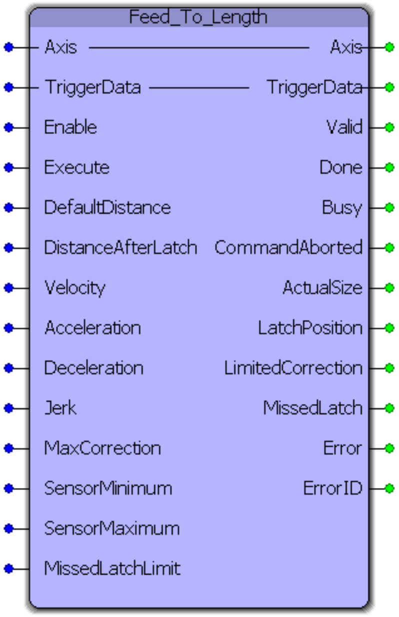
Notes

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

ErrorID	Meaning
0	No 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 POUs.

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	FALSE

			input.	
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	*Jerk*	*LREAL*	*[[[Undefined variable Primary.ParameterNotSupported]]] Value of the jerk in [user units / second^3].*	
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 positons.	
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.

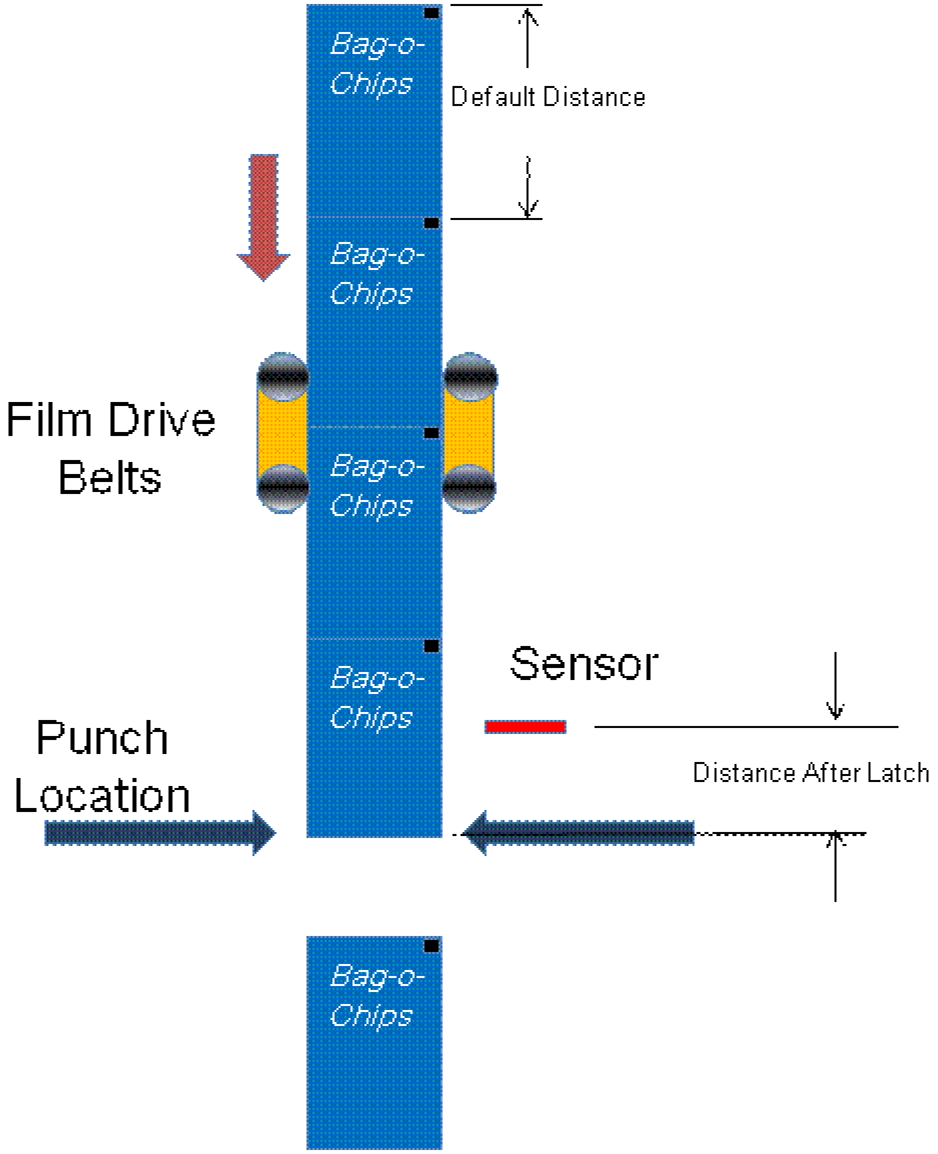
Error Description

ErrorID	Meaning

0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
4391	The function block cannot be used with a virtual axis.
4396	Axis latch function already in use.
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.
4406	Continuous Latch Mode is not supported on Sigma II, Sigma III, or external encoders.
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 POUs.
4630	Trigger reference is not valid.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4648	The parameter number does not exist for the specified axis.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4676	The time value must be within 0 to 10 MECHATROLINK cycles.
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.
10020	ProductSize cannot be less than or equal to zero.
10021	Maximum allowed consecutive missed registration marks reached.
10025	SensorMinimum must be less than SensorMaximum.
10053	DataPoint Error.
57617	Instance object is NULL.
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.

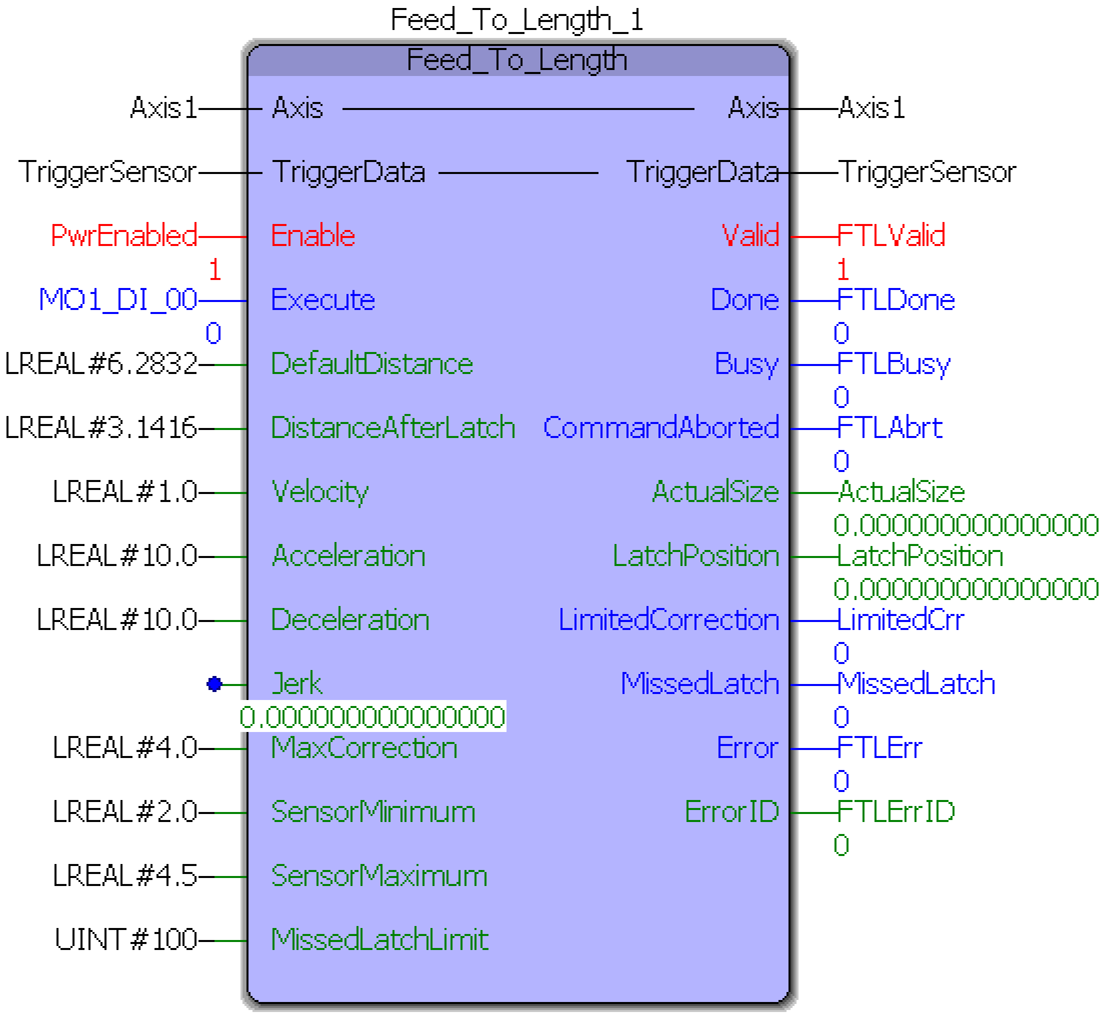
Example

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.

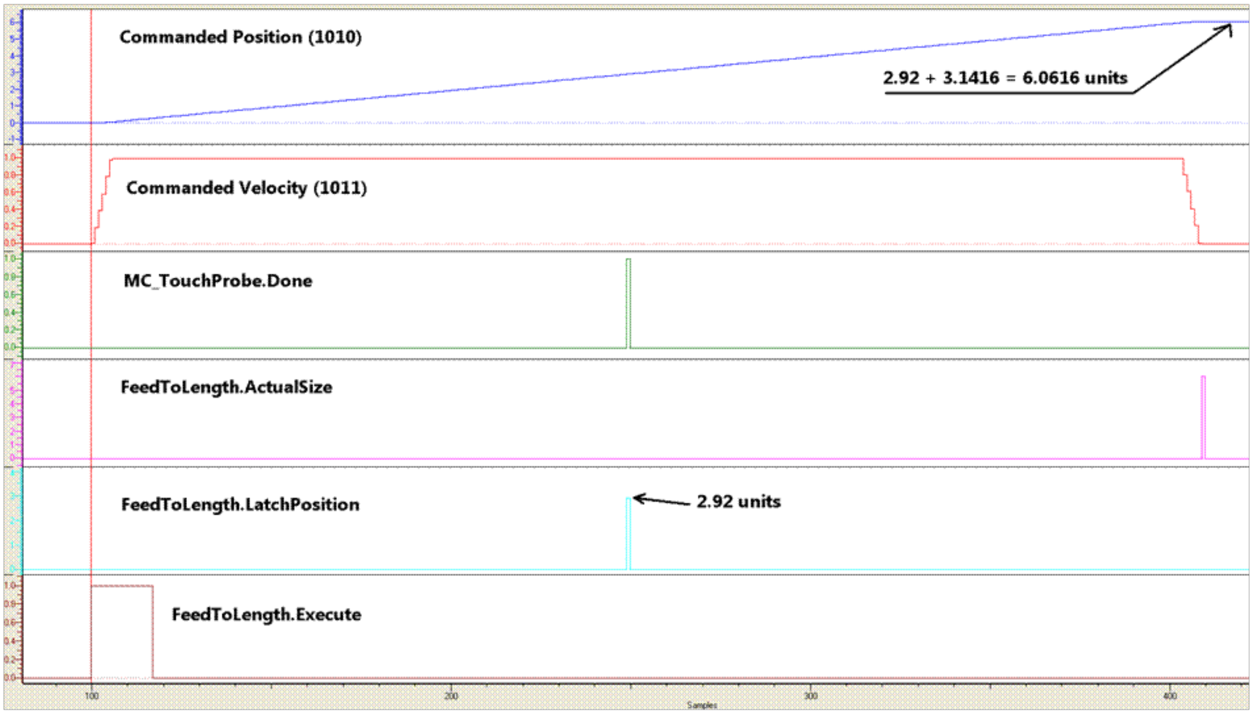


MaxCorrection limits the correction if an erroneous registration mark is captured and the calculation results in a large correction distance.

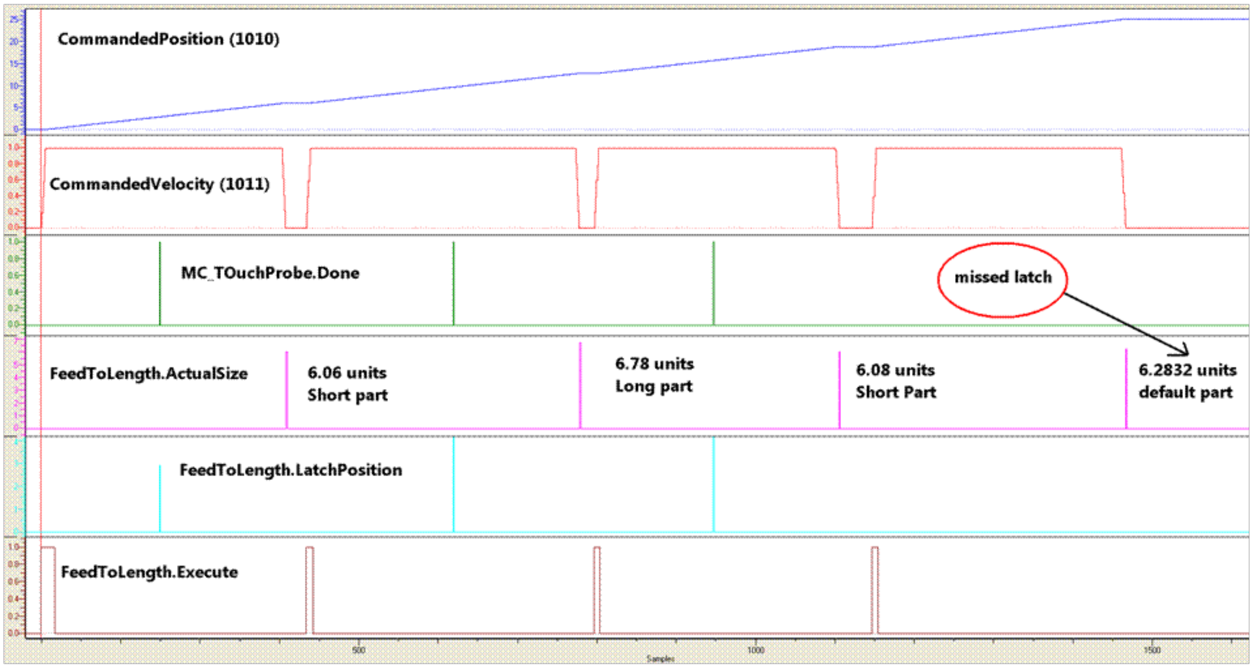
SensorMinimum and SensorMaximum provide window in which a registration mark must be detected to be considered a valid registration mark. In this example, the mark is expected around 3.1416 units, and only marks detected between 2.0 to 4.5 are accepted. Set the window as small as appropriate for the application.



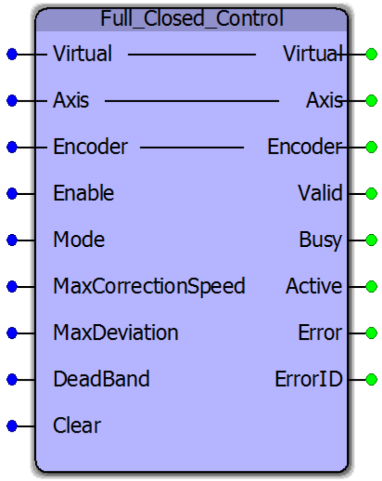
The FeedToLength function block will position the axis exactly 3.1416 units (DistanceAfterLatch) after the registration mark was detected.



The FeedToLength 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	LREAL#0.0

			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.	
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

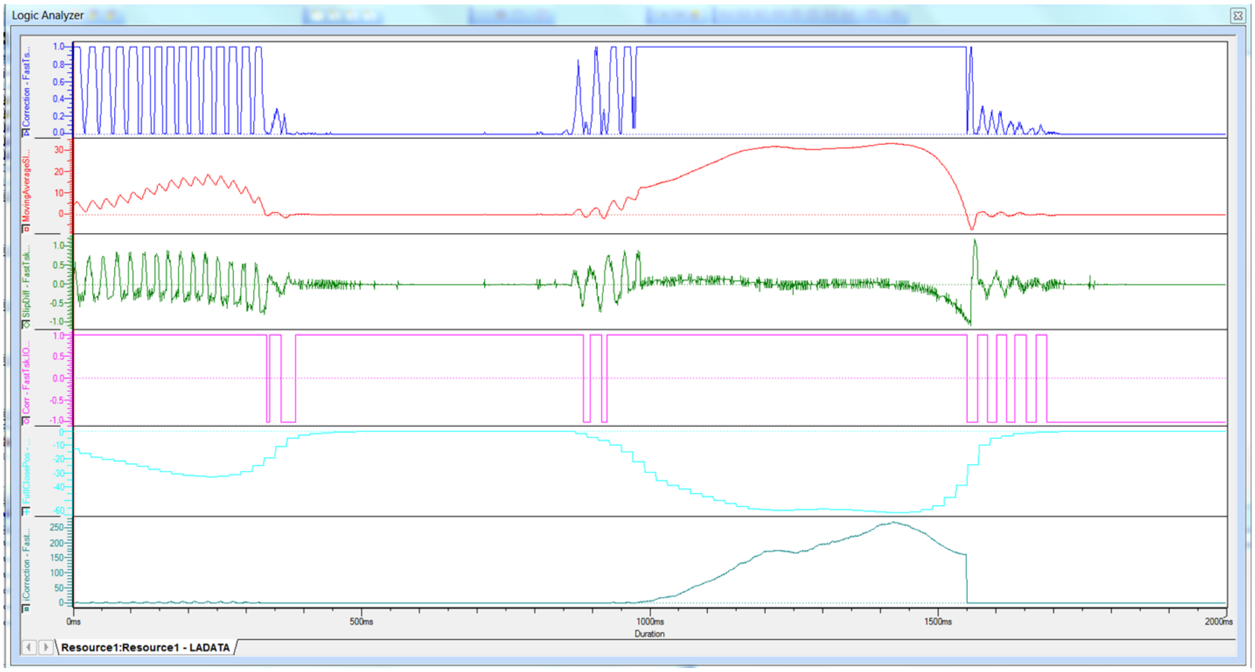
the machine is being set up, or a jam is being cleared. The Full_Closed_Control function block can operate the motor using the motor encoder alone. If during the time that Mode=TRUE the MaxDeviation is exceeded, it will not cause an Error, however, when Mode is again set to FALSE, the difference between 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

ErrorID	Meaning
0	No error.
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.
4378	The function block is not applicable for the external axis specified.
4380	MC_SetPosition cannot be executed while the axis is already moving.
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.
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.
4392	The function block can not be used with an inverter axis.
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.
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 POUs.
4646	Mode does not correspond to a valid enumeration value.
4648	The parameter number does not exist for the specified axis.
4676	The time value must be within 0 to 10 MECHATROLINK cycles.
10170	Position Error between Axis Position and Encoder Position is more than Max Deviation. Increase Max Deviation, or set position to limit position error.
57617	Instance object is NULL.
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.

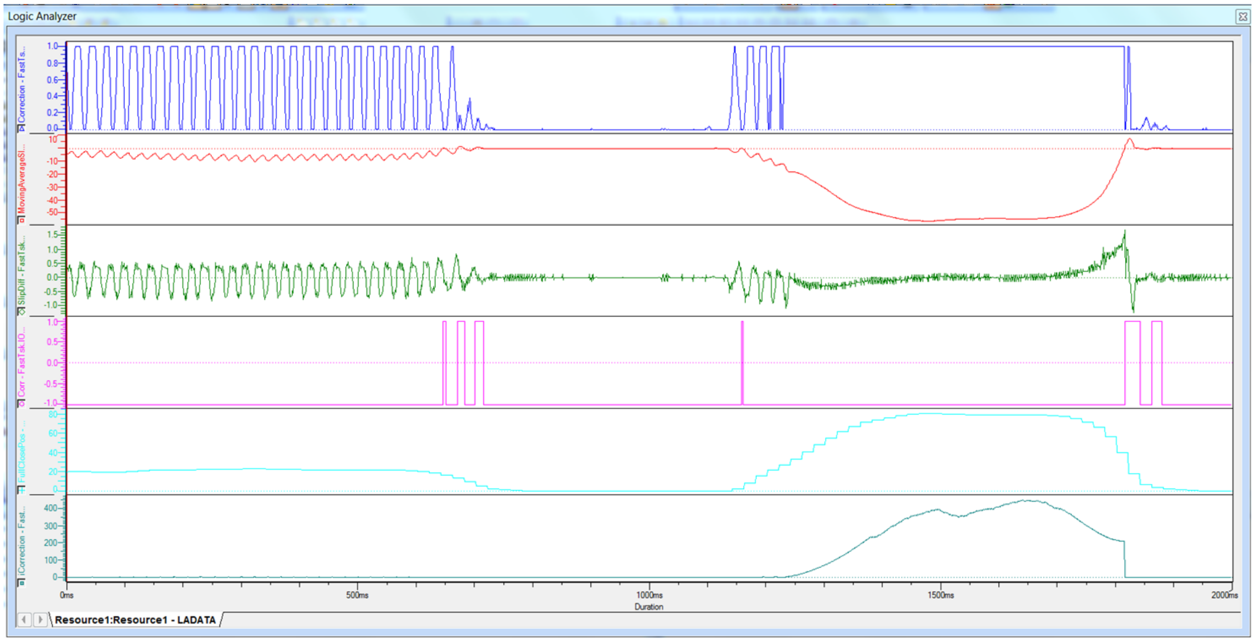
Example 1

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.



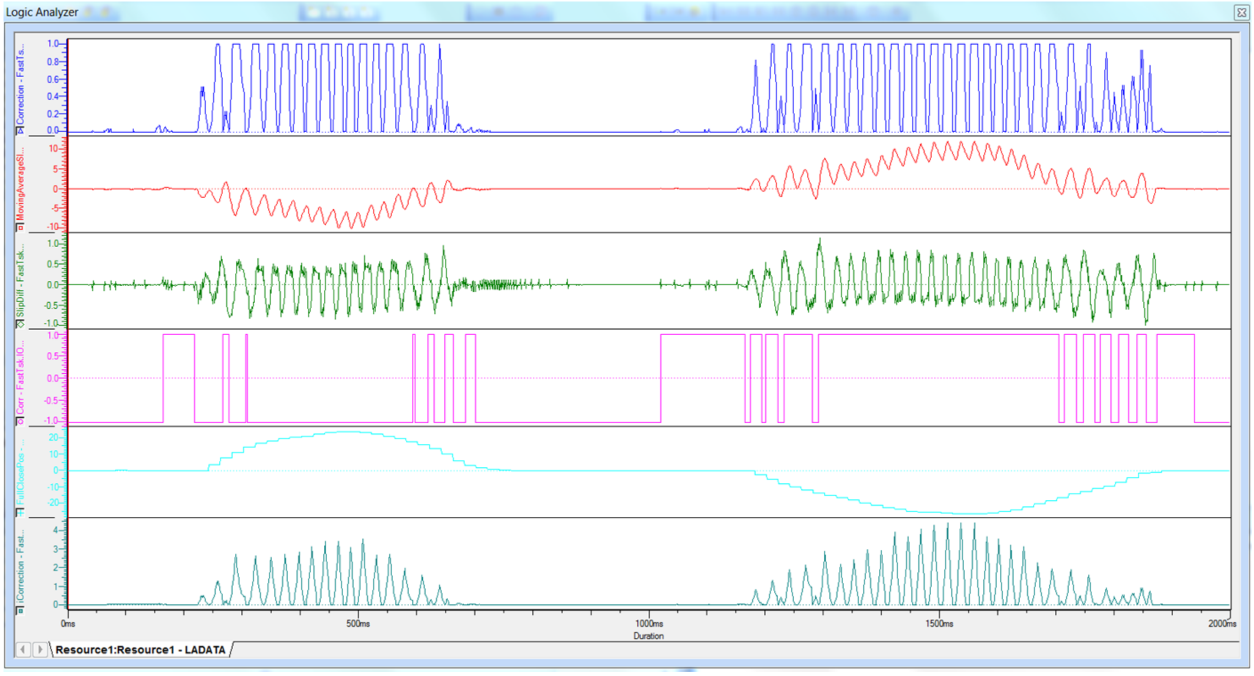
Example 2

In this example, the Logic Analyzer shows negative slip, and the correction added to Axis.



Example 3

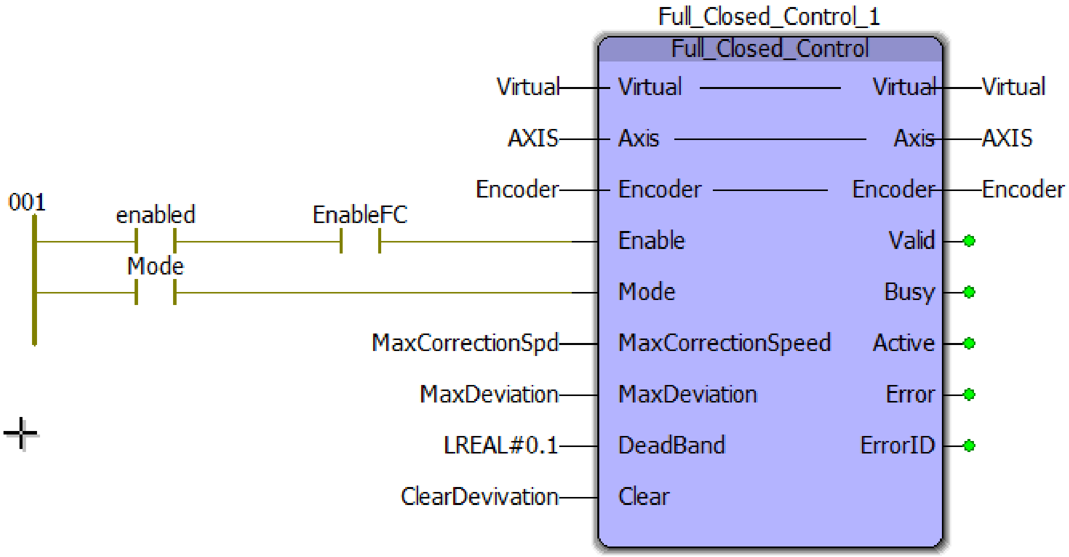
In this example, the Logic Analyzer shows both positive and negative slip.



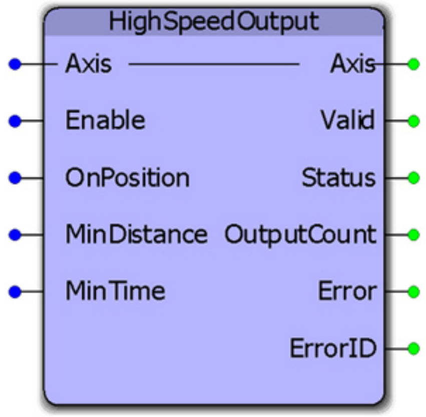
Example 4

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

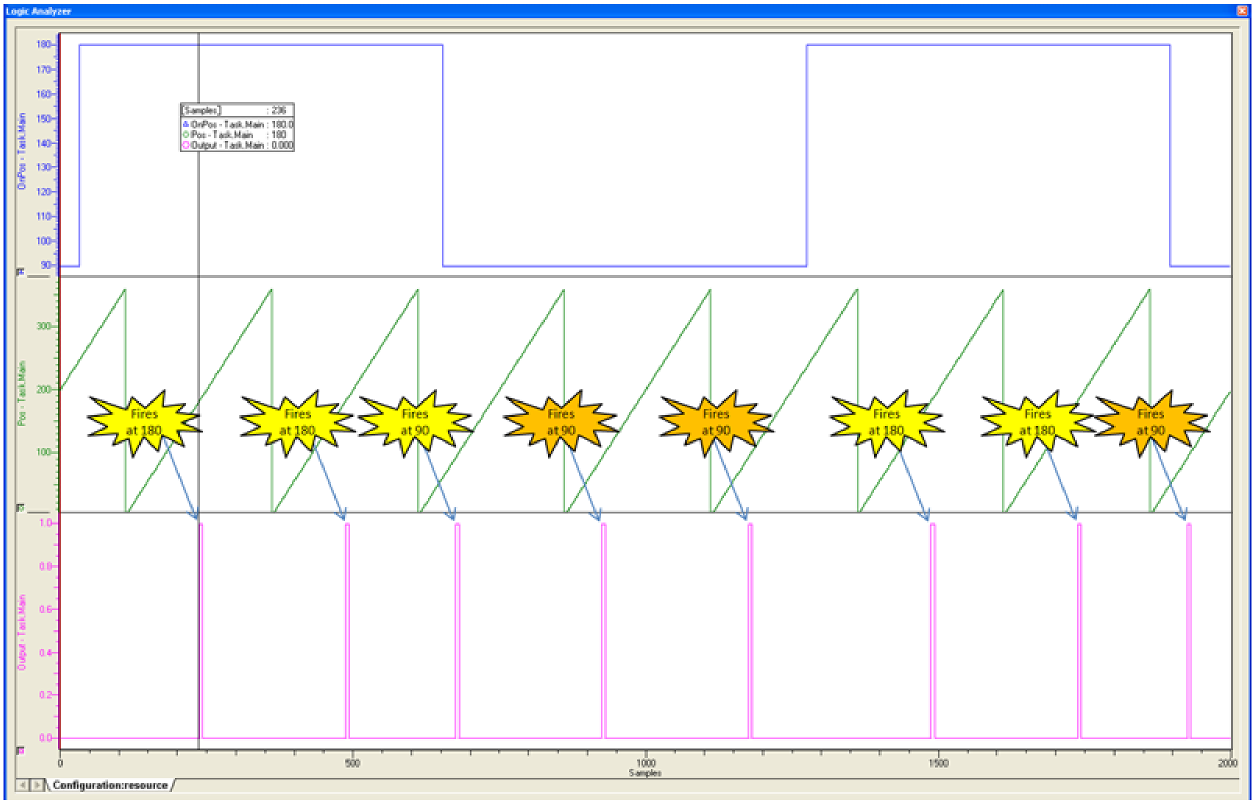
Device	Output Number	Pin Number	Software Default Name
LIO-01	DO-01	A14	Mpp_DO_01
LIO-02	DO-01	A14	Mpp_DO_01
LIO-06	DO-07	49	Mpp_DO_07
MP2600	DO-07	44, 49	MO1_DO_01

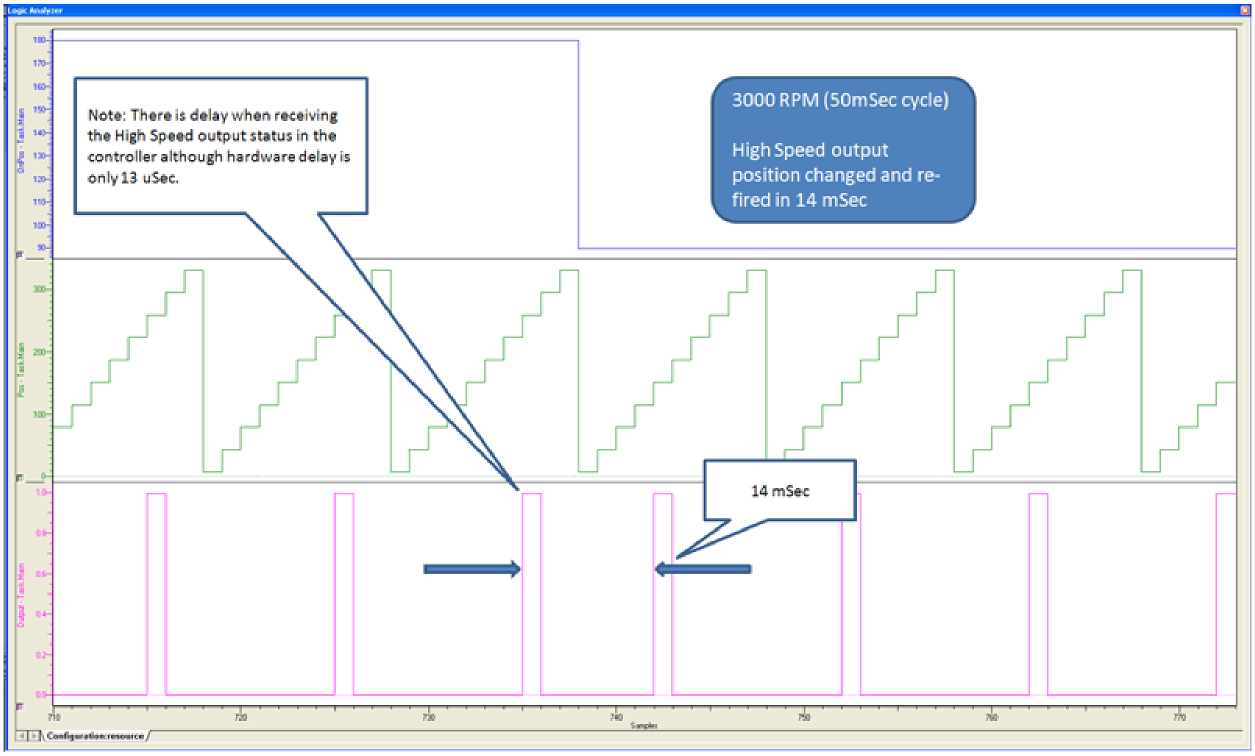
- See the HighSpeedOutput eLearning Module on Yaskawa's YouTube channel.

Error Description

ErrorID	Meaning
0	No error.
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.
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 POUs.

Timing Diagram





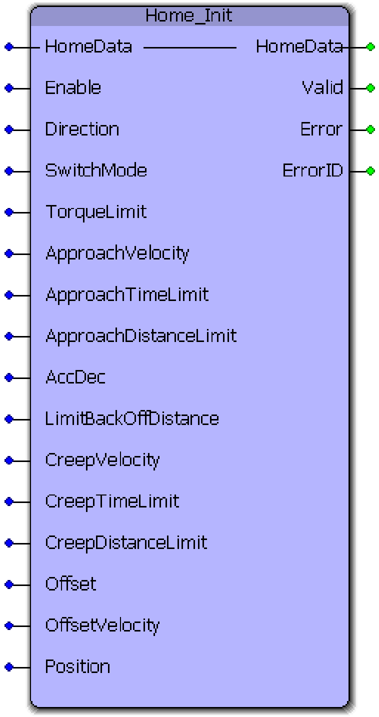
Note: There is delay when receiving the High Speed output status in the controller although hardware delay is only 13 uSec.

3000 RPM (50mSec cycle)

High Speed output position changed and re-fired in 14 mSec

14 mSec

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

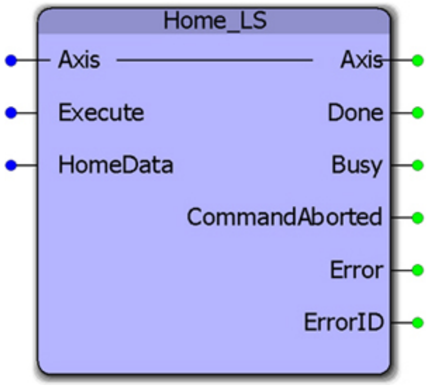
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	HomeData	HomeStruct	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT				**Default**
B	Enable	BOOL	Upon the rising edge, all other function block inputs are read 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_	Edge On is the only mode supported.	

		SwitchMode		
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				**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	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

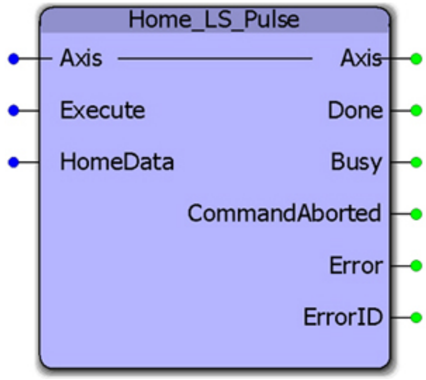
ErrorID	Meaning
0	No error.
1	Time limit exceeded.
2	Distance limit exceeded.
3	Torque limit exceeded.
4369	The move could not be buffered because the motion queue is full. The default queue size for single axis moves is 16. For Mechatrolink groups, the queue size can be set in the Hardware Configuration. 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.
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 aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
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.
4396	Axis latch function already in use.
4397	Over travel is limit still ON after attempting to move away from it.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4646	Mode does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10037	Offset cannot be in the same direction as the original motion into the limit switch.
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.
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.

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 and 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.	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.	

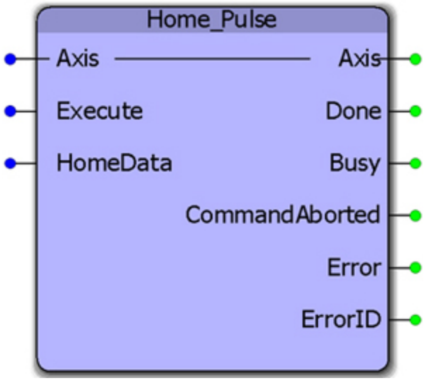
Notes

See the Home_LS_Pulse eLearning Module on Yaskawa's YouTube channel.

Error Description

ErrorID	Meaning
0	No error.
1	Time limit exceeded.
2	Distance limit exceeded.
3	Torque limit exceeded.
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. 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.
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 aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
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.
4396	Axis latch function already in use.
4397	Over travel is limit still ON after attempting to move away from it.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4646	Mode does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10037	Offset cannot be in the same direction as the original motion into the limit switch.
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.
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.

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 and offset move away from the limit, and sets a home position.

Library

PLCopen Toolbox

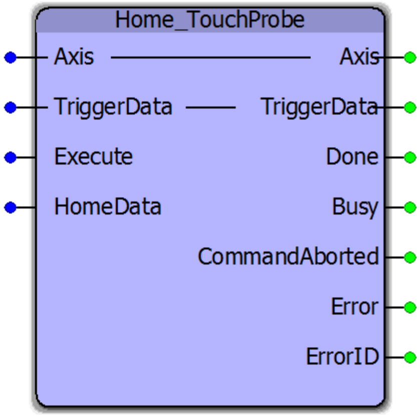
Parameters

*	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 ignored 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	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

ErrorID	Meaning
0	No error.
1	Time limit exceeded.
2	Distance limit exceeded.
3	Torque limit exceeded.
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. 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.
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 aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
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.
4396	Axis latch function already in use.
4397	Over travel is limit still ON after attempting to move away from it.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4646	Mode does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10037	Offset cannot be in the same direction as the original motion into the limit switch.
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.
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.

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 out-	

			put 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.

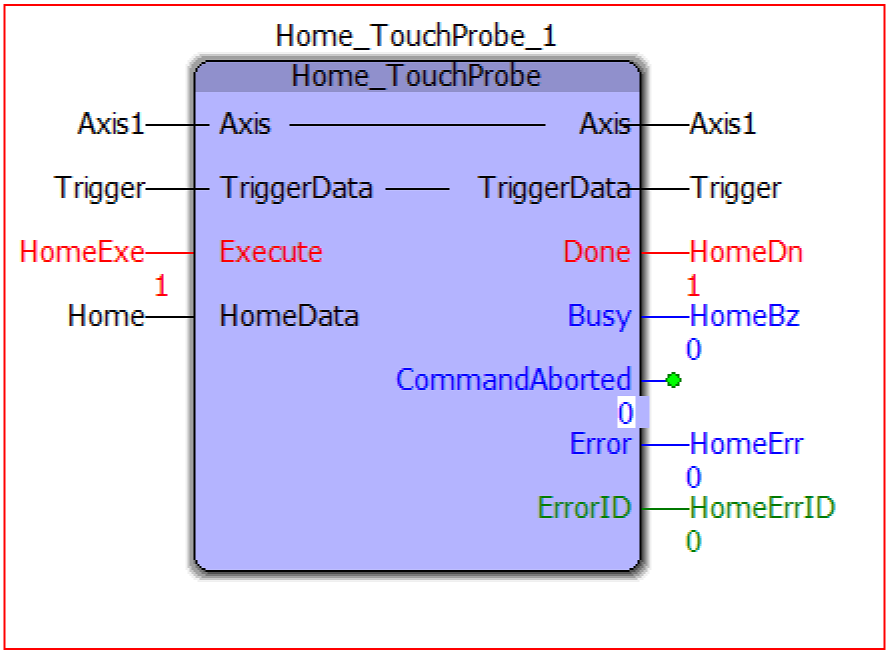
Error Description

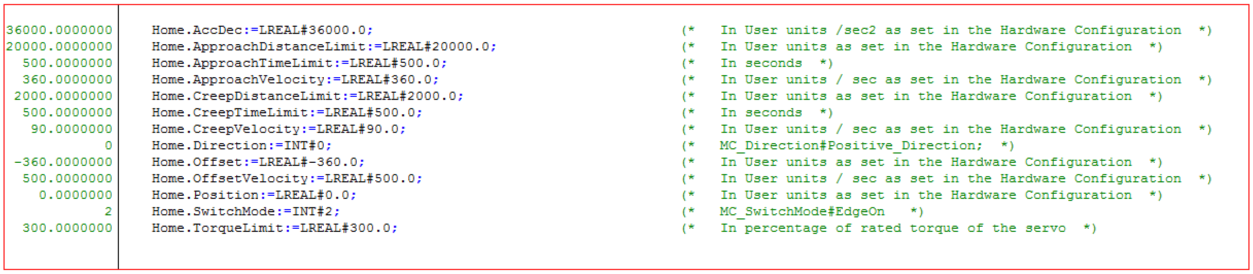
ErrorID	Meaning
0	No error.
1	Time limit exceeded.
2	Distance limit exceeded.
3	Torque limit exceeded.
4369	The move could not be buffered because the motion queue is full. The default queue size for single axis moves is 16. For Mechatrolink groups, the queue size can be set in the Hardware Configuration. 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.
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 aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
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.
4396	Axis latch function already in use.
4397	Over travel is limit still ON after attempting to move away from it.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4646	Mode does not correspond to a valid enumeration value.
4658	Velocity parameter is less than or equal to zero.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10037	Offset cannot be in the same direction as the original motion into the limit switch.
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.
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.

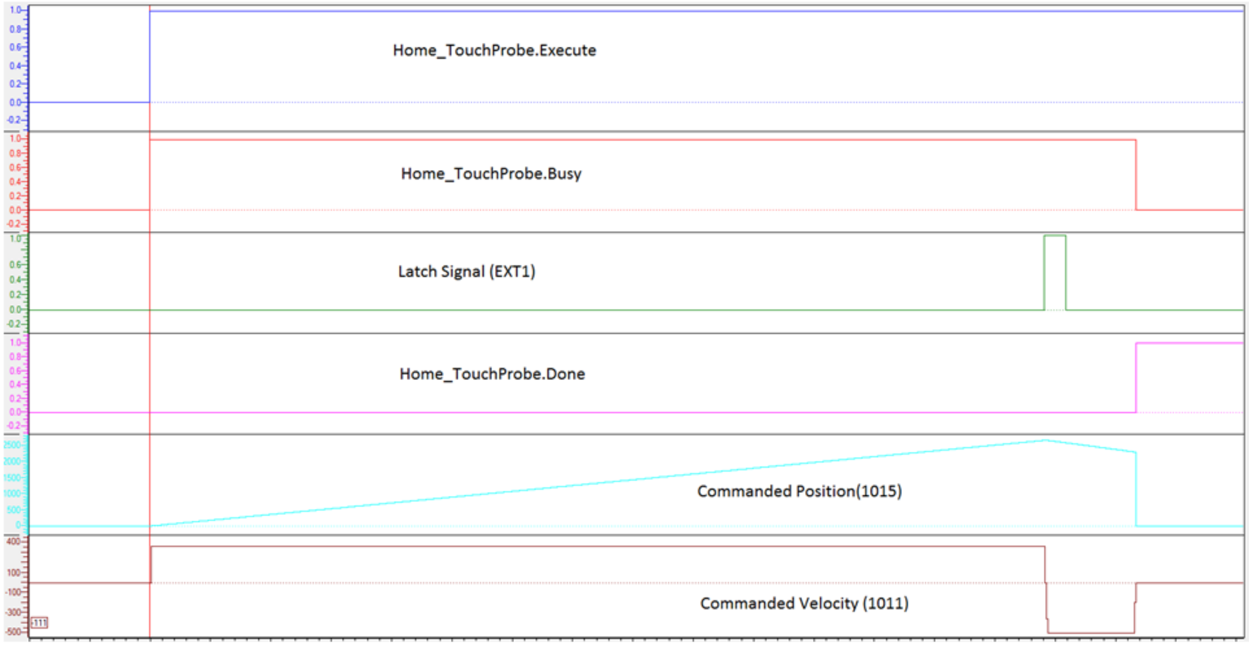
Example

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.

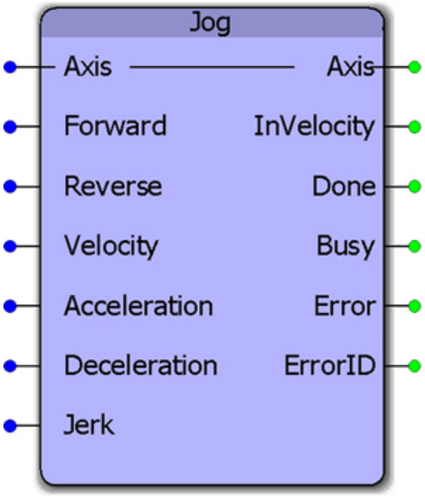




```
36000.0000000    Home.AccDec:=LREAL#36000.0;                      (*   In User units /sec2 as set in the Hardware Configuration  *)
20000.0000000    Home.ApproachDistanceLimit:=LREAL#20000.0;       (*   In User units as set in the Hardware Configuration  *)
  500.0000000    Home.ApproachTimeLimit:=LREAL#500.0;             (*   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  *)
```


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
Home_TouchProbe.Execute
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
Home_TouchProbe.Busy
1.0
0.6
0.4
0.2
0.0
-0.2
Latch Signal (EXT1)
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
Home_TouchProbe.Done
2500
2000
1500
1000
500
0
Commanded Position(1015)
400
100
-100
-300
-500
-111
Commanded Velocity (1011)

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 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	*Jerk*	*LREAL*	*[[[Undefined variable Primary.ParameterNotSupported]]] Use S-Curve parameters 1300 and 1301. Value of the jerk in [user units / second^3].*	*LREAL#0.0*

VAR_OUTPUT			
B	InVelocity	BOOL	Set high when the axis first reaches the specified velocity (function is complete). This output is reset when execute goes 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.

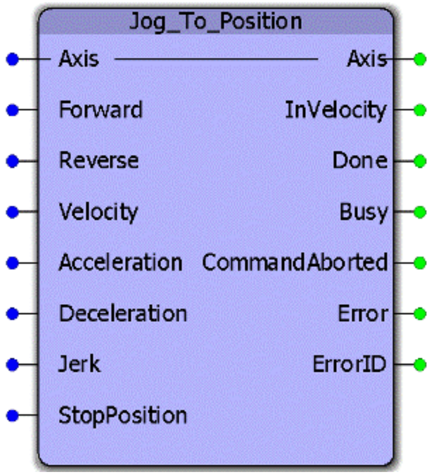
Notes

- 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

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4665	Velocity parameter is negative.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
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.

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

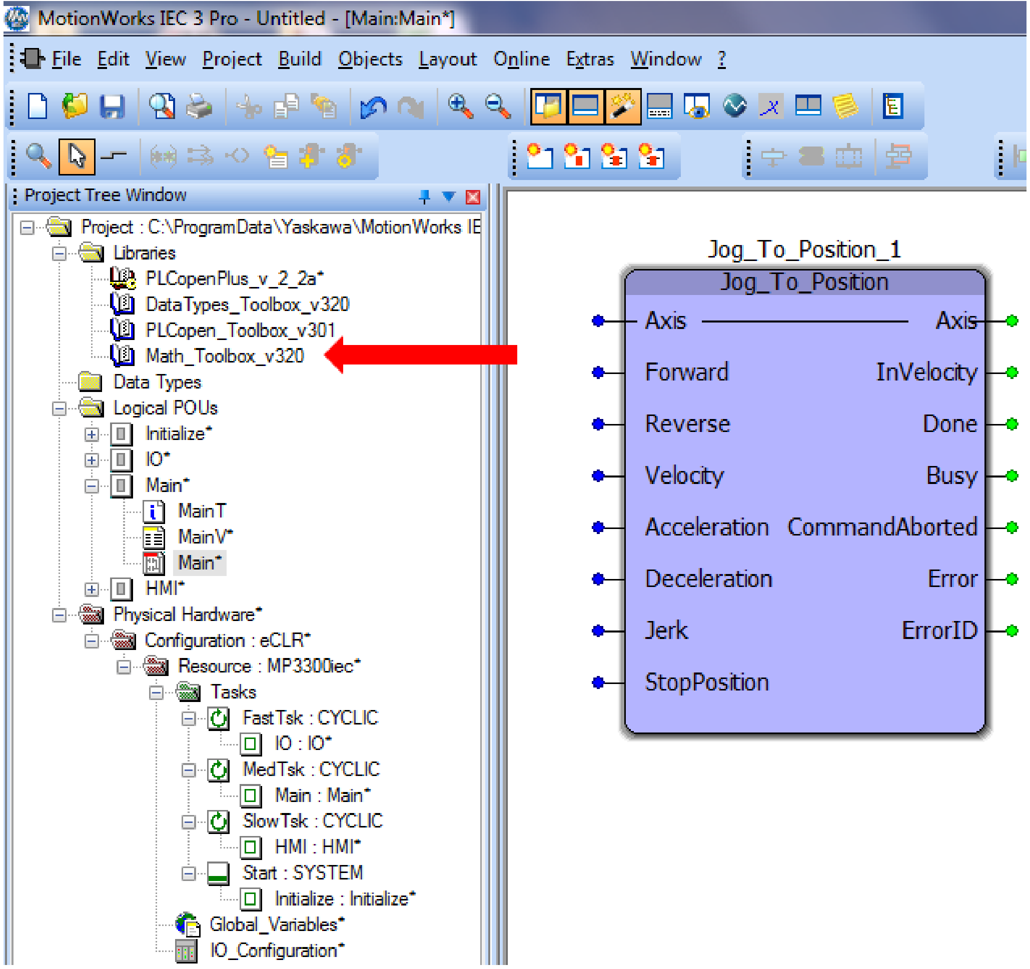
*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	[AXIS_REF](#)	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT				**Default**
V	Forward	BOOL	Runs the axis in a forward direction when TRUE.	FALSE
V	Reverse	BOOL	Runs the axis in a Reverse direction when TRUE.	FALSE
B	Velocity	LREAL	Absolute value of the velocity in user 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	*Jerk*	*LREAL*	*[[[Undefined variable Primary.ParameterNotSupported]]] Use S-Curve parameters 1300 and 1301. Value of the jerk*	*LREAL#0.0*

			in [user units / second^3].	
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). This output is reset when execute goes 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

- 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.



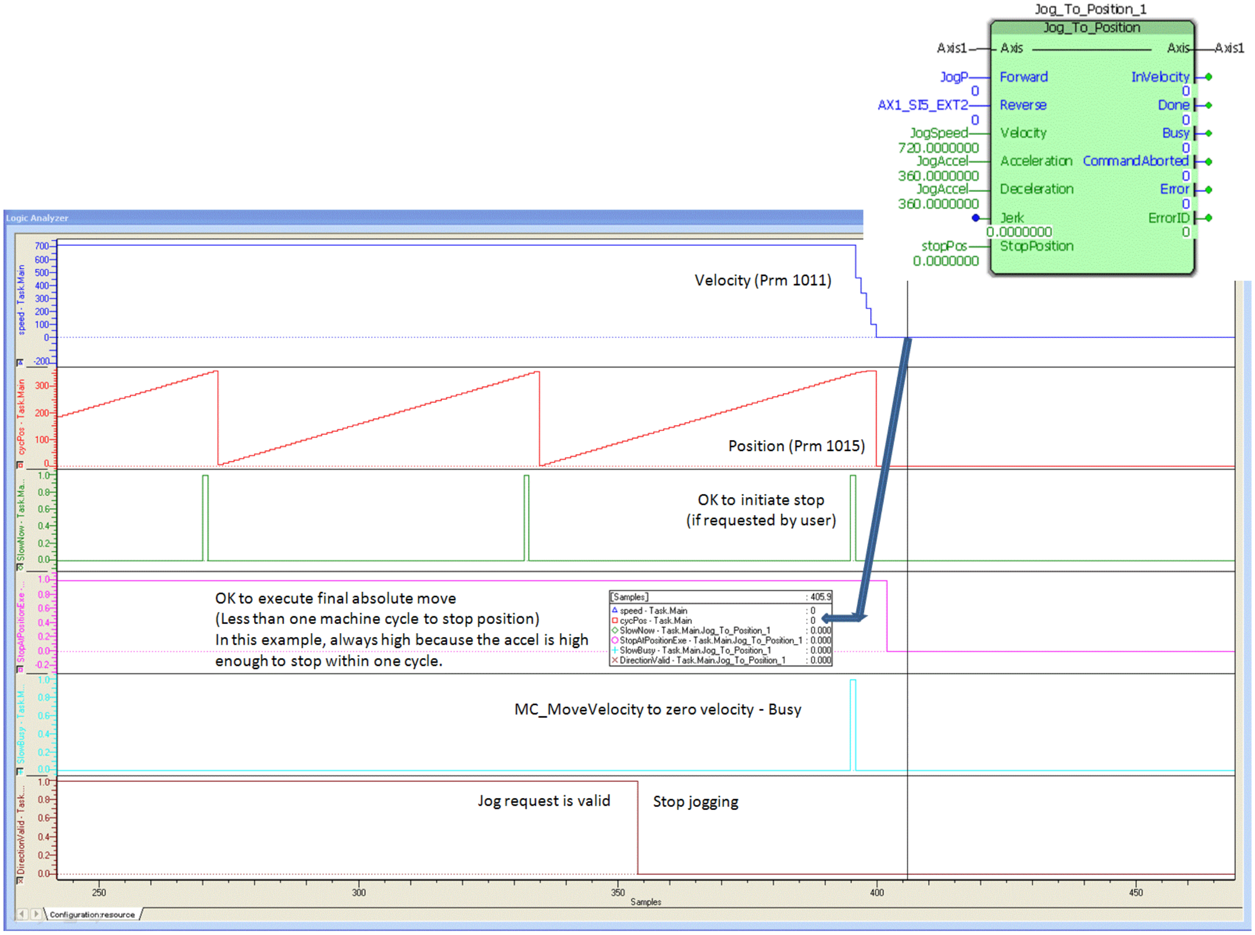
Error Description

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.

4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4665	Velocity parameter is negative.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
10060	The axis must be configured as a rotary type for this function block to be applicable.
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.

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.



Velocity (Prm 1011)

Position (Prm 1015)

OK to initiate stop
(if requested by user)

OK to execute final absolute move
(Less than one machine cycle to stop position)
In this example, always high because the accel is high enough to stop within one cycle.

[Samples]	: 405.9
△ speed - Task.Main	: 0
□ cycPos - Task.Main	: 0
◇ SlowNow - Task.Main.Jog_To_Position_1	: 0.000
○ StopAtPositionExe - Task.Main.Jog_To_Position_1	: 0.000
+ SlowBusy - Task.Main.Jog_To_Position_1	: 0.000
× DirectionValid - Task.Main.Jog_To_Position_1	: 0.000

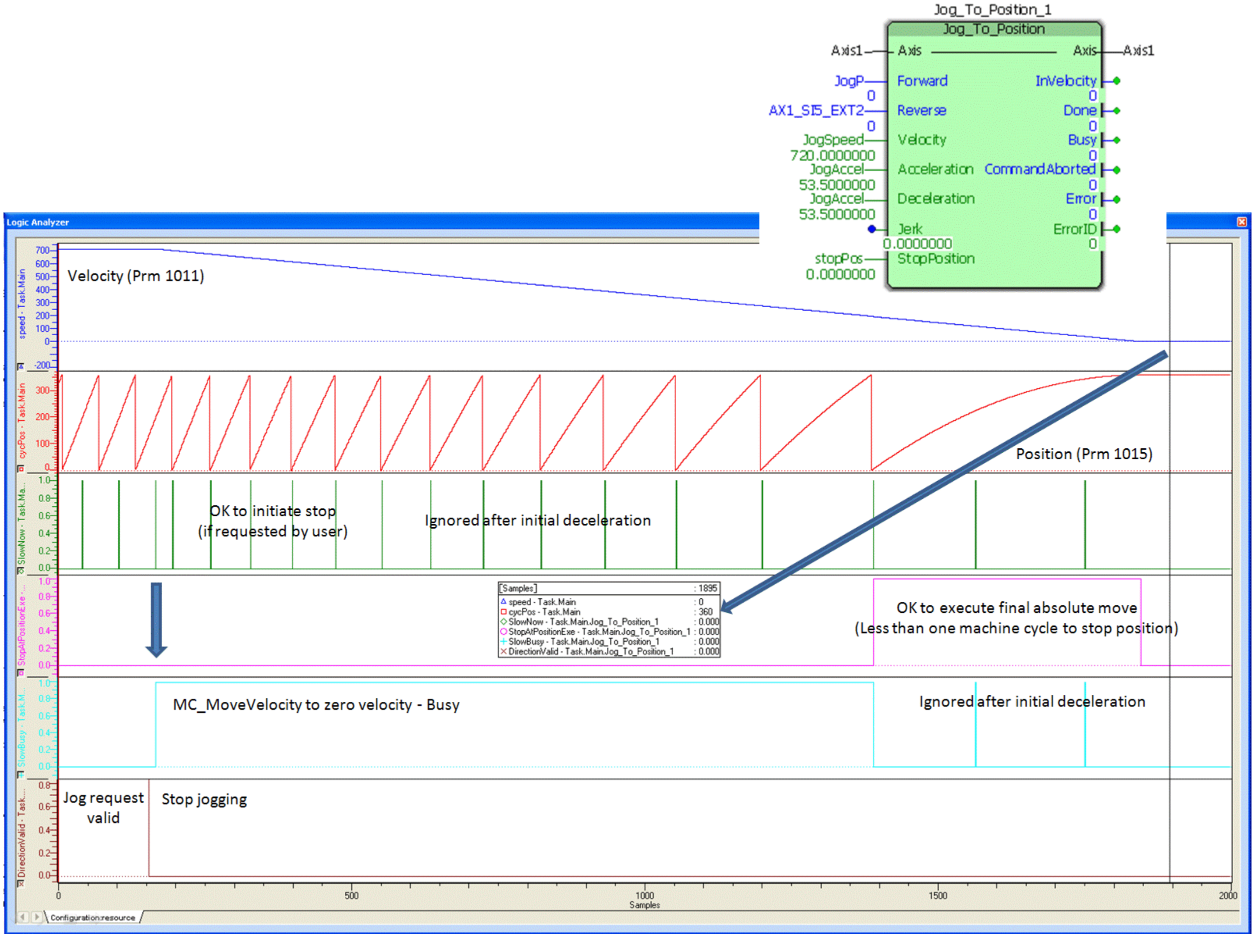
MC_MoveVelocity to zero velocity - Busy

Jog request is valid

Stop jogging

Example 2

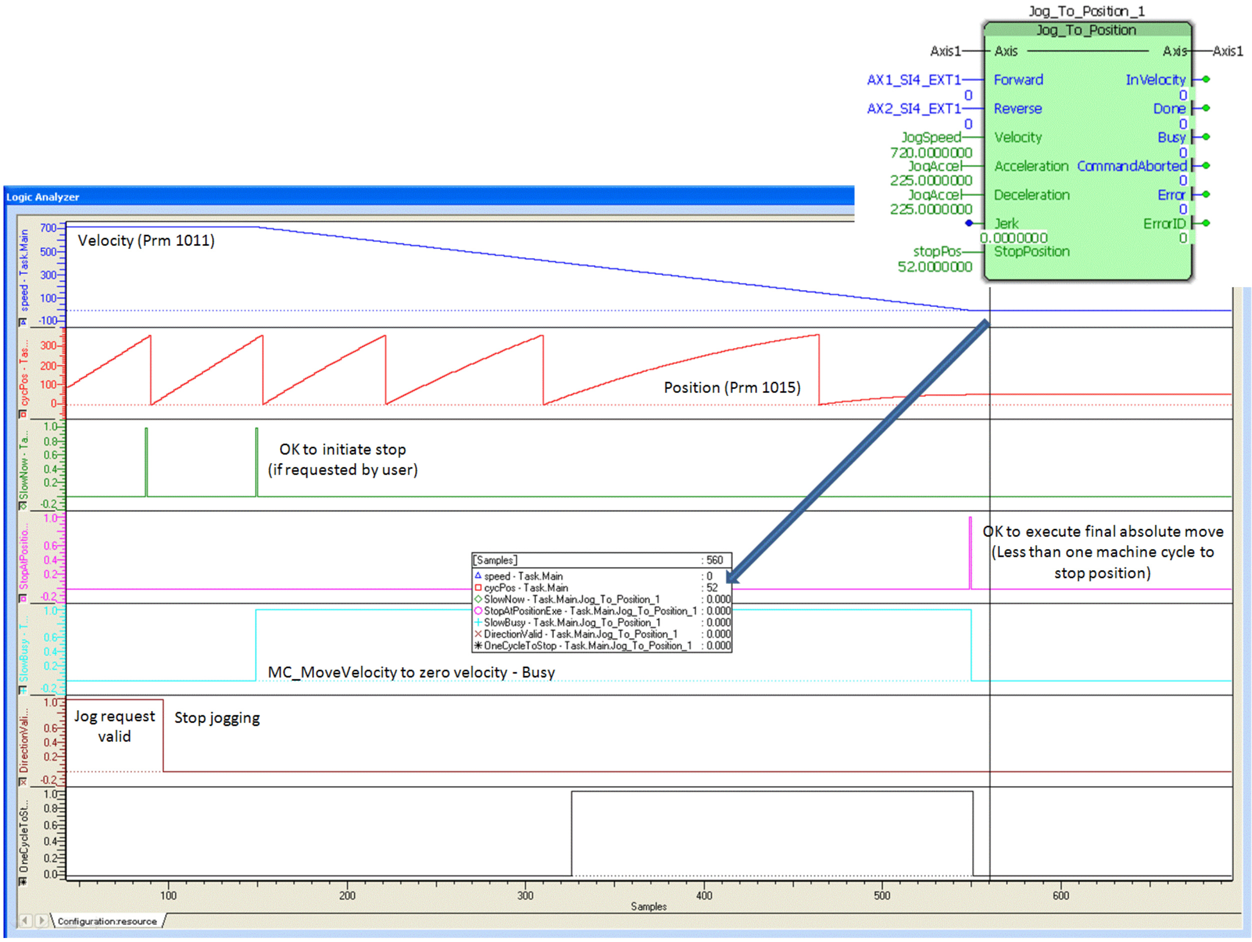
In this example, the axis requires about 13 revolutions to come to a stop at the specified velocity and deceleration. The data "SlowNow" in green is an internal monitoring bit which results from a calculation made to determine a position that will allow the motion profile to follow the deceleration rate 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 StopPositiion. The pink data indicates when the MC_MoveAbsolute is active.



```
[Samples]                                          : 1895
speed - Task.Main                                  : 0
cycPos - Task.Main                                 : 360
SlowNow - Task.Main.Jog_To_Position_1              : 0.000
StopAtPositionExe - Task.Main.Jog_To_Position_1    : 0.000
SlowBusy - Task.Main.Jog_To_Position_1             : 0.000
DirectionValid - Task.Main.Jog_To_Position_1       : 0.000
```

Example 3

The third example shows a deceleration to stop at 52 degrees.



Velocity (Prm 1011)

Position (Prm 1015)

OK to initiate stop
(if requested by user)

OK to execute final absolute move
(Less than one machine cycle to
stop position)

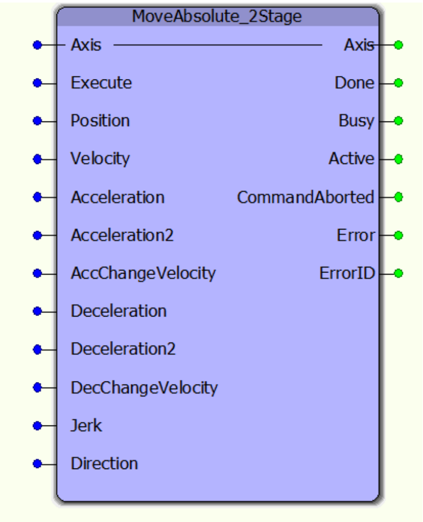
MC_MoveVelocity to zero velocity - Busy

Jog request
valid

Stop jogging

```
[Samples]                                              : 560
△ speed - Task.Main                                     : 0
□ cycPos - Task.Main                                    : 52
◇ SlowNow - Task.Main.Jog_To_Position_1                 : 0.000
○ StopAtPositionExe - Task.Main.Jog_To_Position_1       : 0.000
+ SlowBusy - Task.Main.Jog_To_Position_1                : 0.000
× DirectionValid - Task.Main.Jog_To_Position_1          : 0.000
✳ OneCycleToStop - Task.Main.Jog_To_Position_1          : 0.000
```

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 units/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 units/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 units/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 units/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*	*[[[Undefined variable Primary.ParameterNotSupported]]] Use S-Curve parameters 1300 and 1301. 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.	

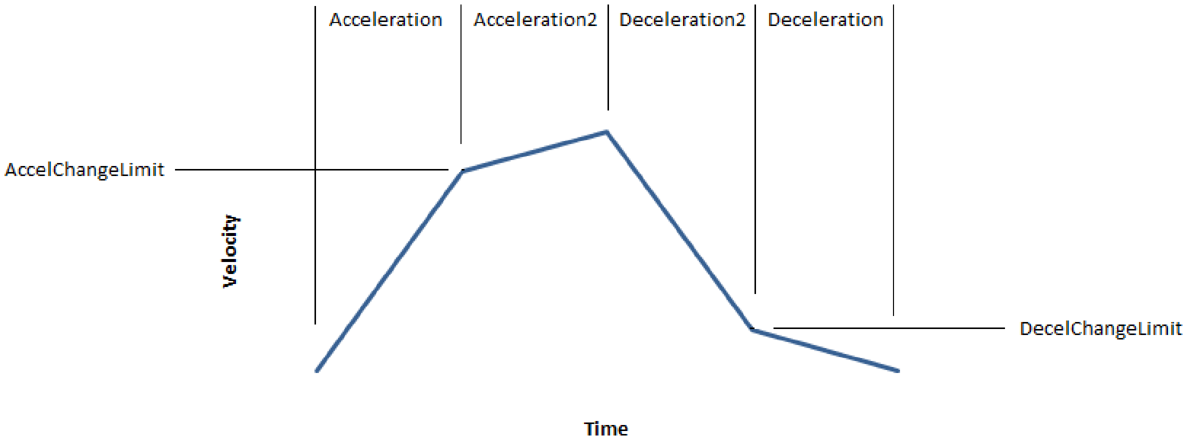
Notes

-

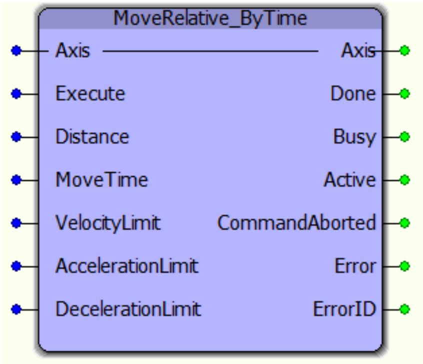
Error Description

ErrorID	Meaning

0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4663	Specified time was less than zero.
4665	Velocity parameter is negative.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.
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.



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				**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	Distance	LREAL	A relative positive or negative value within the coordinate system in user units	LREAL#0.0
V	MoveTime	LREAL	The time required (in seconds) for the move to complete.	LREAL#0.0
V	VelocityLimit	LREAL	Maximum velocity used when moving. If left unconnected, no velocity limit will be applied during move.	LREAL#0.0
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

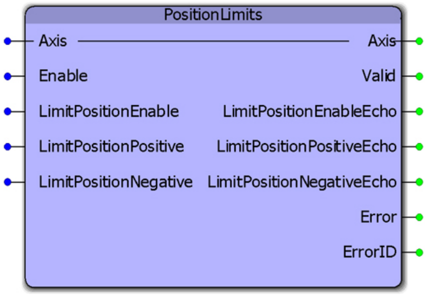
- Prior to v207, this function creates a 1/3, 1/3, 1/3 trapezoidal move, it may not be appropriate for very long moves, because the calculated commanded speed may be too high. New functionality was added for v207 which allows the function to calculate the move parameters to stay within the restraints of the new VAR_ INPUTS VelocityLimit, AccelerationLimit, and DecelerationLimit.
- See the MoveRelative_ByTime eLearning Module on Yaskawa's YouTube channel.

Error Description

ErrorID	Meaning
0	No 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. 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.
4378	The function block is not applicable for the external axis specified.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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 POUs.
4641	Buffer mode does not correspond to a valid enumeration value.
4642	Direction does not correspond to a valid enumeration value.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
4663	Specified time was less than zero.
4665	Velocity parameter is negative.
4667	Jerk is less than or equal to zero.
4893	The specified external axis may not be used. A physical axis is required.

10180	Velocity limit unable to be achieved with specified time constraint.
10181	Acceleration / deceleration limit unable to be achieved with specified time constraint.
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.

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

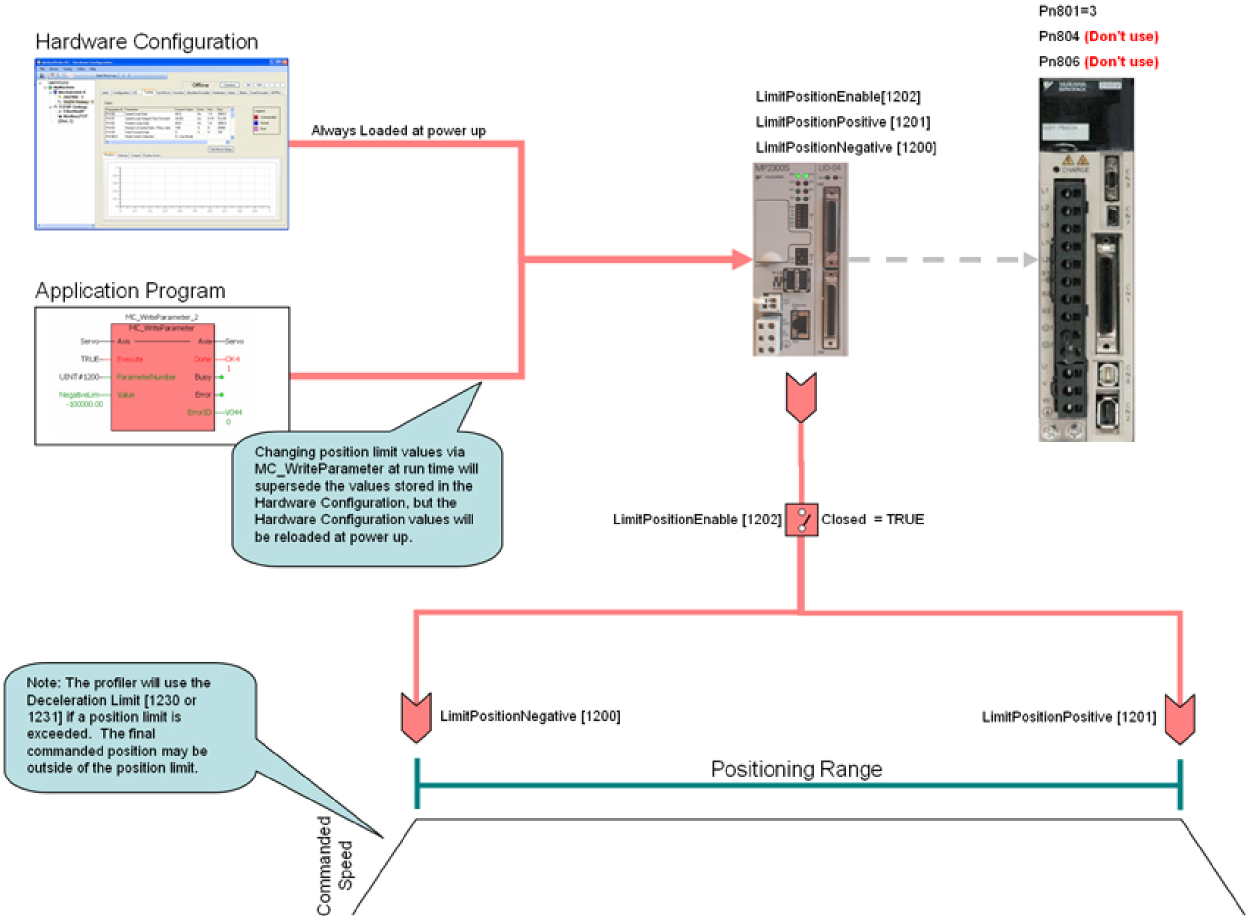
PLCopen Toolbox

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				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	LimitPositionEnable	BOOL	Enables / Disables the position limit function in the motion engine.	FALSE
V	LimitPositionPositive	LREAL	The maximum commanded position allowed	LREAL#0.0
V	LimitPositionNegative	LREAL	The minimum commanded position allowed	LREAL#0.0
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.



Changing position 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.

Note: The profiler will use the Deceleration Limit [1230 or 1231] if a position limit is exceeded. The final commanded position may be outside of the position limit.

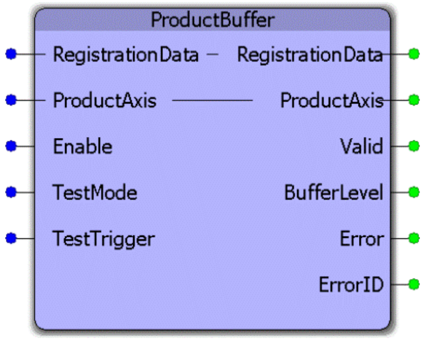
- The software position limits are managed by the MP2000iec controller. The parameters are called LimitPositionPositive and LimitPositionNegative, with values of UINT#1201 and UINT#1200 respectively. Use the MC_WriteParameter function block for these and all controller side parameters. Position limit 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

ErrorID	Meaning

0	No error.
4378	The function block is not applicable for the external axis specified.
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 POUs.
4648	The parameter number does not exist for the specified axis.
10026	Positive Position Limit must be greater than Negative Position Limit.
10027	Negative Position Limit must be less than Positive Position Limit.
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.

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 knifes 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				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	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.

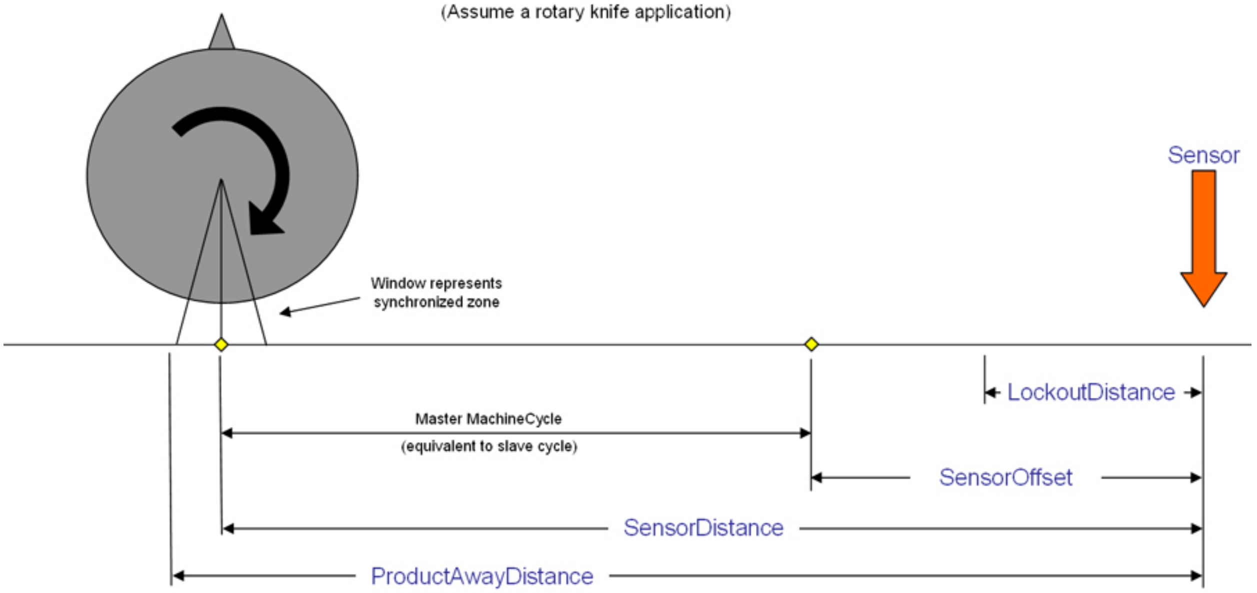
Error Description

ErrorID	Meaning
0	No error.
4396	Axis latch function already in use.
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 POUs.
4630	Trigger reference is not valid.
4894	The specified virtual axis may not be used with this function block.
10022	Product or circular buffer overrun / full.
10023	Buffer size too small / cannot be zero.
10099	Latch Feature not supported for the specified Axis. If not using a servo or external encoder, you must set TestMode=TRUE, and provide a simulated registration input using the TestTrigger input.
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.

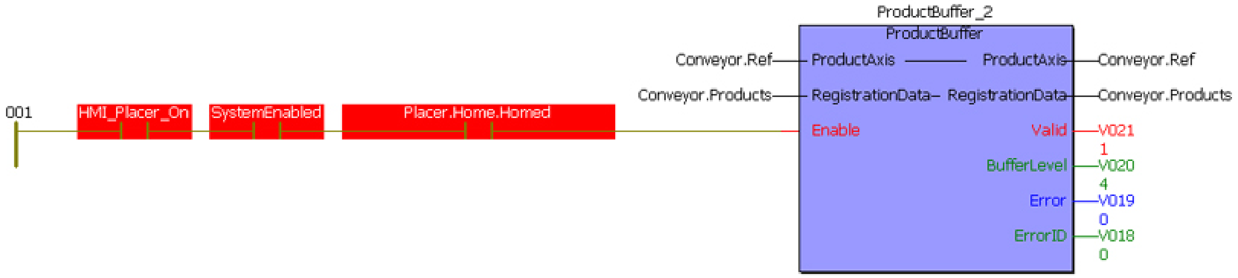
Example 1:

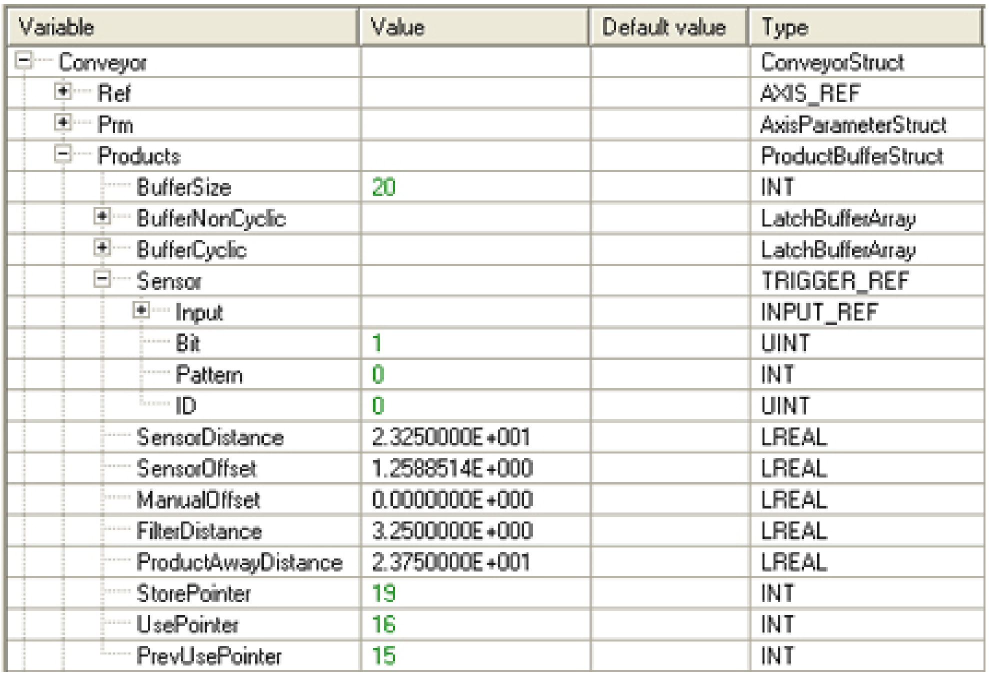
Consider a rotary knife application as shown below.

ProductBufferStruct Definitions

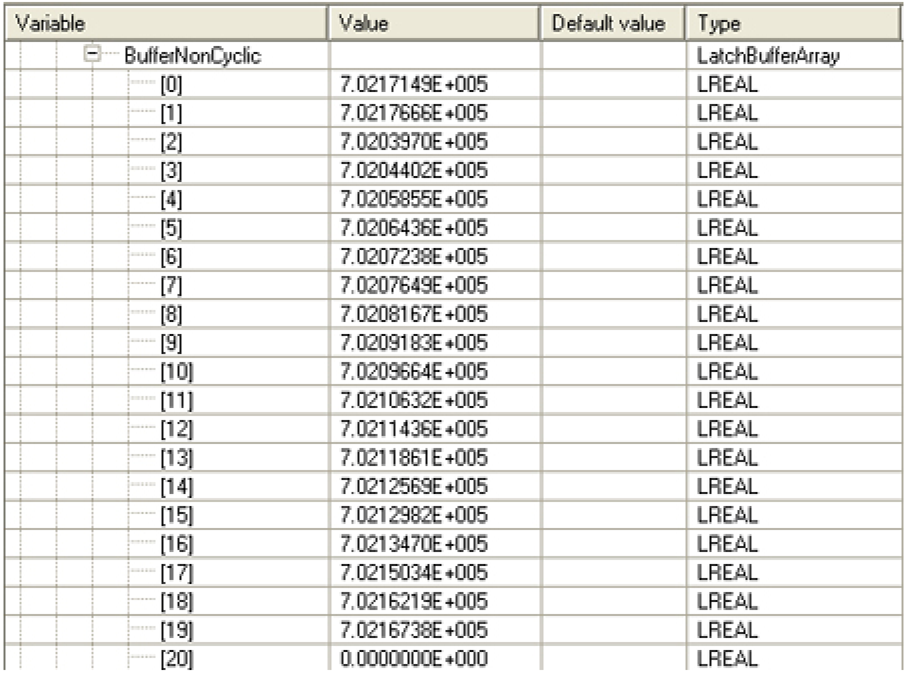

(Assume a rotary knife application)
Sensor
Window represents synchronized zone
LockoutDistance
Master MachineCycle
(equivalent to slave cycle)
SensorOffset
SensorDistance
ProductAwayDistance

```
(*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);
```

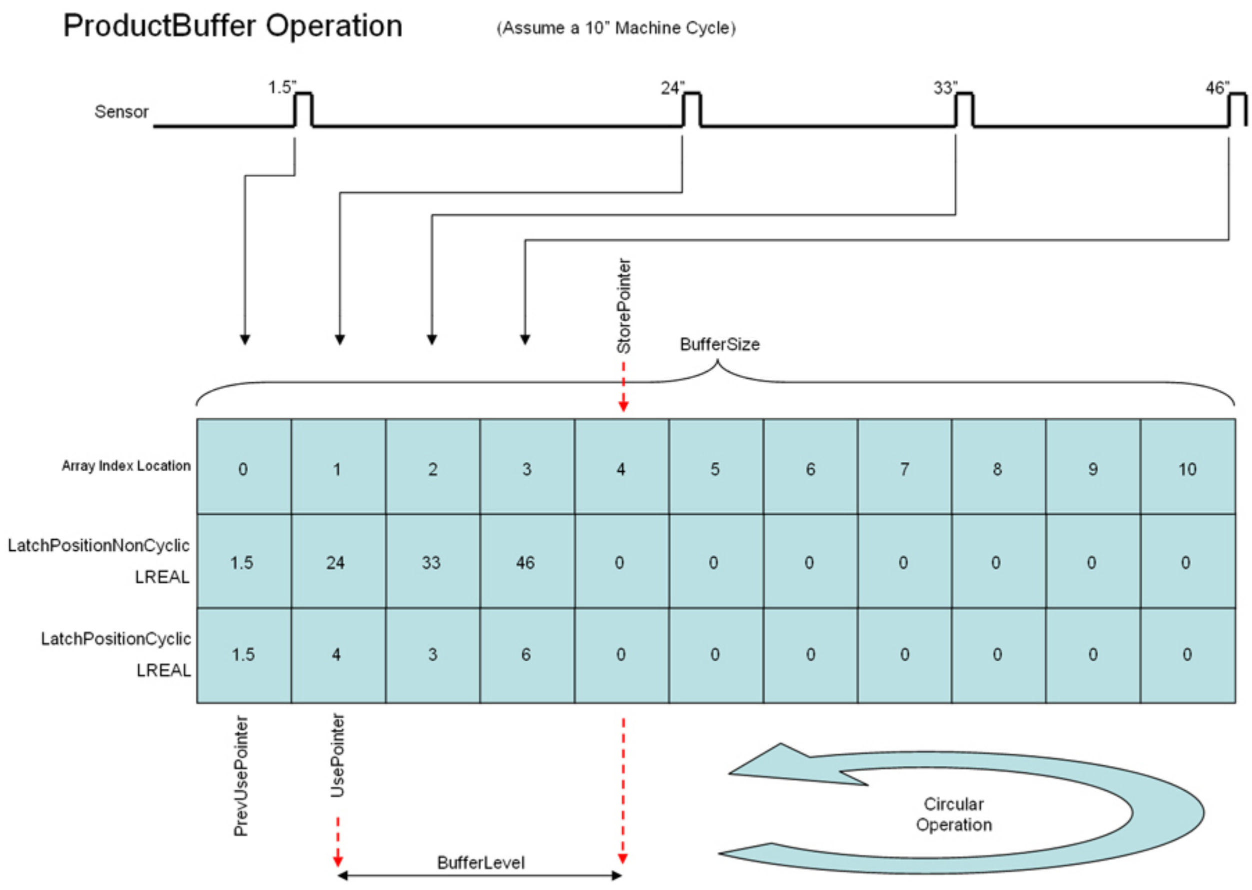




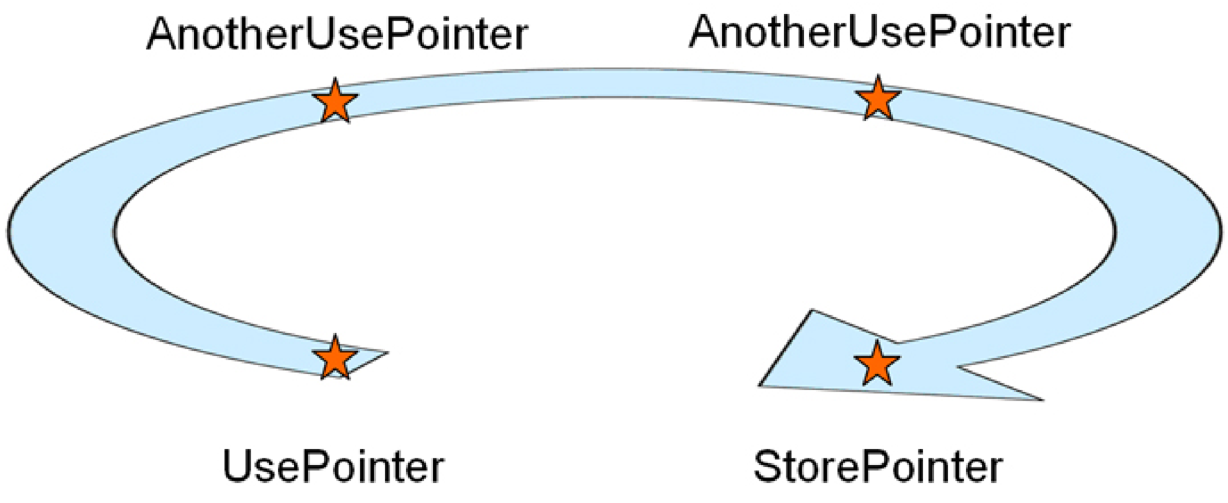
Variable	Value	Default value	Type
Conveyor			ConveyorStruct
Ref			AXIS_REF
Prm			AxisParameterStruct
Products			ProductBufferStruct
BufferSize	20		INT
BufferNonCyclic			LatchBufferArray
BufferCyclic			LatchBufferArray
Sensor			TRIGGER_REF
Input			INPUT_REF
Bit	1		UINT
Pattern	0		INT
ID	0		UINT
SensorDistance	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

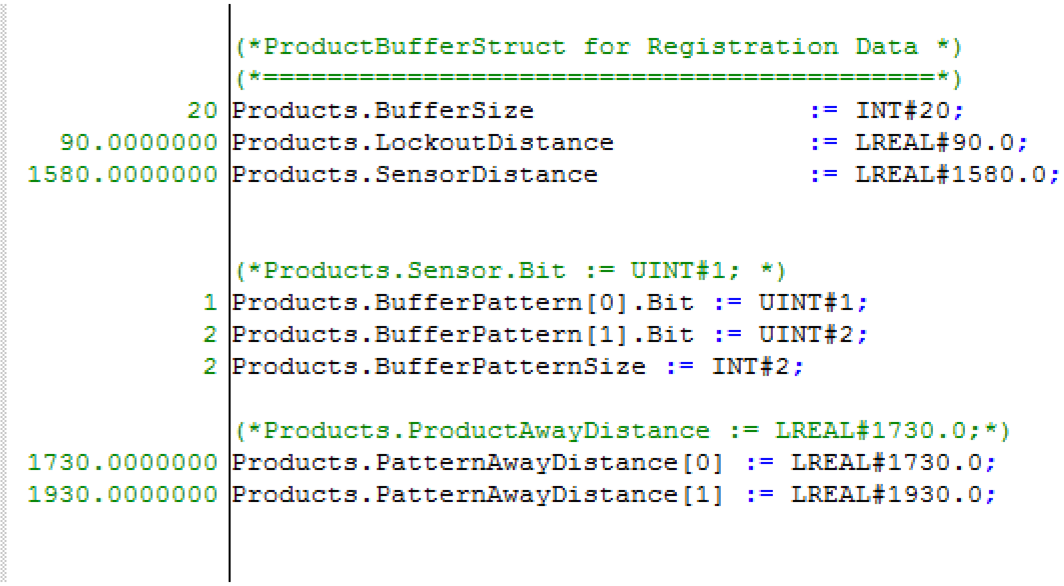


Array Index Location	0	1	2	3	4	5	6	7	8	9	10
LatchPositionNonCyclic LREAL	1.5	24	33	46	0	0	0	0	0	0	0
LatchPositionCyclic LREAL	1.5	4	3	6	0	0	0	0	0	0	0



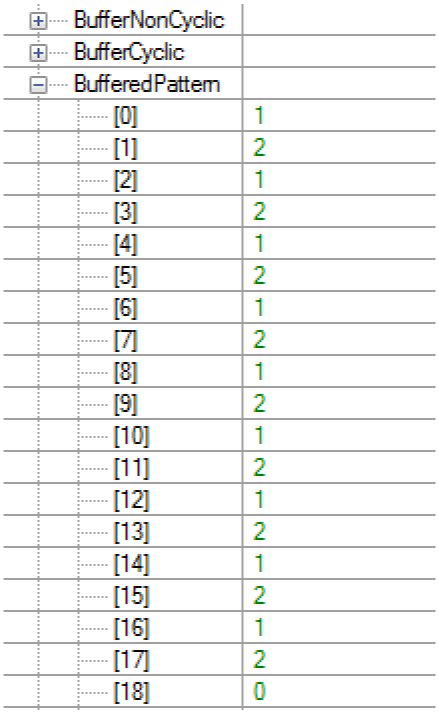
Example 2:

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.



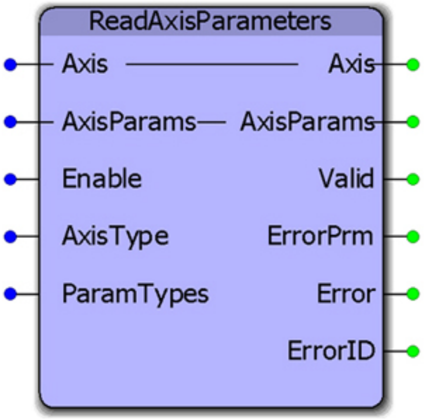
```
                (*ProductBufferStruct for Registration Data *)
                (*=========================================*)
          20    Products.BufferSize                  := INT#20;
  90.0000000    Products.LockoutDistance             := LREAL#90.0;
1580.0000000    Products.SensorDistance              := LREAL#1580.0;


                (*Products.Sensor.Bit := UINT#1; *)
           1    Products.BufferPattern[0].Bit := UINT#1;
           2    Products.BufferPattern[1].Bit := UINT#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

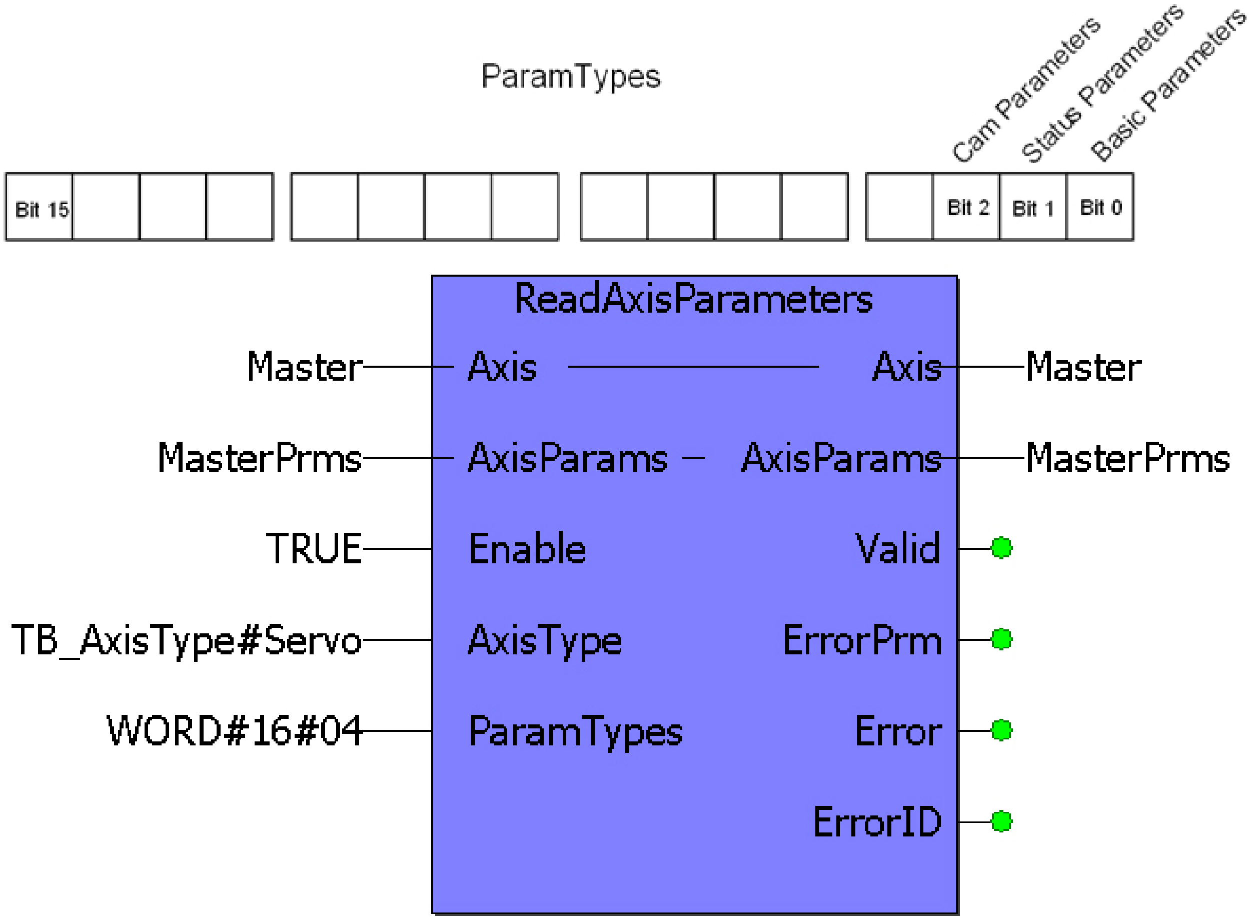
*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
V	AxisParams	AxisParameterStruct	User Defined DataType declared in the PLCopen Toolbox.	
VAR_INPUT				**Default**
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

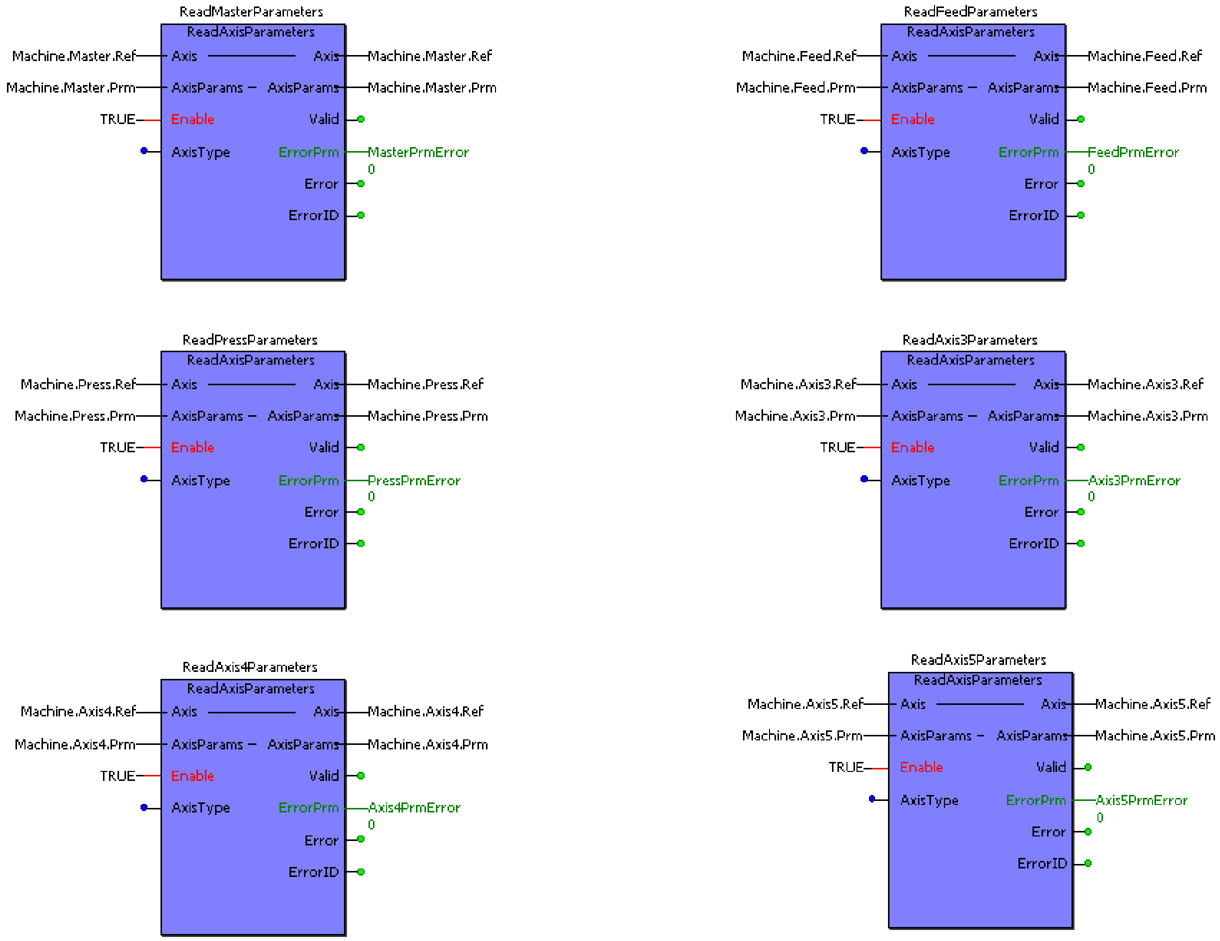
Cam	CamMasterCycle	1512
Cam	CamMasterPosition	1500
Cam	CamMasterScale	1510
Cam	CamMasterShift	1511
Cam	CamMasterShiftedCyclic	1502
Cam	CamMasterShiftedPosition	1501
Cam	CamOffset	1531
Cam	CamScale	1530
Cam	CamShiftRemaining	1513
Cam	CamState	1540
Cam	CamTableIDEngaged	1541
Cam	CamTableOutput	1520
Status	BufferedMotionBlocks	1600
Status	CommandedAcceleration	1012
Status	PositionWindow	1120

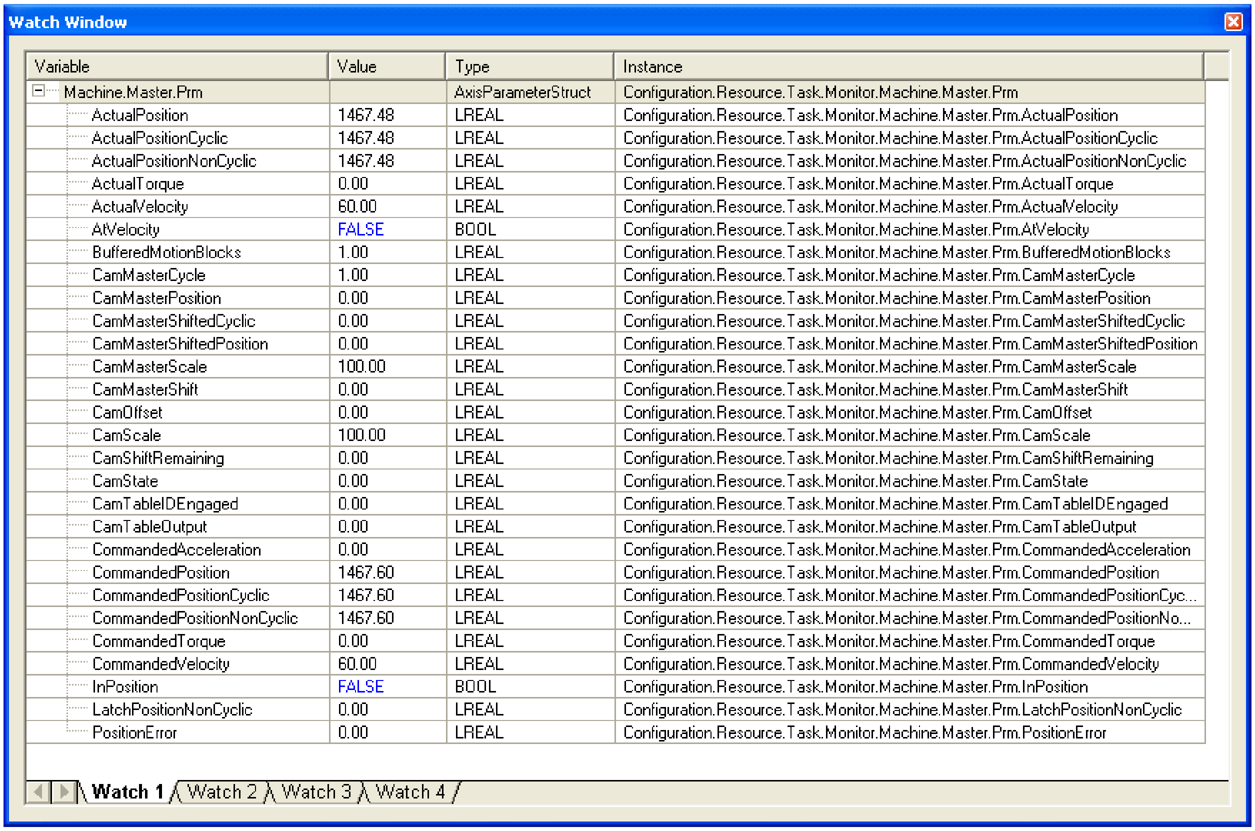
- See the ReadAxisParameters eLearning Module on Yaskawa's YouTube channel.

Error Description

ErrorID	Meaning
0	No error.
4378	The function block is not applicable for the external axis specified.
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 POUs.
4648	The parameter number does not exist for the specified axis.
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.

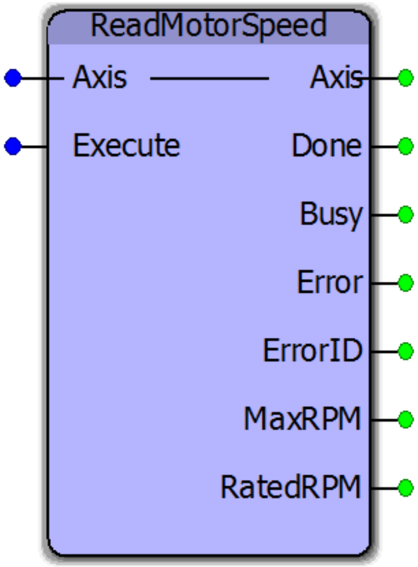
Example


ReadMasterParameters
ReadAxisParameters
Machine.Master.Ref
Axis
Axis
Machine.Master.Ref
Machine.Master.Prm
AxisParams
AxisParams
Machine.Master.Prm
TRUE
Enable
Valid
AxisType
ErrorPrm
MasterPrmError
0
Error
ErrorID
ReadFeedParameters
ReadAxisParameters
Machine.Feed.Ref
Axis
Axis
Machine.Feed.Ref
Machine.Feed.Prm
AxisParams
AxisParams
Machine.Feed.Prm
TRUE
Enable
Valid
AxisType
ErrorPrm
FeedPrmError
0
Error
ErrorID
ReadPressParameters
ReadAxisParameters
Machine.Press.Ref
Axis
Axis
Machine.Press.Ref
Machine.Press.Prm
AxisParams
AxisParams
Machine.Press.Prm
TRUE
Enable
Valid
AxisType
ErrorPrm
PressPrmError
0
Error
ErrorID
ReadAxis3Parameters
ReadAxisParameters
Machine.Axis3.Ref
Axis
Axis
Machine.Axis3.Ref
Machine.Axis3.Prm
AxisParams
AxisParams
Machine.Axis3.Prm
TRUE
Enable
Valid
AxisType
ErrorPrm
Axis3PrmError
0
Error
ErrorID
ReadAxis4Parameters
ReadAxisParameters
Machine.Axis4.Ref
Axis
Axis
Machine.Axis4.Ref
Machine.Axis4.Prm
AxisParams
AxisParams
Machine.Axis4.Prm
TRUE
Enable
Valid
AxisType
ErrorPrm
Axis4PrmError
0
Error
ErrorID
ReadAxis5Parameters
ReadAxisParameters
Machine.Axis5.Ref
Axis
Axis
Machine.Axis5.Ref
Machine.Axis5.Prm
AxisParams
AxisParams
Machine.Axis5.Prm
TRUE
Enable
Valid
AxisType
ErrorPrm
Axis5PrmError
0
Error
ErrorID



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
CamTableIDEngaged	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamTableIDEngaged
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.CommandedPositionCyc...
CommandedPositionNonCyclic	1467.60	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CommandedPositionNo...
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

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				**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	

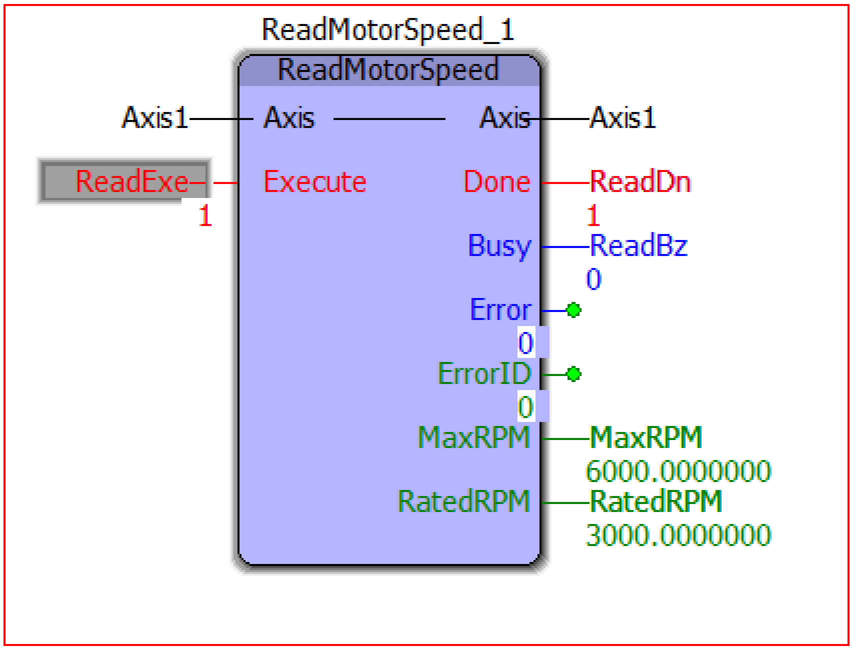
			is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.
E	MaxRPM	LREAL	Peak speed of the motor
E	RatedRPM	LREAL	Rated speed of the motor

Error Description

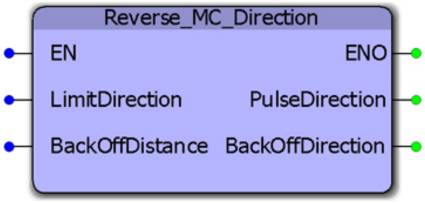
ErrorID	Meaning
0	No error.
4378	The function block is not applicable for the external axis specified.
4625	AXIS_REF.AxisNum does not correspond to an axis configured on the system. Verify the value of AxisNum matches a logical axis number in the configuration. Tip: Make sure a variable of type AXIS_REF is properly declared as a VAR or VAR_GLOBAL in all relevant POUs.
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.

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

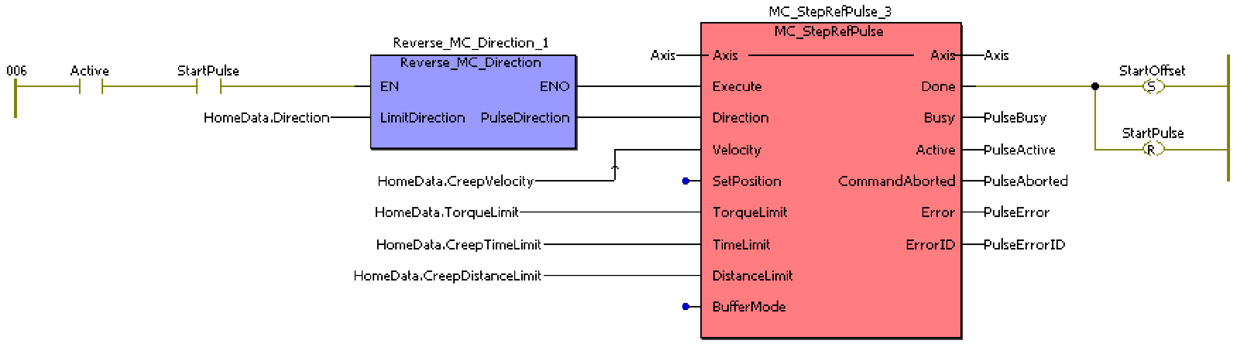
Parameters

*	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		

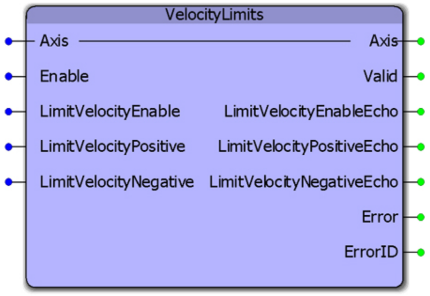
Error Description

No Errors will result, but if there is a problem with the ENum input for MC_Direction, then ENO will be FALSE.

Example


006
Active
StartPulse
Reverse_MC_Direction_1
Reverse_MC_Direction
EN
ENO
HomeData.Direction
LimitDirection
PulseDirection
MC_StepRefPulse_3
MC_StepRefPulse
Axis
Axis
Axis
Axis
Execute
Done
Direction
Busy
PulseBusy
Velocity
Active
PulseActive
HomeData.CreepVelocity
SetPosition
CommandAborted
PulseAborted
HomeData.TorqueLimit
TorqueLimit
Error
PulseError
HomeData.CreepTimeLimit
TimeLimit
ErrorID
PulseErrorID
HomeData.CreepDistanceLimit
DistanceLimit
BufferMode
StartOffset
S
StartPulse
R

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

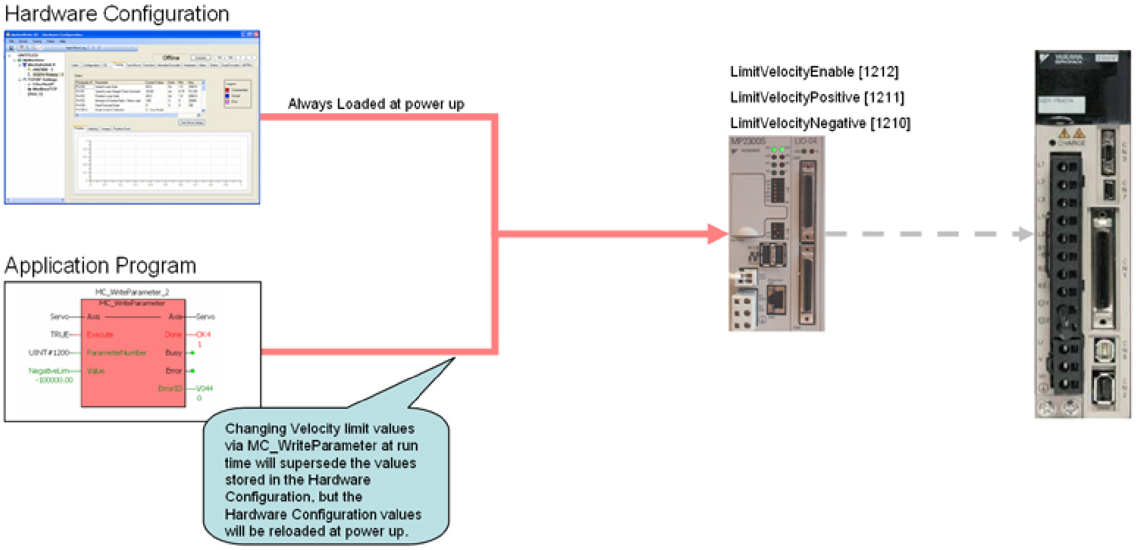
PLCopen Toolbox

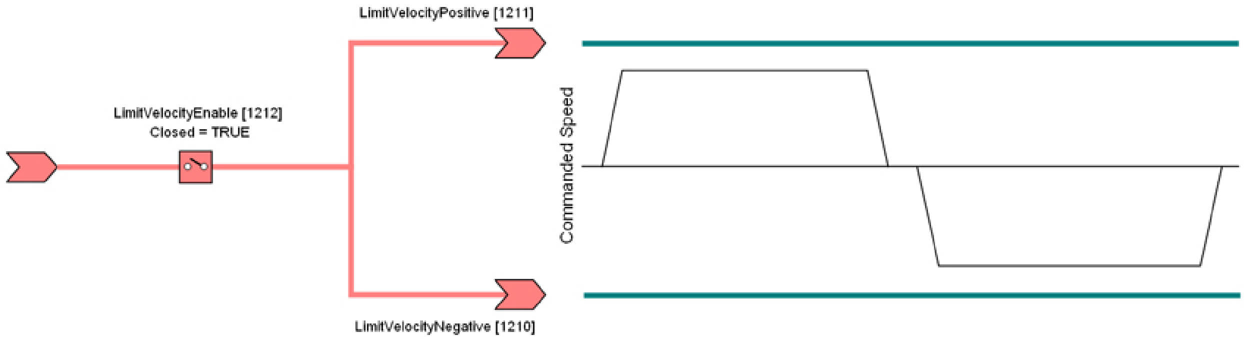
Parameters

*	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				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	LimitVelocityEnable	BOOL	Enables / Disables the velocity limit function in the motion engine.	FALSE
V	LimitVelocityPositive	LREAL	The maximum commanded velocity allowed	LREAL#0.0
V	LimitVelocityNegative	LREAL	The minimum commanded velocity allowed	LREAL#0.0
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.


Hardware Configuration
Always Loaded at power up
Application Program
MC_WriteParameter_2
MC_WriteParameter
Servo
Axis
Axis
Servo
TRUE
Execute
Done
OK4
1
UINT#1200
ParameterNumber
Busy
NegativeLim
-100000.00
Value
Error
ErrorID
V044
0
Changing Velocity 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.
LimitVelocityEnable [1212]
LimitVelocityPositive [1211]
LimitVelocityNegative [1210]
MP2300S
LIO-04

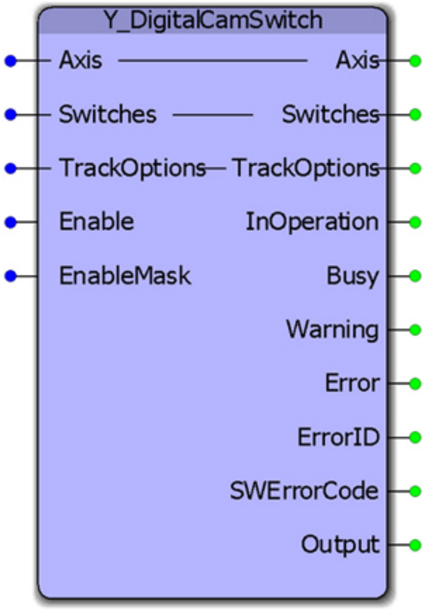

LimitVelocityPositive [1211]
LimitVelocityEnable [1212]
Closed = TRUE
Commanded Speed
LimitVelocityNegative [1210]

- The software velocity limits are managed by the MP2000iec 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

ErrorID	Meaning
0	No error.
10028	Positive Velocity Limit must be LREAL#0.0 or greater.
10029	Negative Velocity Limit must be LREAL#0.0 or lower.

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.

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.

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

ErrorID	Meaning
0	No 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 POUs.
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.
10064	Track Number outside the 0-31 range.
10065	FirstOnPosition is not equal to 0.
10066	LastOnPosition is not equal to 0.
10067	AxisDirection is not equal to 0.
10068	CamSwitchMode is not equal to 0.
10069	Duration is set to 0 or a negative value.
10070	OnCompensationScaler is set to an invalid value.
10071	OffCompensationScaler is set to an invalid value.
10072	ImproperOnPos_SetError.
10073	OnOffPosition_Error.

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

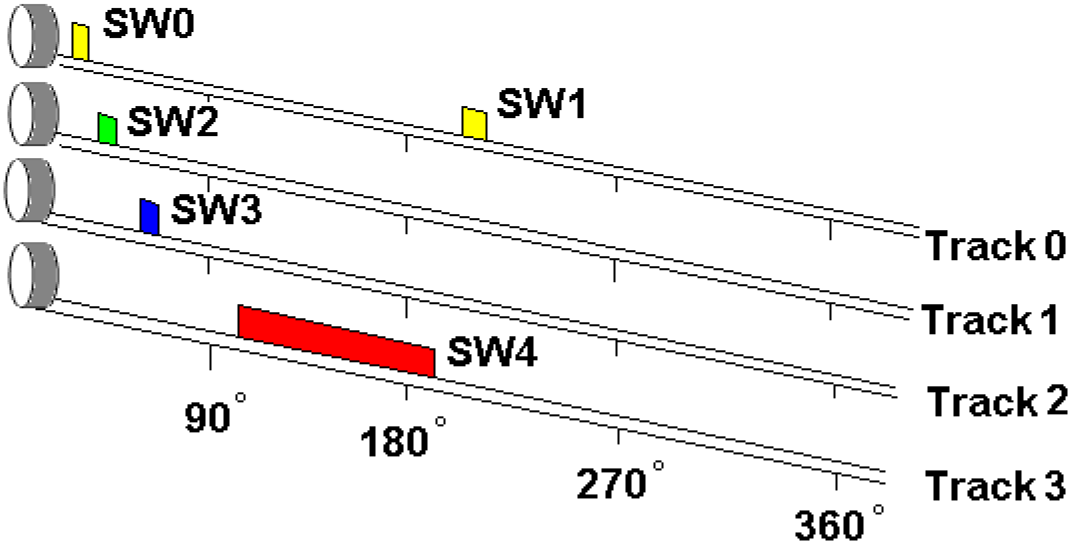
Switch 3: On Position : 50 degrees

Off Position : 60 degrees

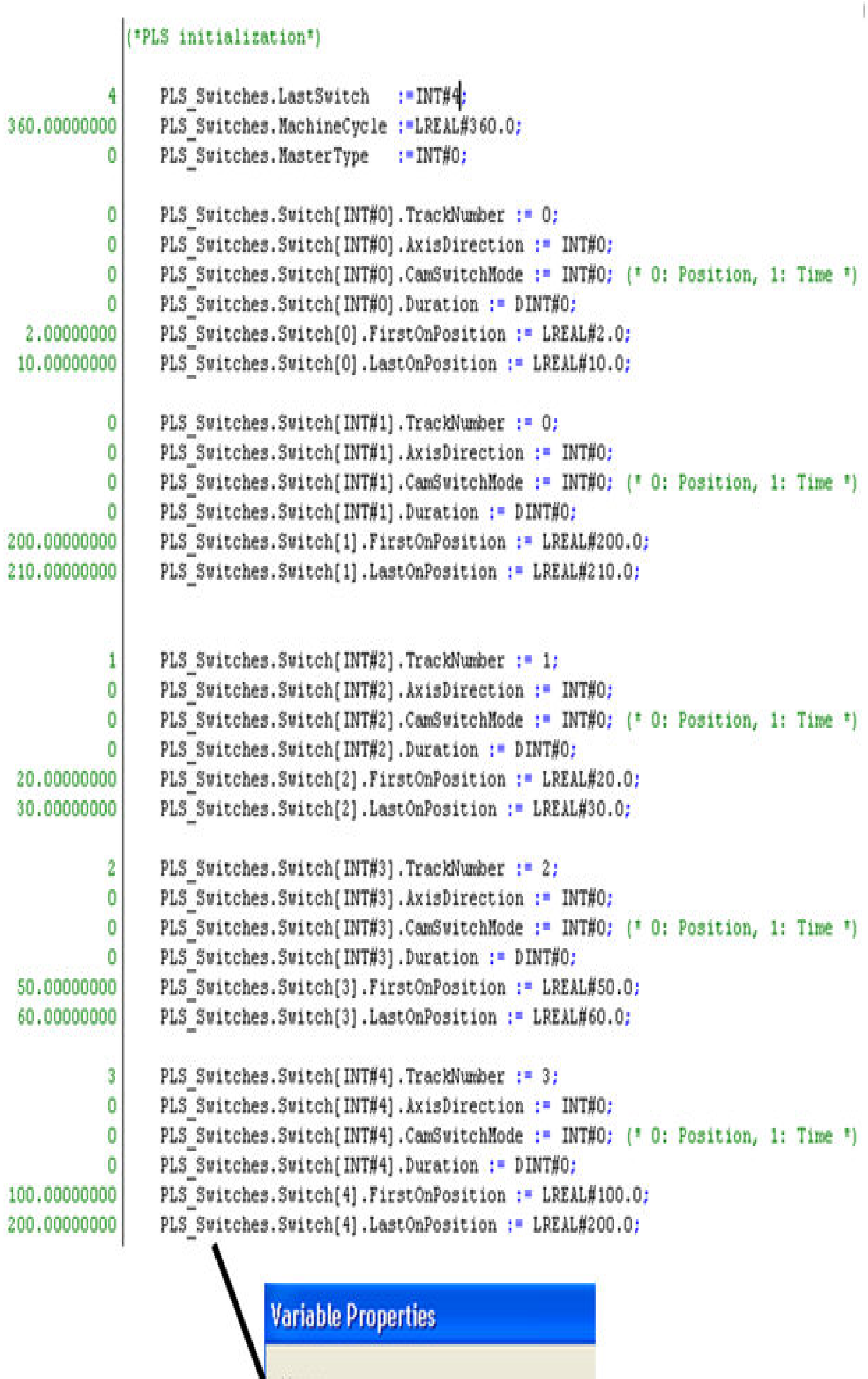
Track 3 has 1 switch associated with it

Switch 4: On Position : 100 degrees

Off Position : 200 degrees

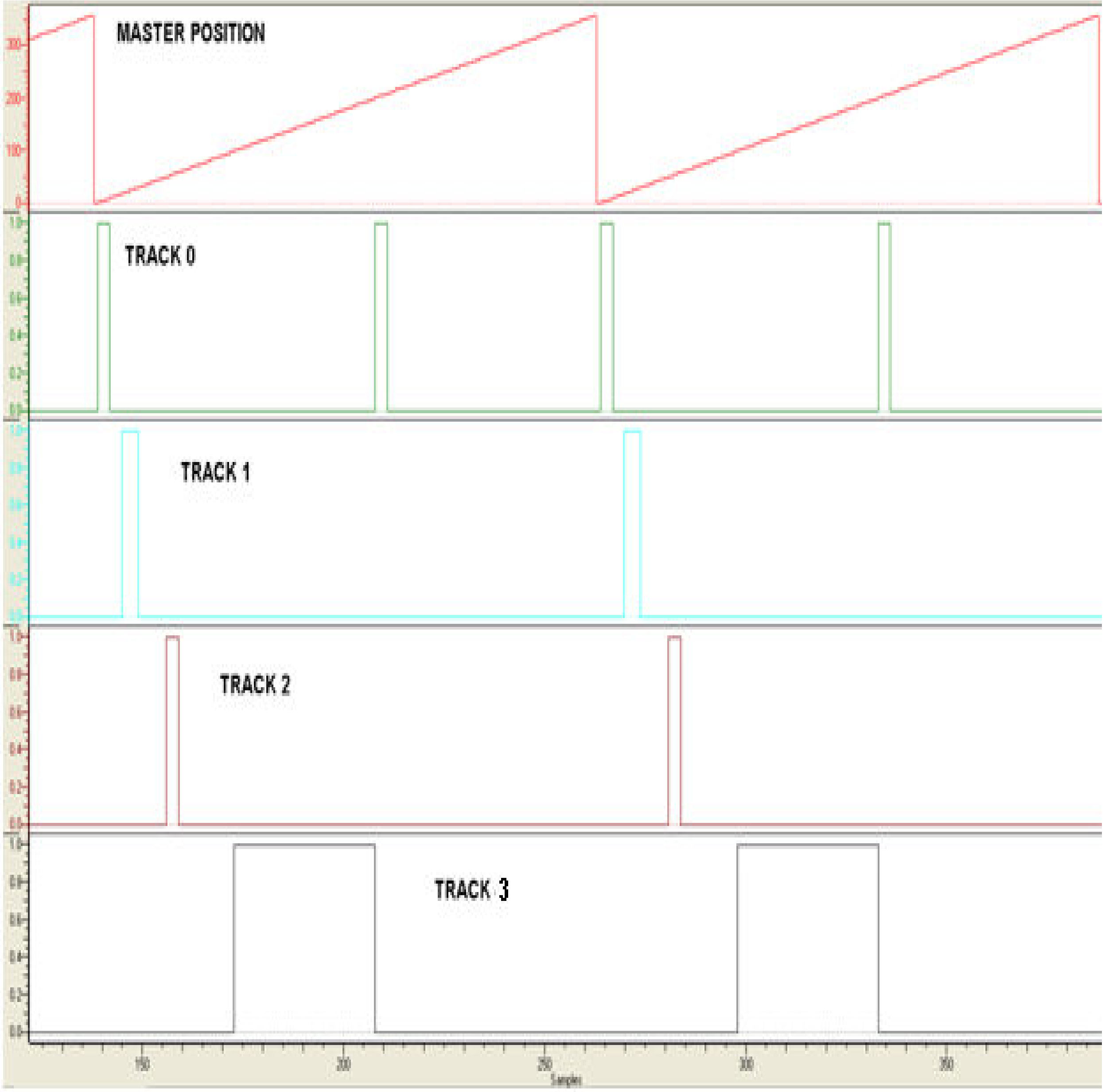


The switches can be defined and initialized as follows:



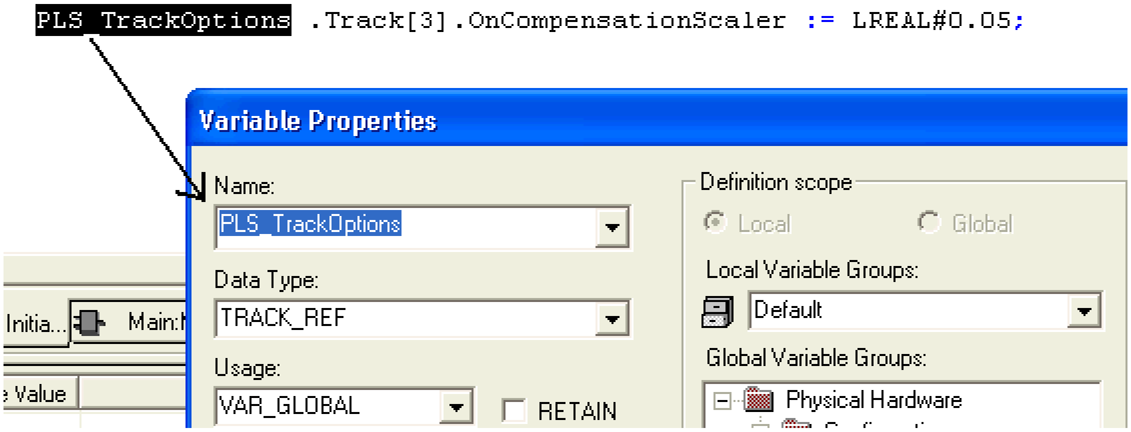
```
                (*PLS initialization*)

           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;
```

Once the Y_DgitalCamSwitch 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_TrackOptions .Track[3].OnCompensationScaler := LREAL#0.05;
```

Variable Properties

Name: PLS_TrackOptions

Data Type: TRACK_REF

Usage: VAR_GLOBAL

□ RETAIN

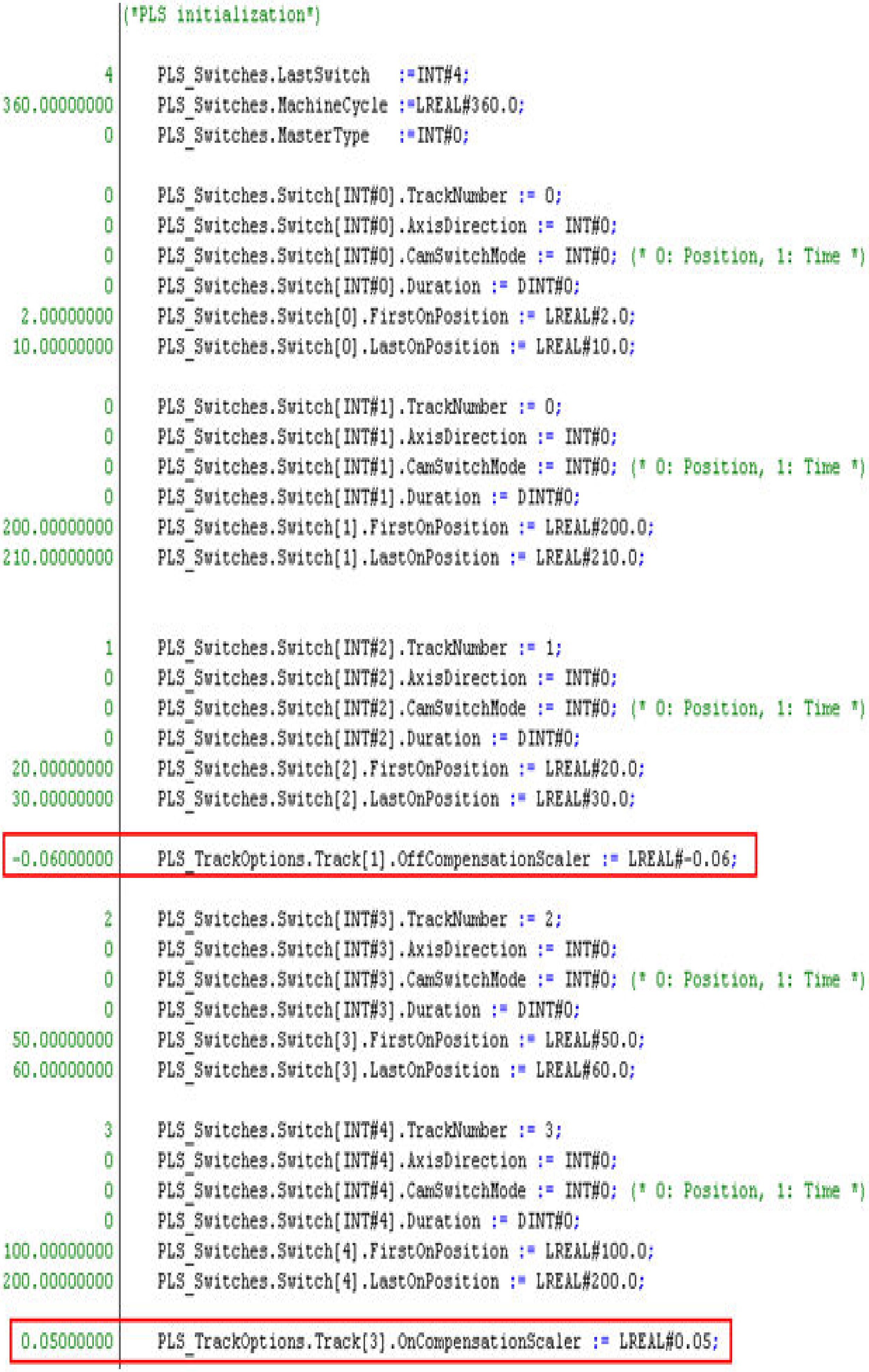
Definition scope

● Local ○ Global

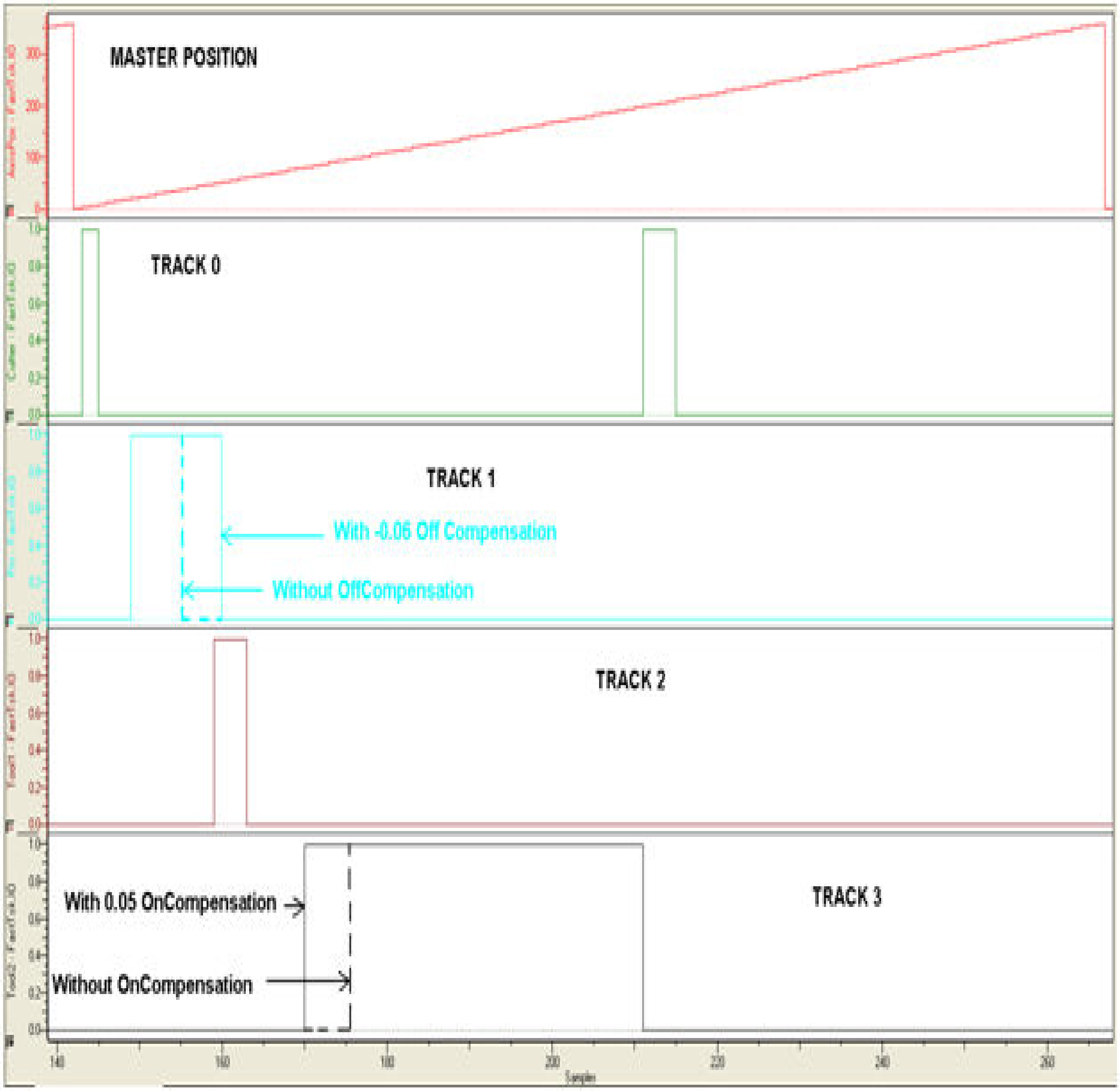
Local Variable Groups: Default

Global Variable Groups:

⊟ Physical Hardware



```
             (*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;
```


MASTER POSITION
0
100
200
300
TRACK 0
TRACK 1
With -0.06 Off Compensation
Without OffCompensation
TRACK 2
TRACK 3
With 0.05 OnCompensation
Without OnCompensation
0.0
0.2
0.4
0.6
0.8
1.0
140
160
180
200
220
240
260
Samples

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.

Yaskawa Toolbox

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

Getting Started with Yaskawa Toolbox

Requirements for v204

To use the Yaskawa Toolbox, your project must also contain the following:

Firmware libraries:

- YDeviceComm
- PROCONOS

User libraries:

- None

Yaskawa Revision History

Current Version:

(****************************** 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.

Previous Versions:

(***************************** 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 sad 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 Toolboox

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:

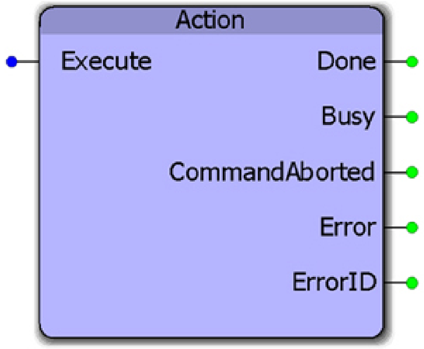
Dummy FB to show simulation of the template function blocks.

Toolbox Help Documentation

Help version created 1/25/2017

YASKAWA

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.	

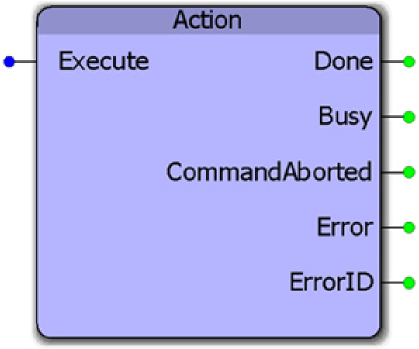
Error Description

This function provides no Errors.

Example

See the Enable_F_Template, Enable_ST_Template, Execute_FB_Template, and Execute_ST_Template function blocks.

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.	

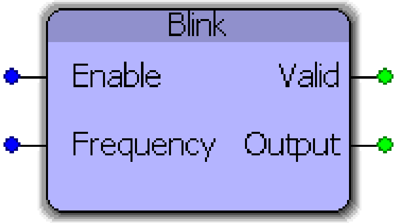
Error Description

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

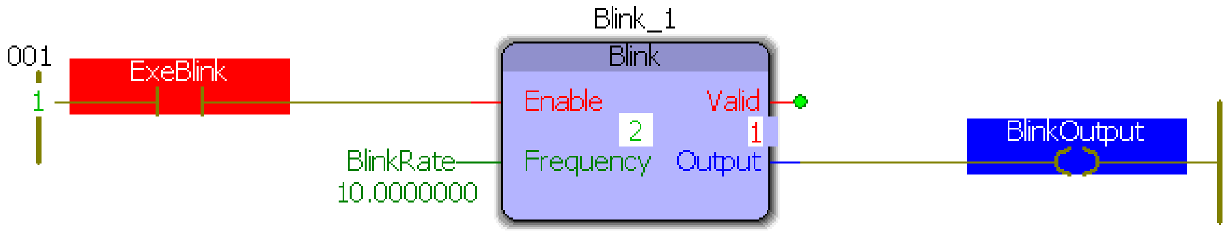
Parameter		Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Frequency	REAL	The cycle rate in Hertz.	REAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates if the function is operating.	
V	Output	BOOL	Toggled at the specified frequency when the function is enabled.	

Error Description

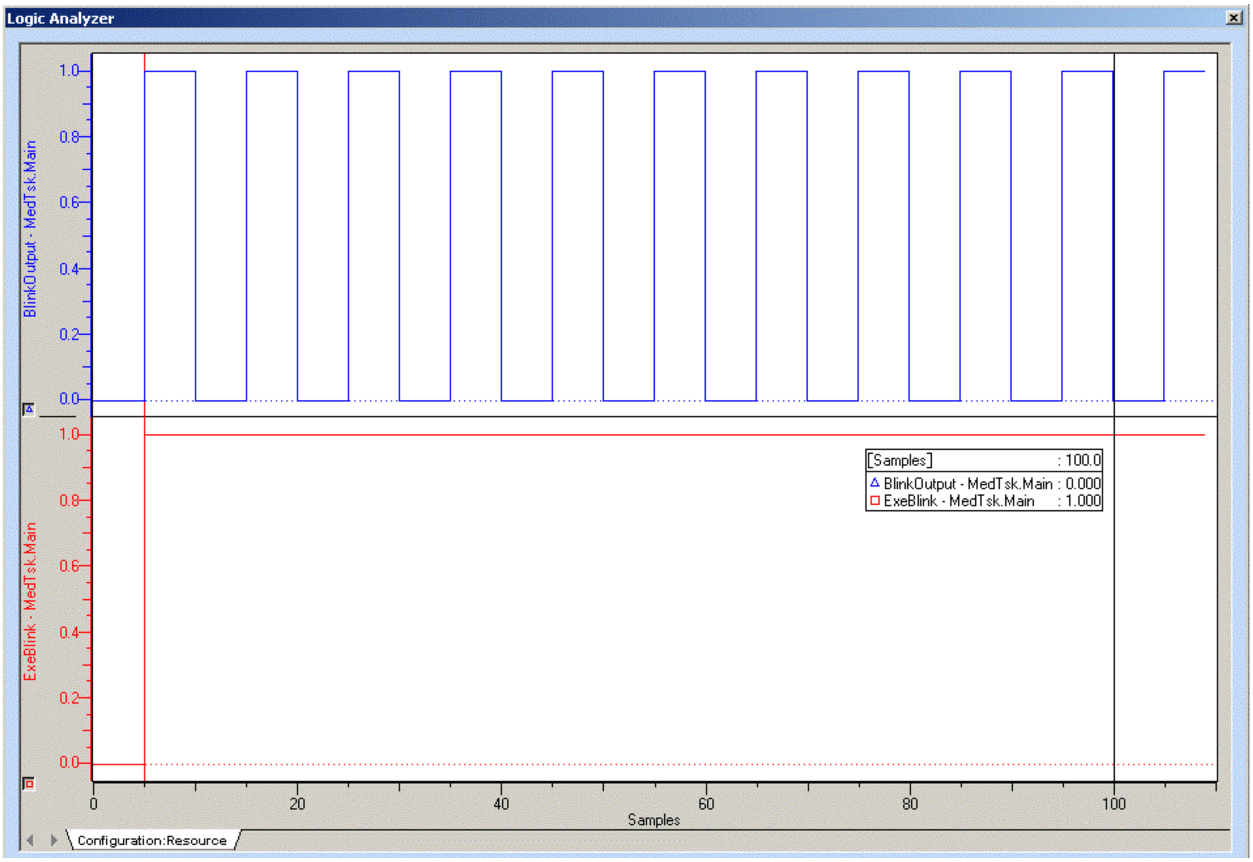
The valid output will be high if the function is operating. If Enable is held high and the Frequency is not greater than zero, the valid output will be low.

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.

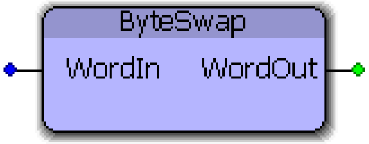


Logic Analyzer output:



```
[Samples]                   : 100.0
△ BlinkOutput - MedTsk.Main : 0.000
□ ExeBlink - MedTsk.Main    : 1.000
```

ByteSwap



This function block swaps the upper and lower byte of a word.

Library

Yaskawa Toolbox

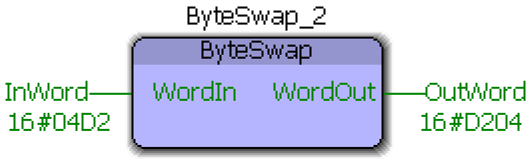
Parameters

Parameter		Data Type	Description	
VAR_INPUT				**Default**
V	WordIn	WORD	Input word	WORD#0
VAR_OUTPUT				
V	WordOut	WORD	Output word	

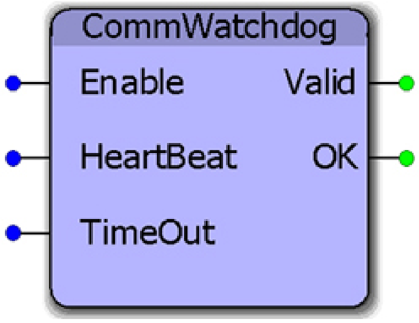
Error Description

This block will not produce any errors.

Example:



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

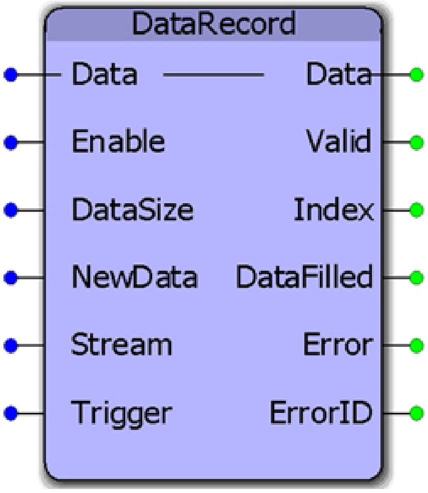
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	HeartBeat	DINT	Value that the master changes and sends to the MPiec controller.	DINT#0
V	WatchDog	DINT	The HeartBeat input 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				
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.	

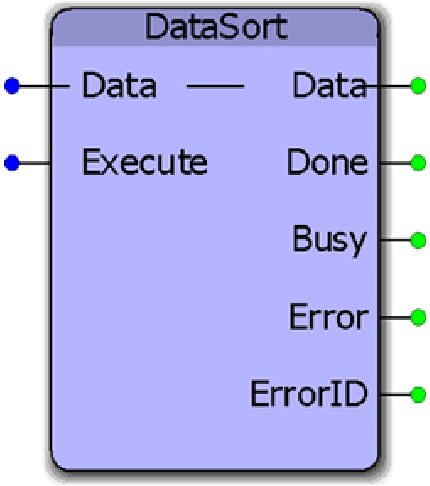
B	Error	BOOL	Set high if an error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
E	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Error Description

ErrorID	Meaning
0	No error.
10024	DataSize must be greater than zero.

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.	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.	

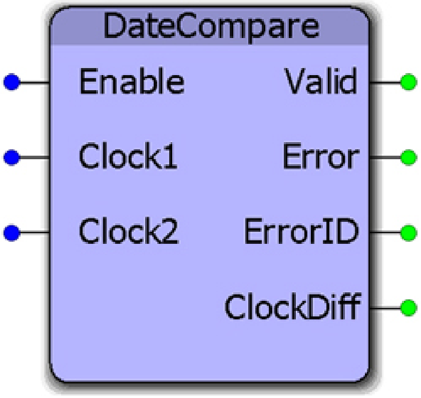
Notes

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

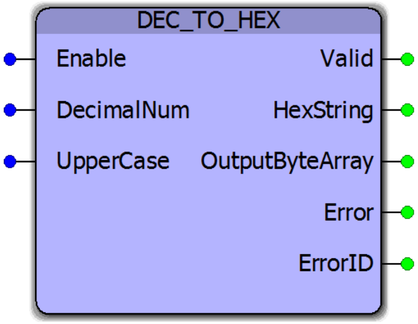
Parameters

*	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 first (older) real time clock value.	All zeros in structure
V	Clock2	RTCStruct	The second (newer) 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 and 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

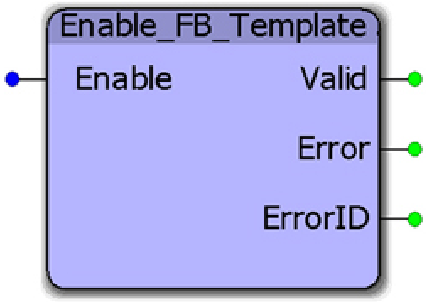
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	DecimalNum	DINT	Input value to be converted.	DINT#0
V	UpperCase	BOOL	If True, the output string will contain upper case ASCII characters. If False, the output string will contain lower case ASCII characters.	FALSE
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 will output ErrroIDs from the BUF_TO_STRING function.

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.	

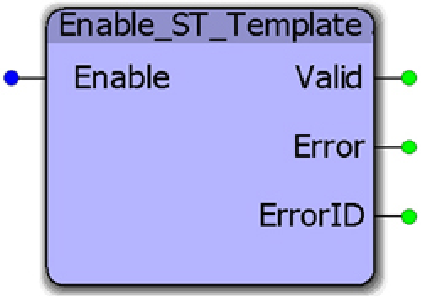
Error Description

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.	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.	

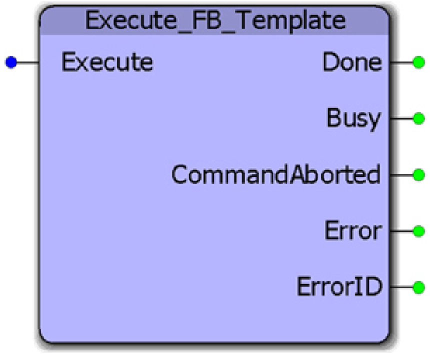
Error Description

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.

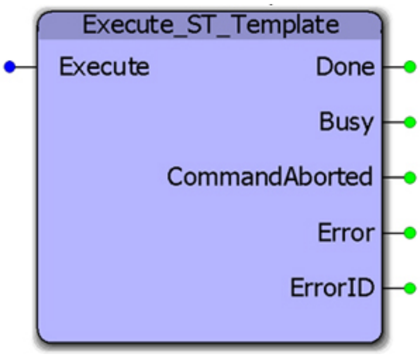
Error Description

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.

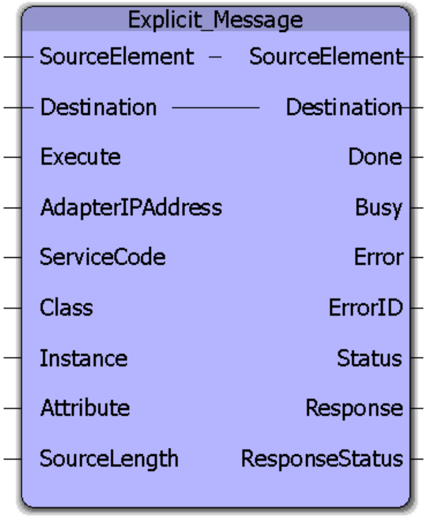
Error Description

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

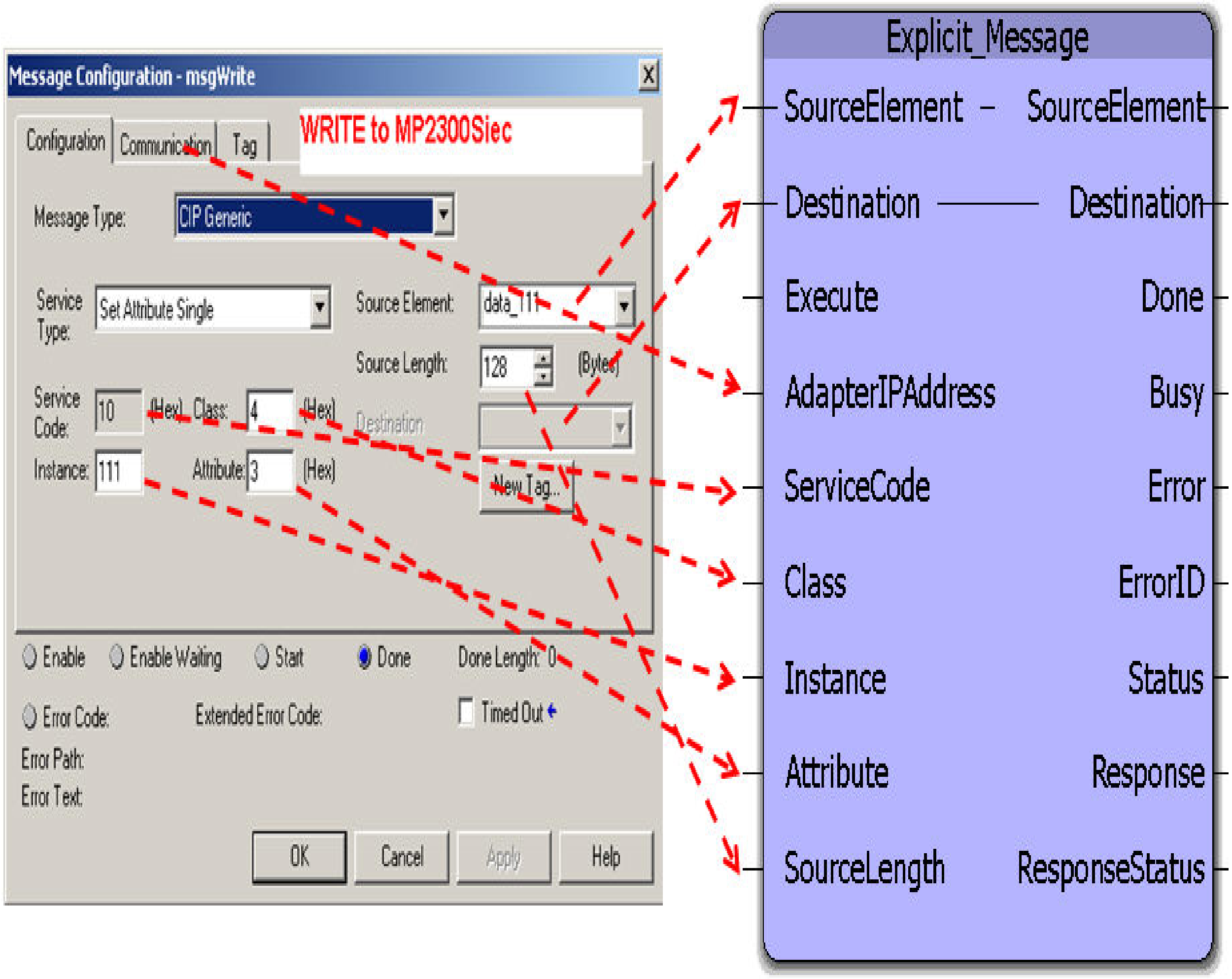
Parameters

*	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**
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)	BYTE#0

			documentation.	
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.



Message Type: CIP Generic

Service Type: Set Attribute Single

Source Element: data_111

Source Length: 128 (Bytes)

Service Code: 10 (Hex) Class: 4 (Hex)

Instance: 111 Attribute: 3 (Hex)

Destination

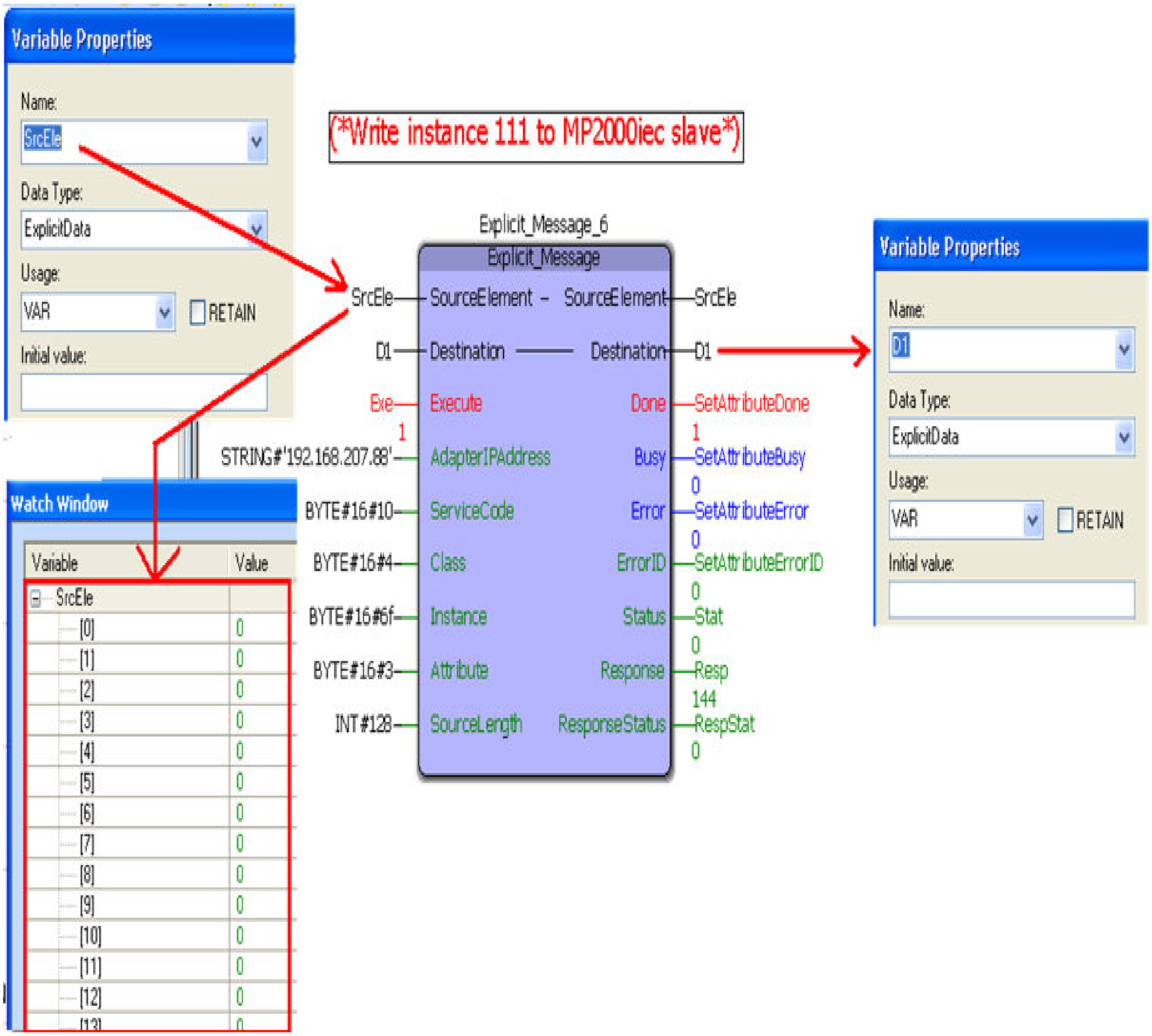
Error Description

ErrorID	Meaning
0	No error.
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.
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.

Example 1

Set Single Attribute



Variable Properties

Name: SrcEle

Data Type: ExplicitData

Usage: VAR □ RETAIN

Initial value:

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

(*Write instance 111 to MP2000iec slave*)

Variable Properties

Name: D1

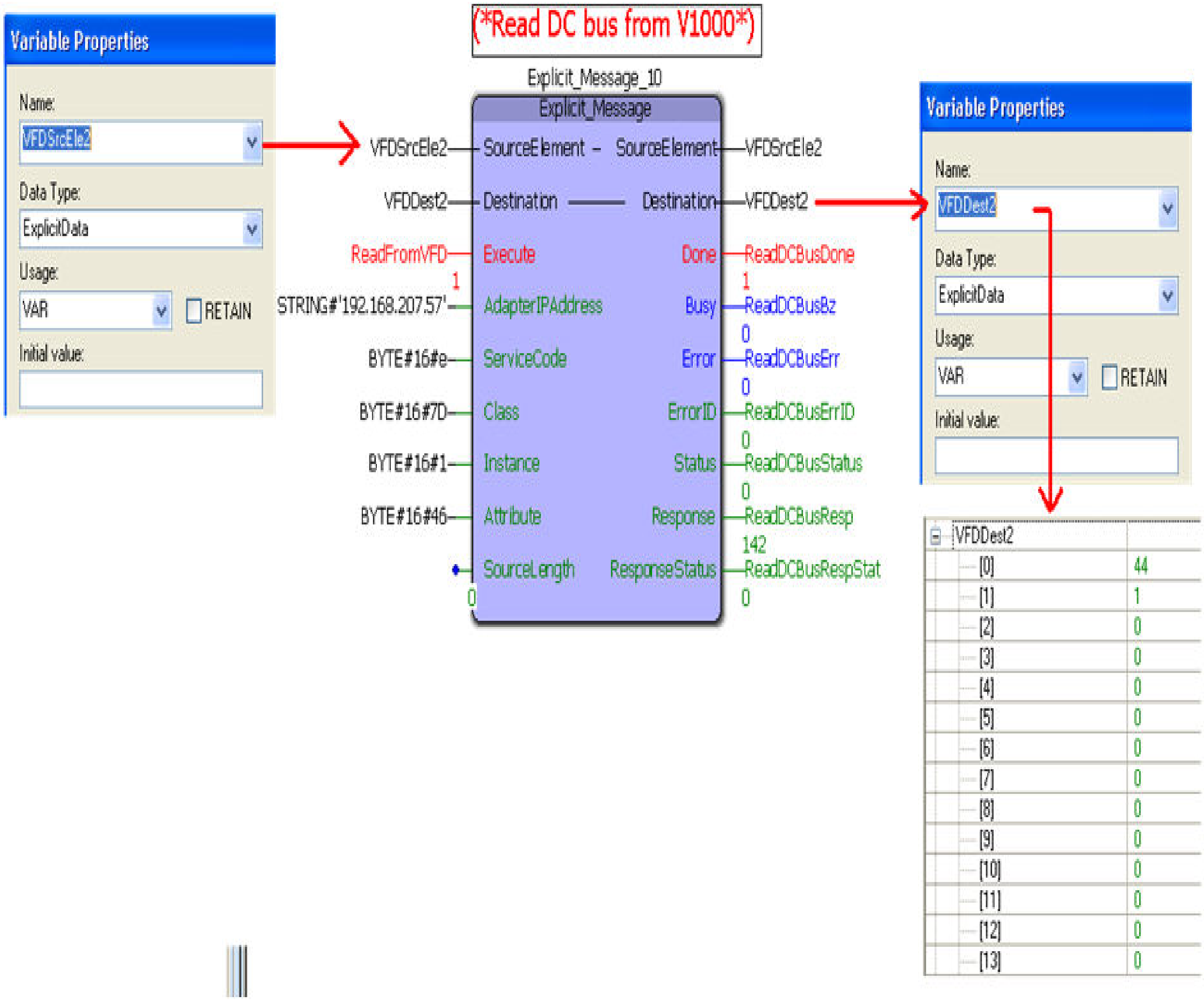
Data Type: ExplicitData

Usage: VAR □ RETAIN

Initial value:

Example 2

Get Attribute single



(*Read DC bus from V1000*)

Variable Properties

Name: VFDSrcEle2

Data Type: ExplicitData

Usage: VAR □ RETAIN

Initial value:

Explicit_Message_10

Explicit_Message

```
VFDSrcEle2—SourceElement - SourceElement—VFDSrcEle2
VFDDest2—Destination ——— Destination—VFDDest2
ReadFromVFD—Execute Done—ReadDCBusDone
1 1
STRING#'192.168.207.57'—AdapterIPAddress Busy—ReadDCBusBz
0
BYTE#16#e—ServiceCode Error—ReadDCBusErr
0
BYTE#16#7D—Class ErrorID—ReadDCBusErrID
0
BYTE#16#1—Instance Status—ReadDCBusStatus
0
BYTE#16#46—Attribute Response—ReadDCBusResp
142
•—SourceLength ResponseStatus—ReadDCBusRespStat
0 0
```

Variable Properties

Name: VFDDest2

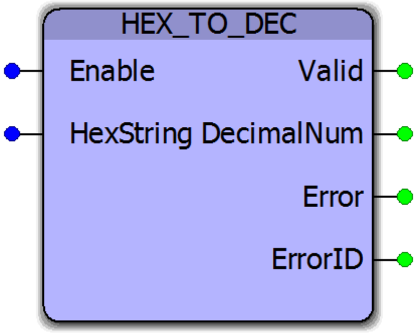
Data Type: ExplicitData

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

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#'FFFFFFFF'	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

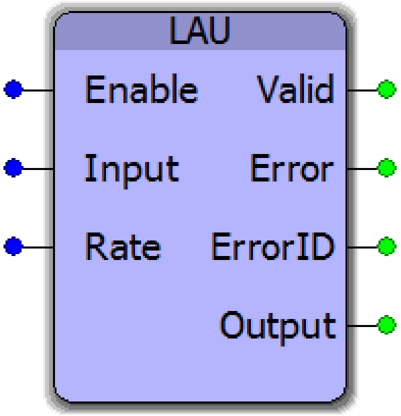
- Converts both upper and lower case ASCII characters to a DINT value.

Error Description

ErrorID	Meaning
0	No error.

This function may output ErrorIDs from the STRING_TO_BUF Status output.

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

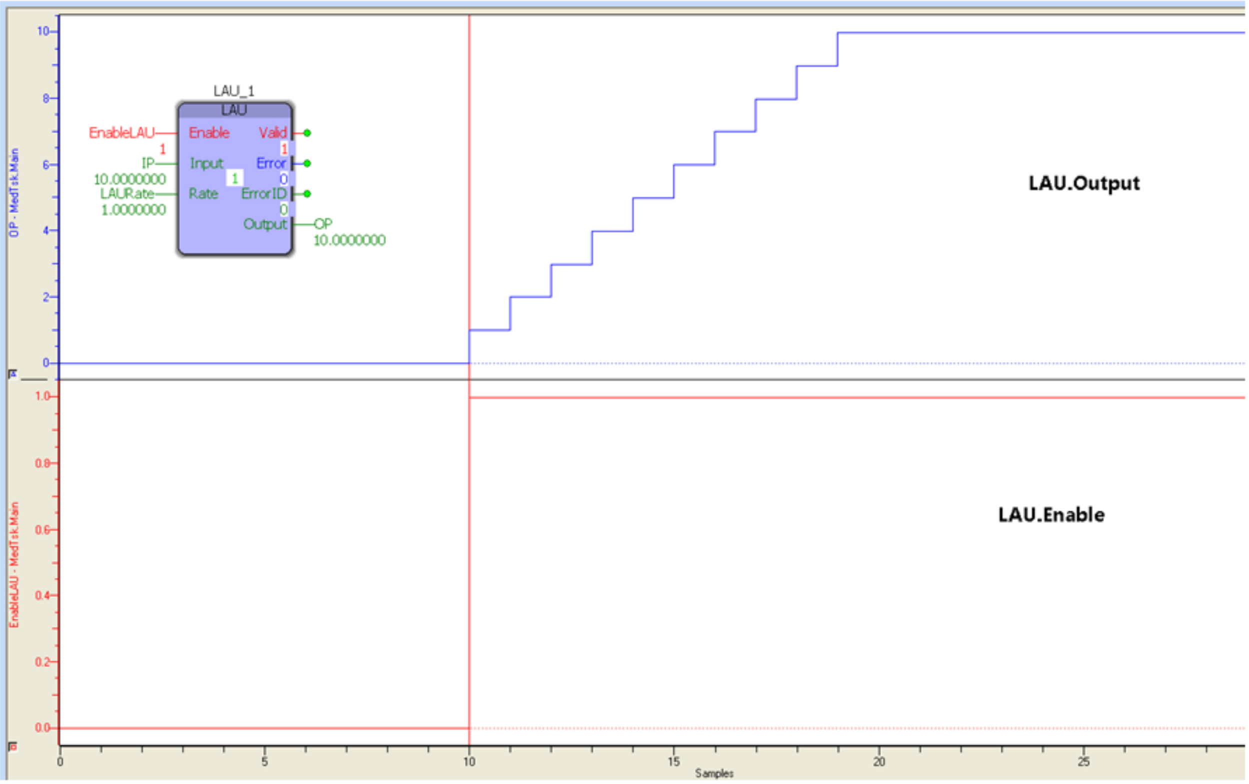
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Input	LREAL	Target value.	LREAL#0.0
V	Rate	LREAL	Acceleration/Deceleration per scan. The time required for the Output to 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

ErrorID	Meaning
10093	Rate cannot be less than or equal to zero.

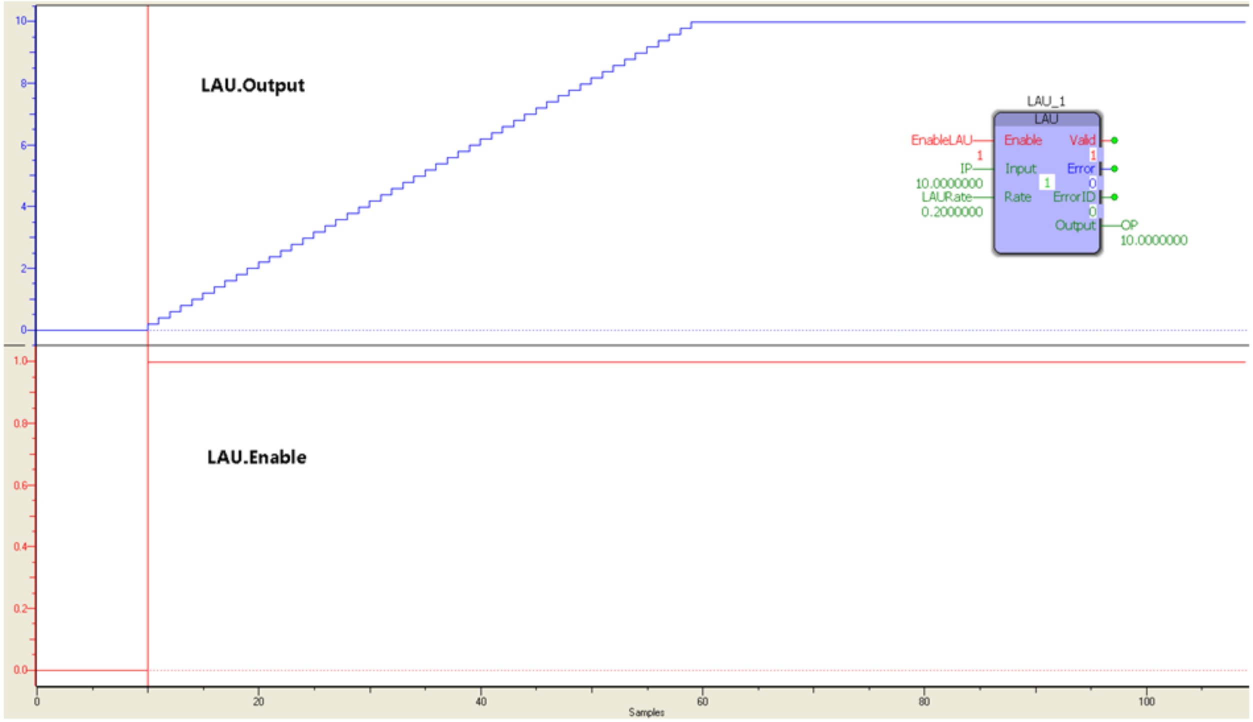
Example 1

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.

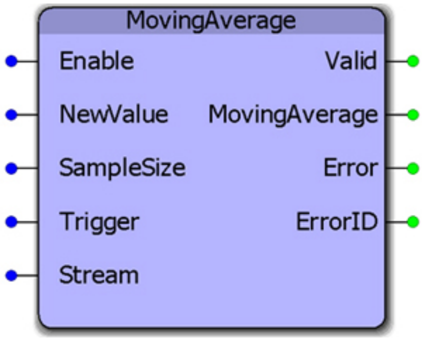


Example 2

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.


10
8
6
4
2
0
LAU.Output
LAU_1
LAU
EnableLAU
1
Enable
Valid
1
IP
10.0000000
Input
1
Error
0
LAURate
0.2000000
Rate
ErrorID
0
Output
OP
10.0000000
1.0
0.8
0.6
0.4
0.2
0.0
LAU.Enable
0
20
40
60
80
100
Samples

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 value 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

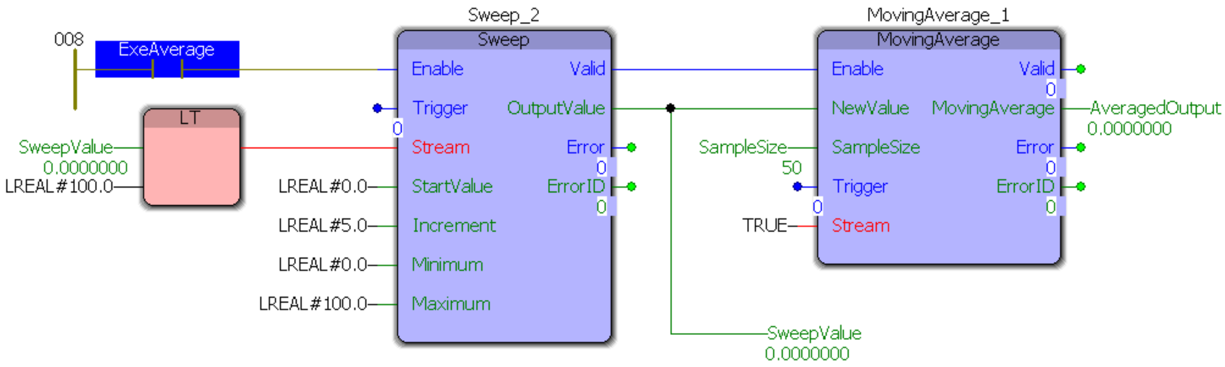
- See Yaskawa's Youtube webinar - MPiec Web Tension Control Applications for more info on using this function block.

Error Description

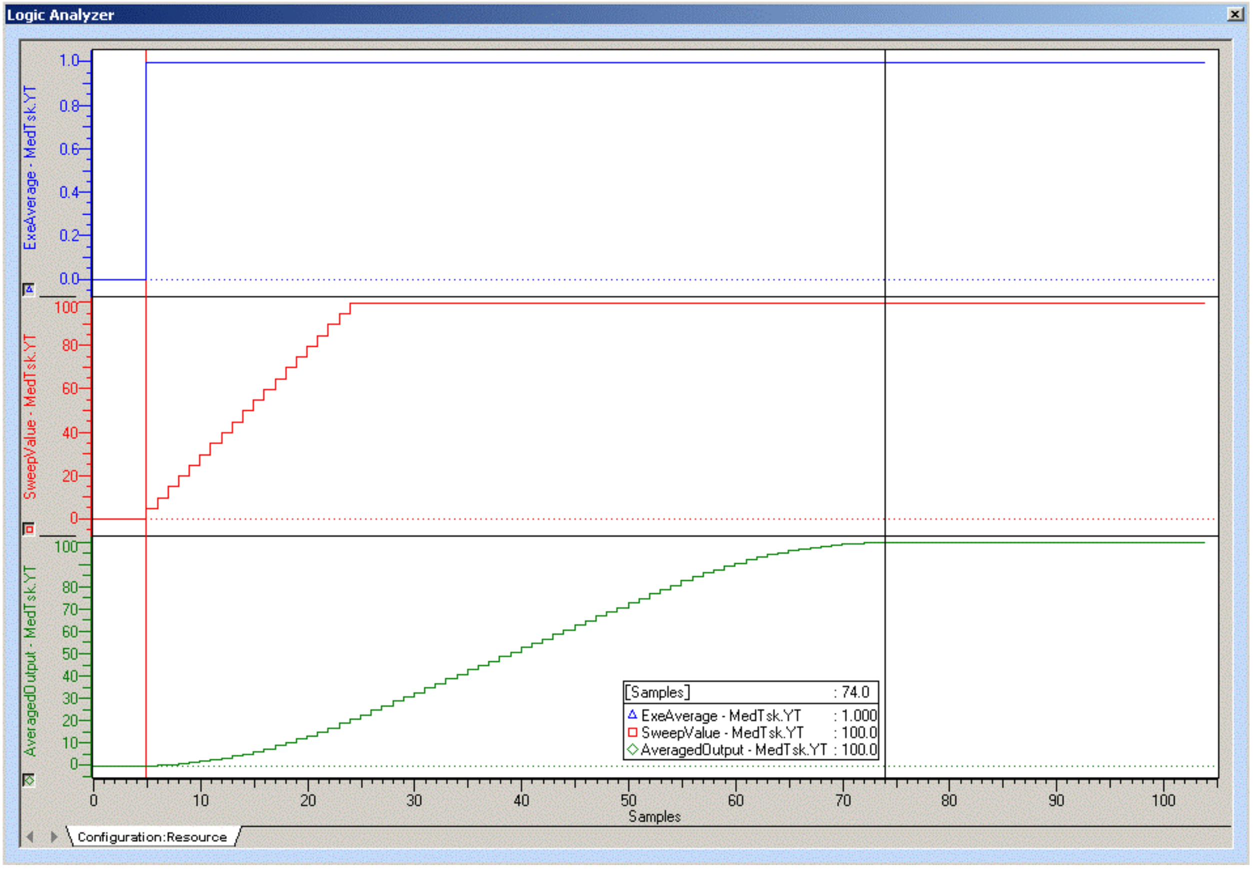
ErrorID	Meaning
0	No error.
10024	DataSize must be greater than zero.

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.

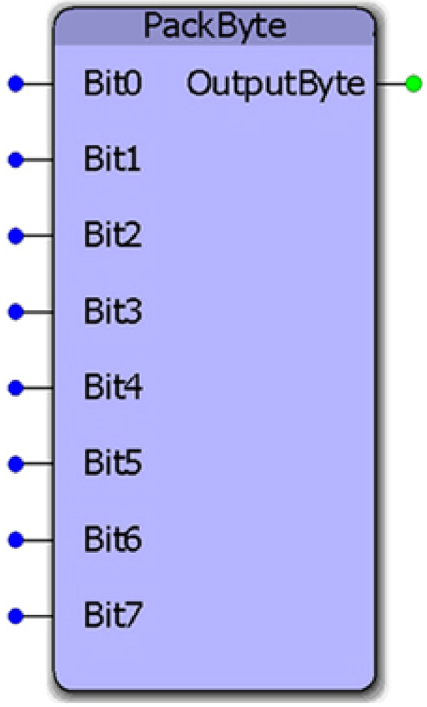


The Moving average function has a sample size of 50 which means that if Sweep reaches its maximum value after 19 cycles, MovingAverage will output the maximum value after 69 cycles. By looking at the Logic Analyzer plot below, notice there is a 5 cycle pre-record that must be taken in to account: 74 - 5 = 69 cycles.



[Samples]	: 74.0
△ ExeAverage - MedTsk.YT	: 1.000
□ SweepValue - MedTsk.YT	: 100.0
◇ AveragedOutput - MedTsk.YT	: 100.0

PackByte



This function block converts 8 BOOL inputs to a single BYTE output.

Library

Yaskawa Toolbox

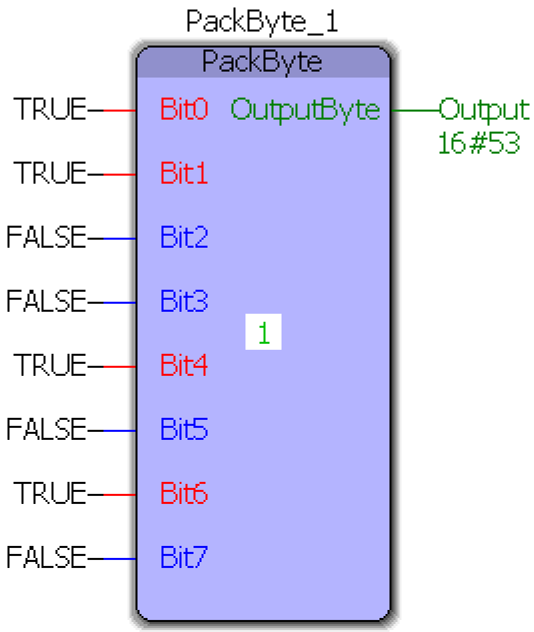
Parameters

*	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
VAR_OUTPUT				
V	OutputByte	BYTE	Resulting byte of the input bits	

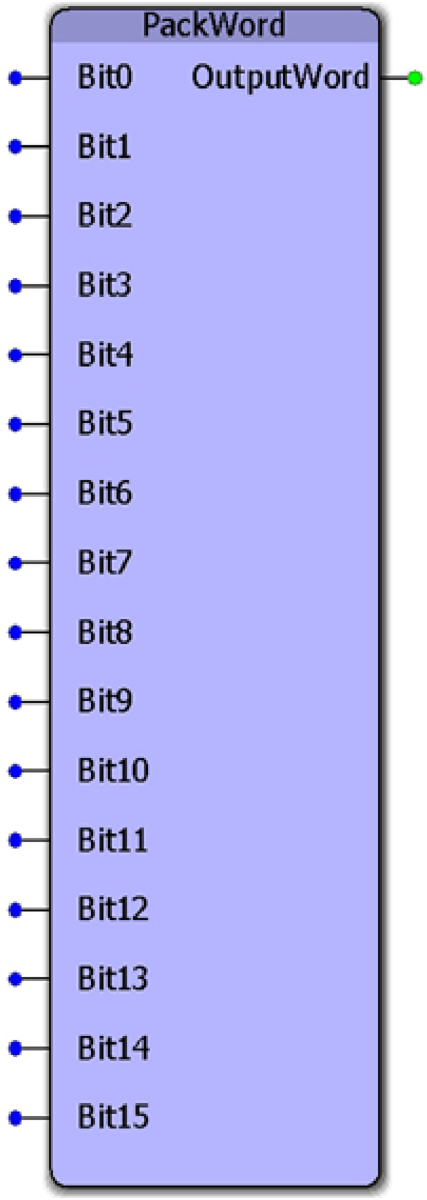
Error Description

No errors will be generated

Example



PackWord



This function block converts 16 Boolean inputs to a single WORD output.

Library

Yaskawa Toolbox

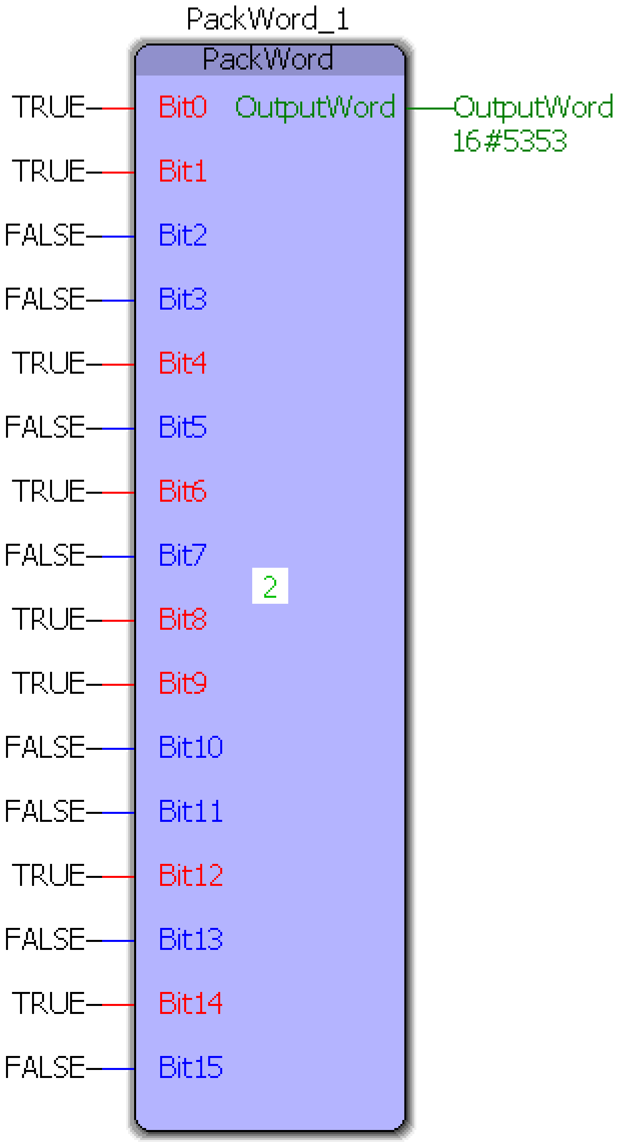
Parameters

*	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	

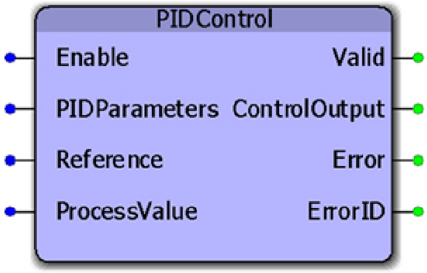
Error Description

No errors will be generated

Example


PackWord_1
PackWord
TRUE
Bit0
OutputWord
OutputWord
16#5353
TRUE
Bit1
FALSE
Bit2
FALSE
Bit3
TRUE
Bit4
FALSE
Bit5
TRUE
Bit6
FALSE
Bit7
2
TRUE
Bit8
TRUE
Bit9
FALSE
Bit10
FALSE
Bit11
TRUE
Bit12
FALSE
Bit13
TRUE
Bit14
FALSE
Bit15

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:

$$\frac{Y}{X}.Kp + \frac{Ki}{Ti.S} = Kd.Td.S$$

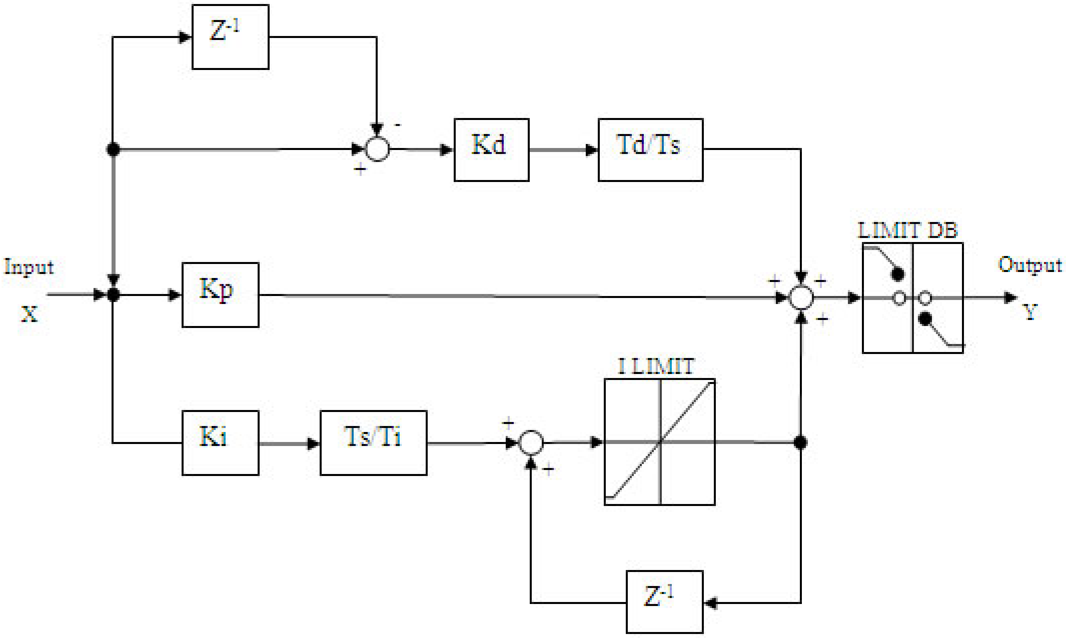
$$\frac{Y}{X} = Kp + Kd.Td.S$$

X: deviation input value; Y: output value

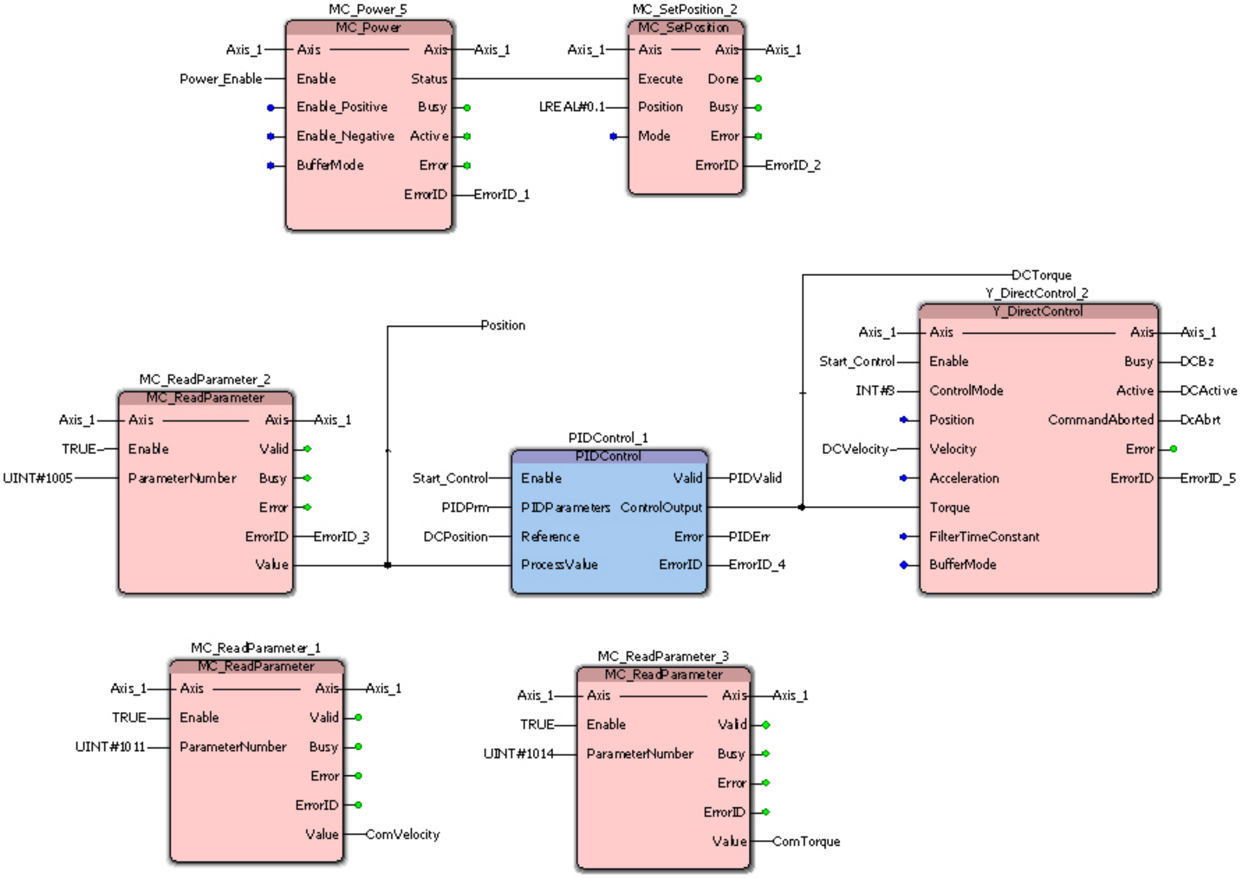
The following operation is performed within the PID instruction:

$$Y = Kp.X + \left\{ \frac{Ki.X + IREM}{\frac{Ti}{Ts}} + Yi' \right\} + Kd.(X - X').\frac{Td}{Ts}$$

X’: previous input value;   Yi’: previous I output value;   Ts: scan time set value

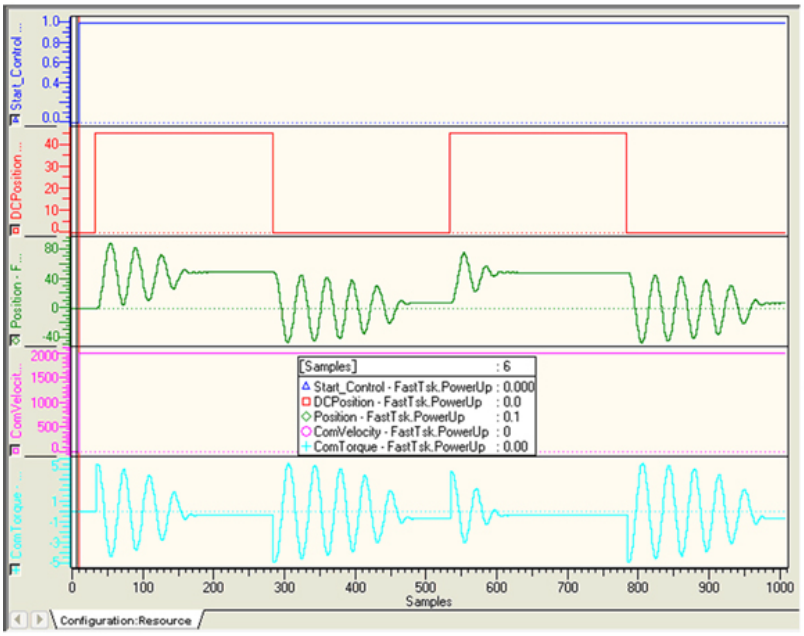


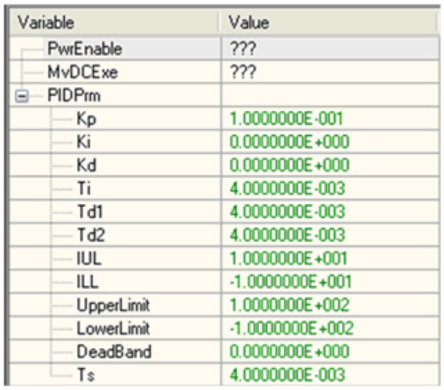
1. An example controlling a servo in torque mode:



The following series of graphs show changes made to the PID gains to minimize error:

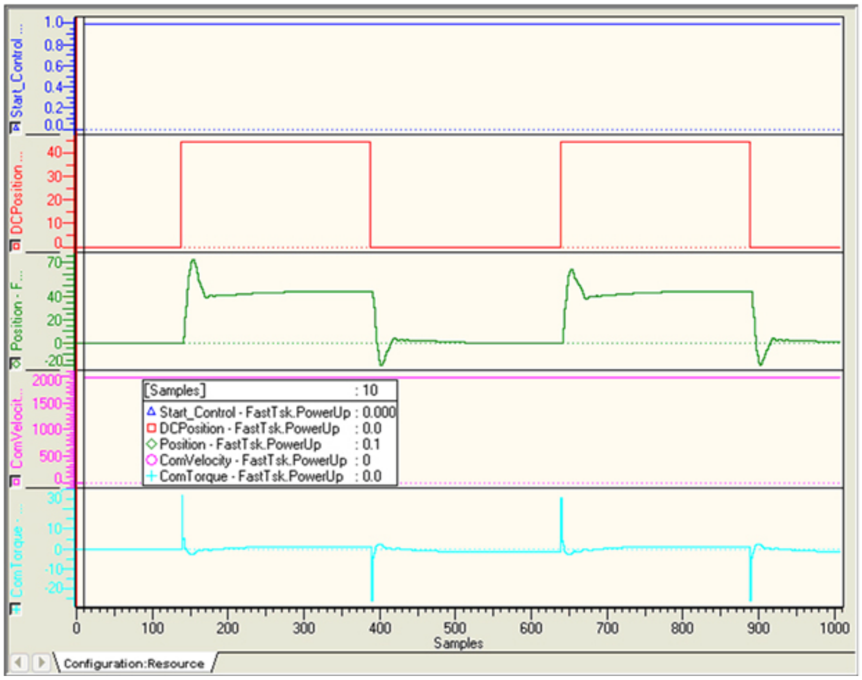
a. Proportional Control Only. Severe oscillation:

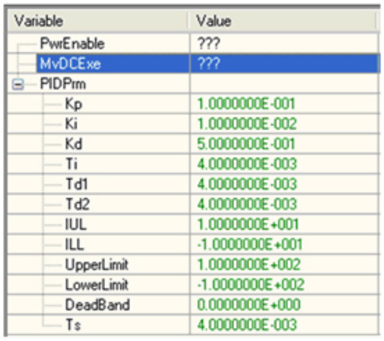




Variable	Value
PwrEnable	???
MvDCExe	???
PIDPrm	
Kp	1.0000000E-001
Ki	0.0000000E+000
Kd	0.0000000E+000
Ti	4.0000000E-003
Td1	4.0000000E-003
Td2	4.0000000E-003
IUL	1.0000000E+001
ILL	-1.0000000E+001
UpperLimit	1.0000000E+002
LowerLimit	-1.0000000E+002
DeadBand	0.0000000E+000
Ts	4.0000000E-003

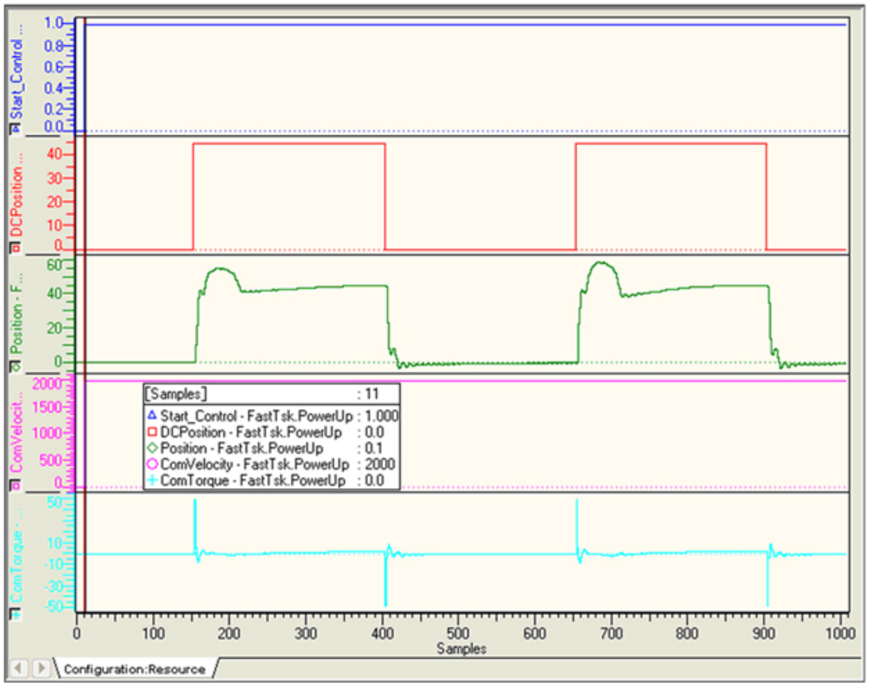
b. PID Control. Derivative helps to control oscillation:



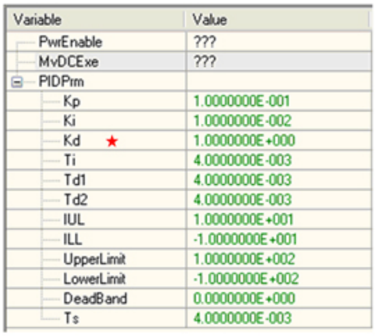


Variable	Value
PwrEnable	???
MvDCExe	???
PIDPrm	
Kp	1.0000000E-001
Ki	1.0000000E-002
Kd	5.0000000E-001
Ti	4.0000000E-003
Td1	4.0000000E-003
Td2	4.0000000E-003
IUL	1.0000000E+001
ILL	-1.0000000E+001
UpperLimit	1.0000000E+002
LowerLimit	-1.0000000E+002
DeadBand	0.0000000E+000
Ts	4.0000000E-003

c. PID Control – Increasing the derivative gain:

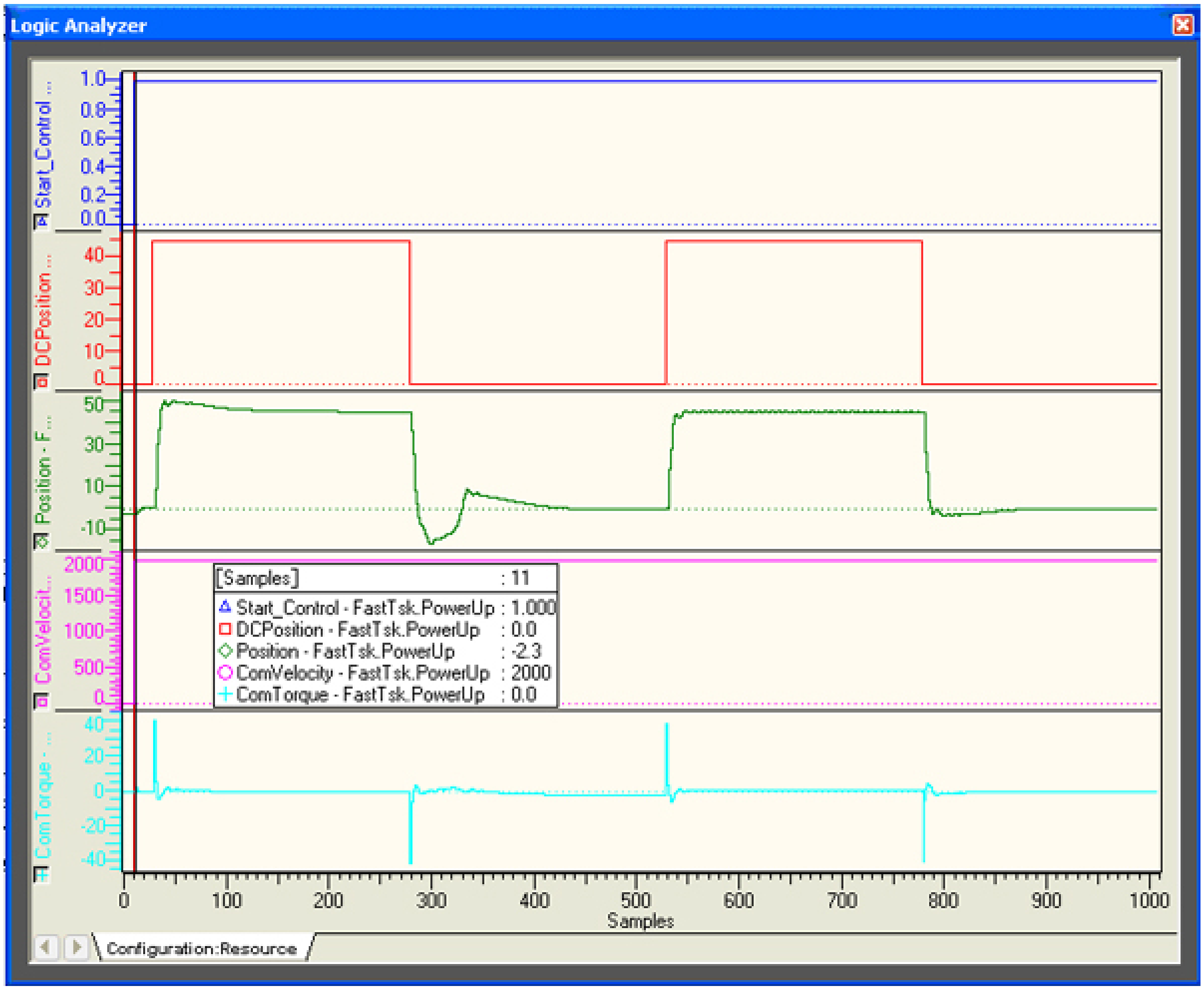


```
[Samples]                               : 11
△ Start_Control - FastTsk.PowerUp : 1.000
□ DCPosition - FastTsk.PowerUp     : 0.0
◇ Position - FastTsk.PowerUp       : 0.1
○ ComVelocity - FastTsk.PowerUp   : 2000
+ ComTorque - FastTsk.PowerUp     : 0.0
```

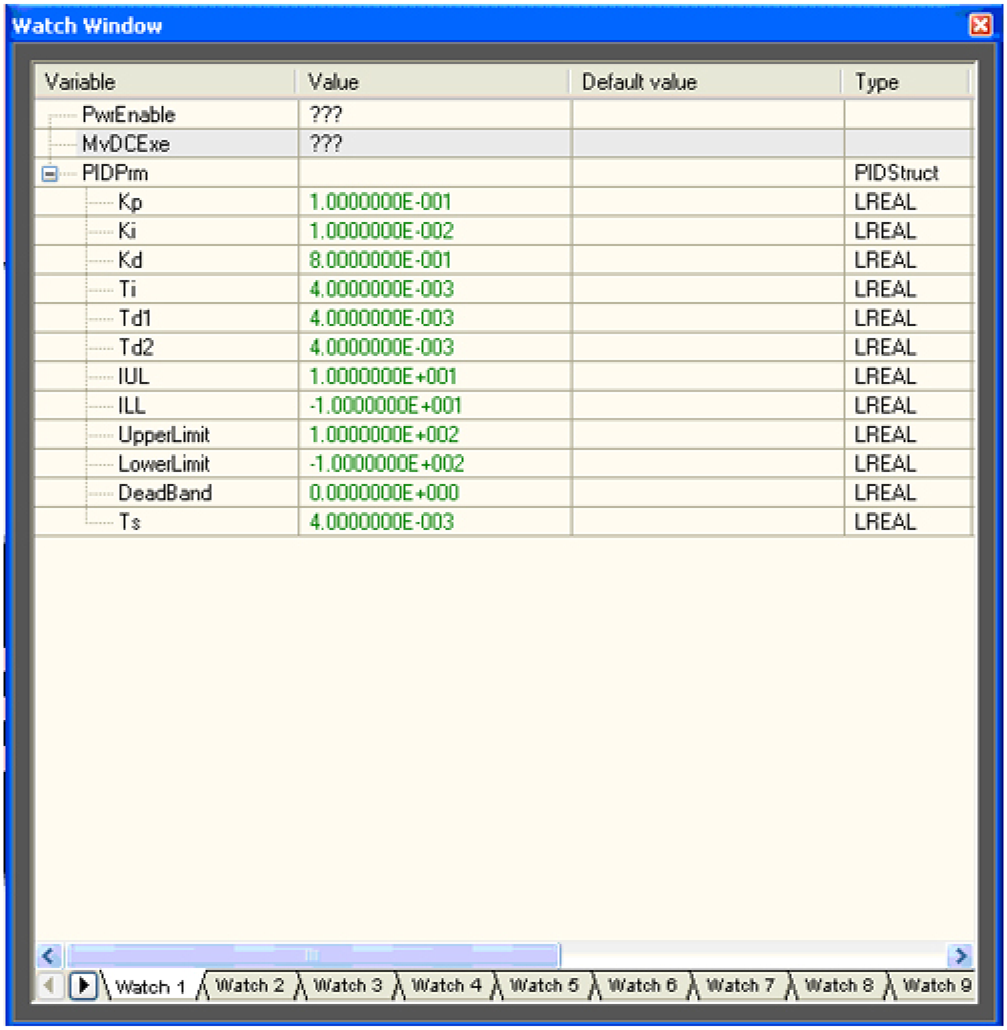


Variable	Value
PwrEnable	???
MvDCExe	???
⊟ PIDPrm	
Kp	1.0000000E-001
Ki	1.0000000E-002
Kd ★	1.0000000E+000
Ti	4.0000000E-003
Td1	4.0000000E-003
Td2	4.0000000E-003
IUL	1.0000000E+001
ILL	-1.0000000E+001
UpperLimit	1.0000000E+002
LowerLimit	-1.0000000E+002
DeadBand	0.0000000E+000
Ts	4.0000000E-003

d. Further increase in the derivative gain:

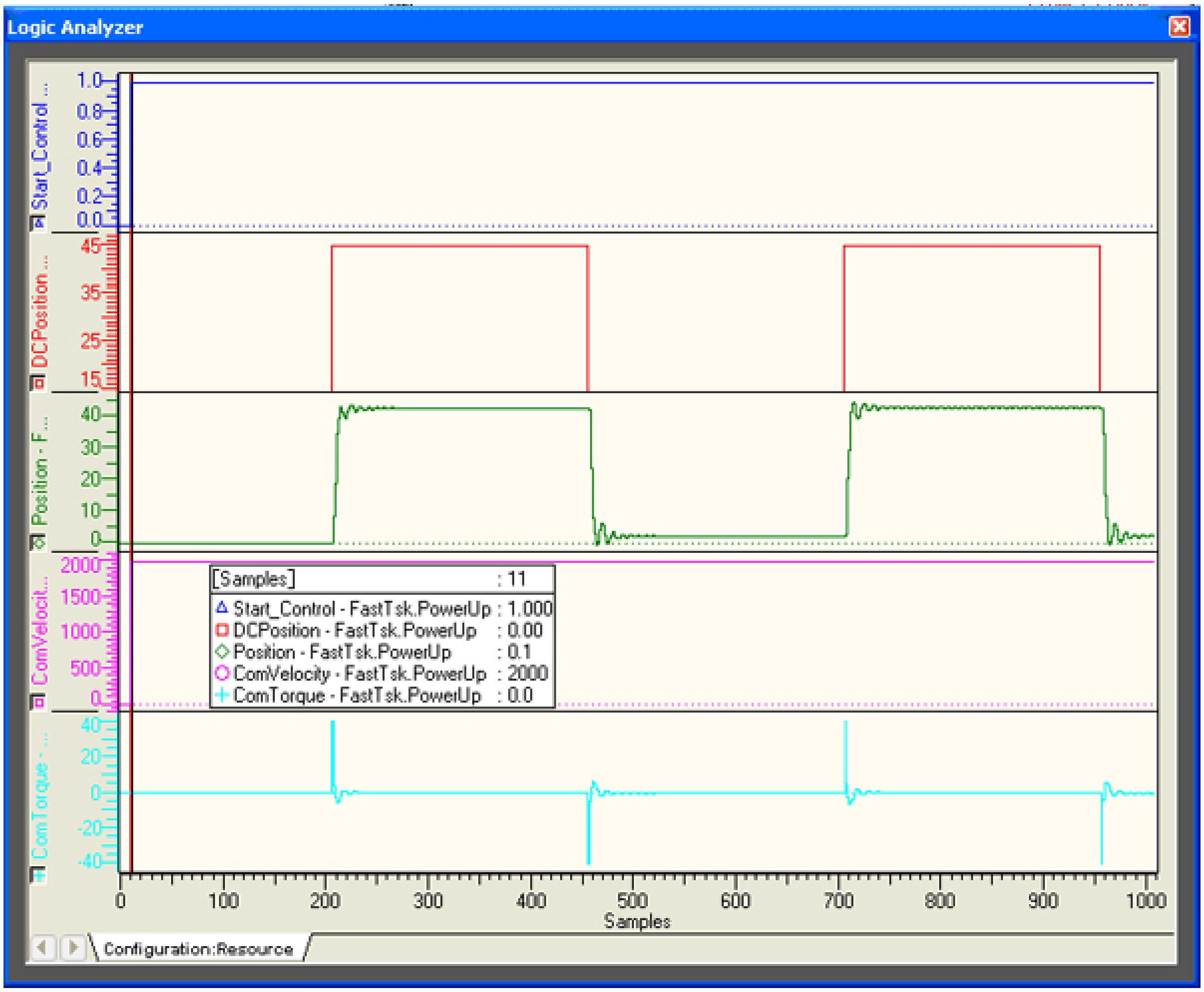


```
[Samples]                               : 11
△ Start_Control - FastTsk.PowerUp : 1.000
□ DCPosition - FastTsk.PowerUp     : 0.0
◇ Position - FastTsk.PowerUp       : -2.3
○ ComVelocity - FastTsk.PowerUp   : 2000
+ ComTorque - FastTsk.PowerUp     : 0.0
```

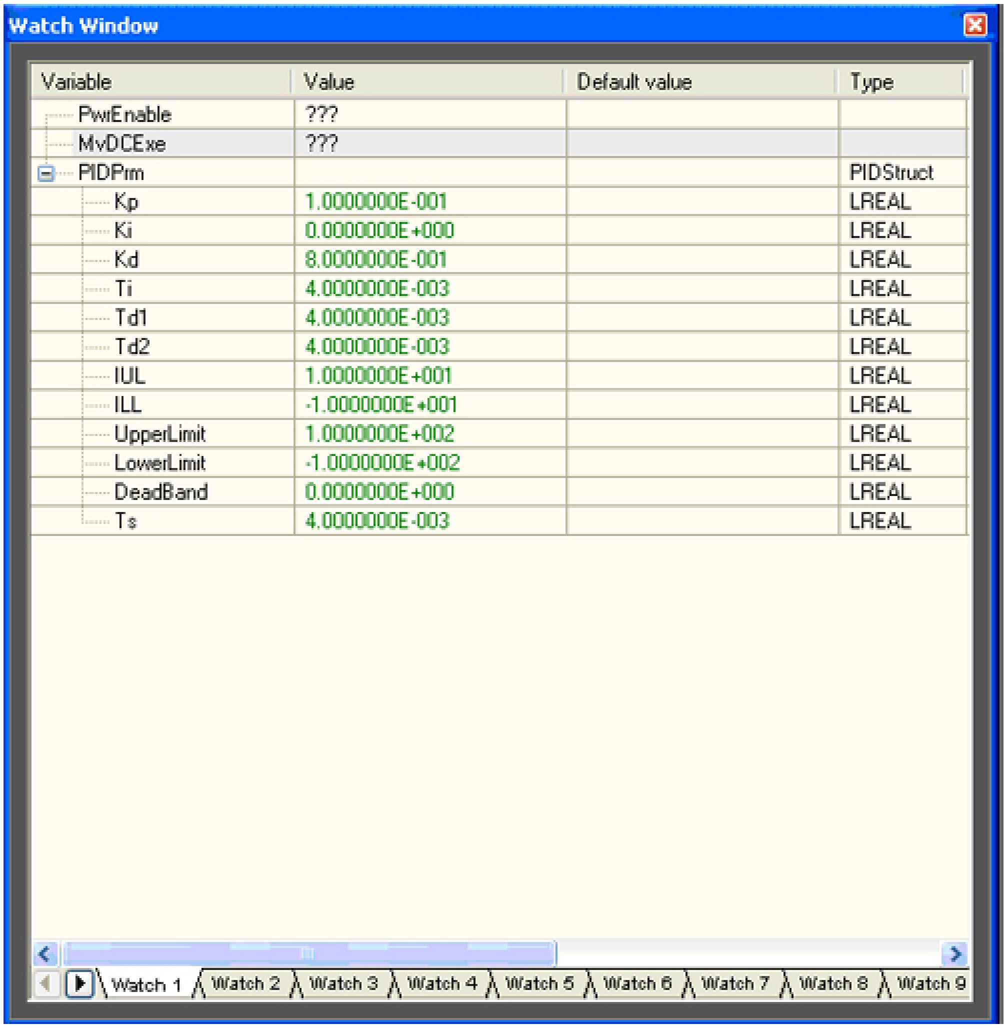


Variable	Value	Default value	Type
PwEnable	???		
MvDCExe	???		
PIDPrm			PIDStruct
Kp	1.0000000E-001		LREAL
Ki	1.0000000E-002		LREAL
Kd	8.0000000E-001		LREAL
Ti	4.0000000E-003		LREAL
Td1	4.0000000E-003		LREAL
Td2	4.0000000E-003		LREAL
IUL	1.0000000E+001		LREAL
ILL	-1.0000000E+001		LREAL
UpperLimit	1.0000000E+002		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.

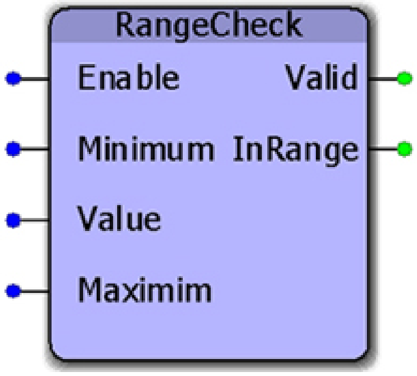


```
[Samples]                            : 11
△ Start_Control - FastTsk.PowerUp   : 1.000
□ DCPosition - FastTsk.PowerUp      : 0.00
◇ Position - FastTsk.PowerUp        : 0.1
○ ComVelocity - FastTsk.PowerUp     : 2000
+ ComTorque - FastTsk.PowerUp       : 0.0
```



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

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

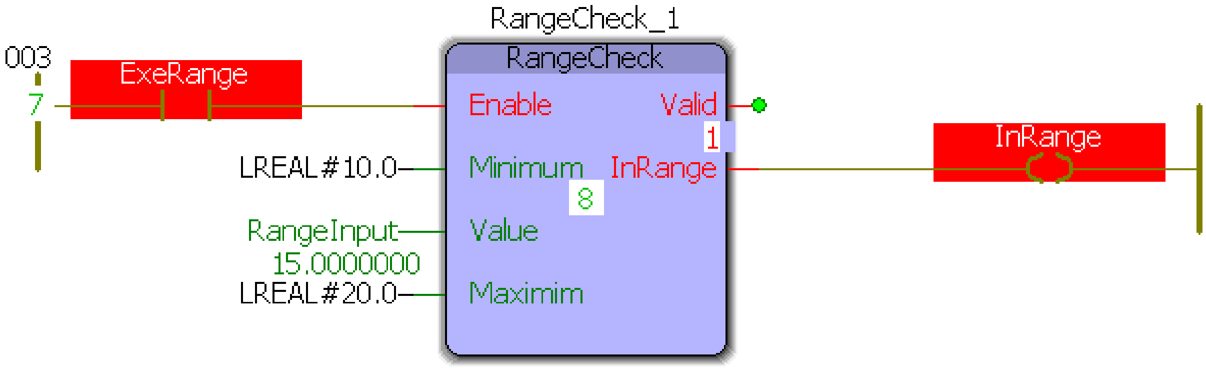
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Minimum	LREAL	The smallest 'Value' that will set the InRange output high.	LREAL#0.0
V	Value	LREAL	The data to be tested 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)	

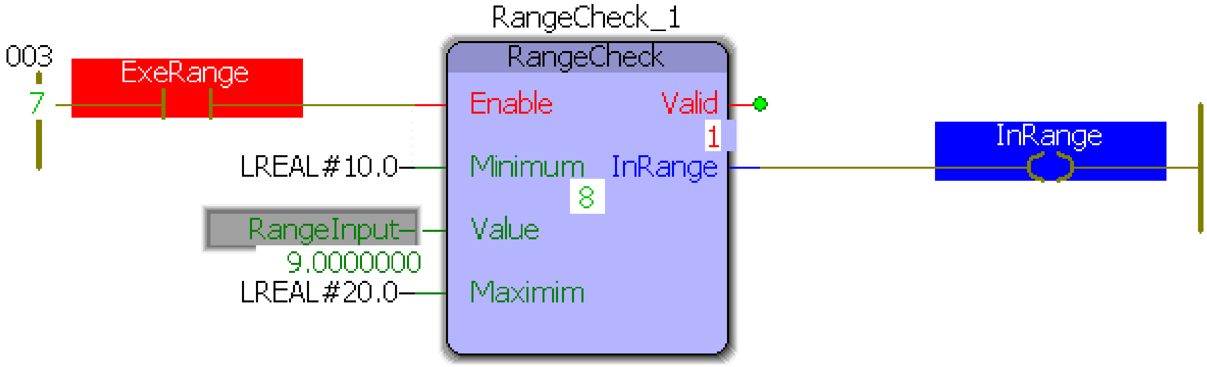
Error Description

No errors will be generated.

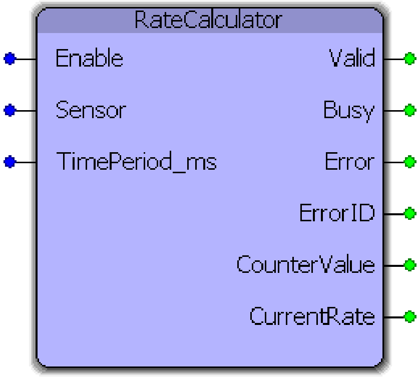
Example

ExeRange does not need to be toggled if Value is changed, as demonstrated below:


RangeCheck_1
RangeCheck
003
7
ExeRange
Enable
Valid
1
LREAL#10.0
Minimum
InRange
8
RangeInput
Value
15.0000000
LREAL#20.0
Maximim
InRange


RangeCheck_1
RangeCheck
003
7
ExeRange
Enable
Valid
1
LREAL#10.0
Minimum
InRange
8
RangeInput
Value
9.0000000
LREAL#20.0
Maximim
InRange

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.	FALSE
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.	
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.	

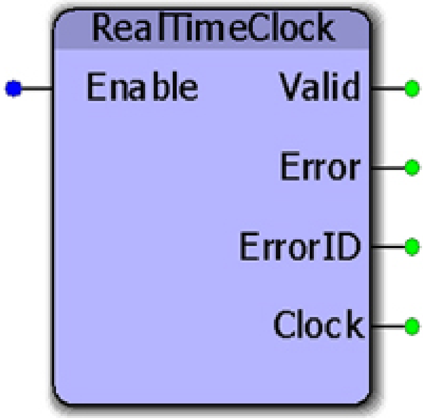
Notes

- 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

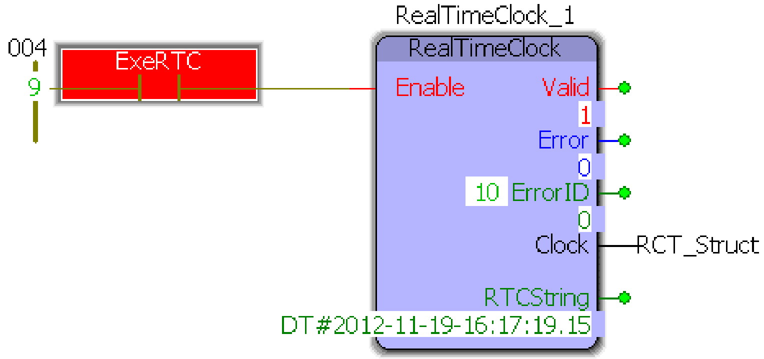
The controllers clock can be set from the web server, or by using the Y_SetRTC function block from the YMotion firmware library, which requires firmware version 2.0.0 or greater.

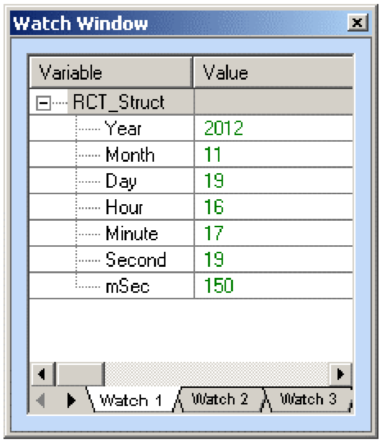
Error Description

No errors will be generated.

Example

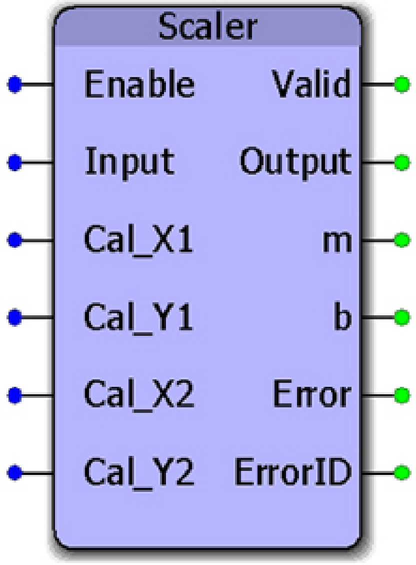
The output of this block is continually updated as long as Enable is TRUE.





Variable	Value
RCT_Struct	
Year	2012
Month	11
Day	19
Hour	16
Minute	17
Second	19
mSec	150

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				
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.

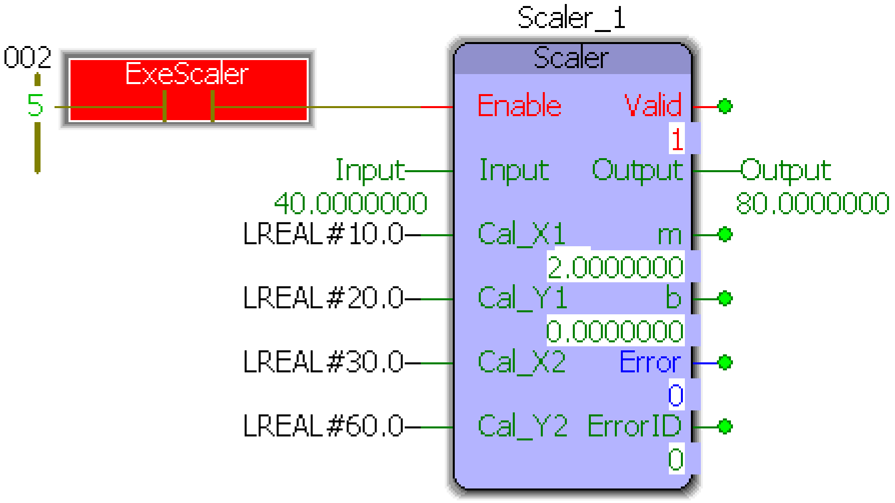
Notes

- 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.

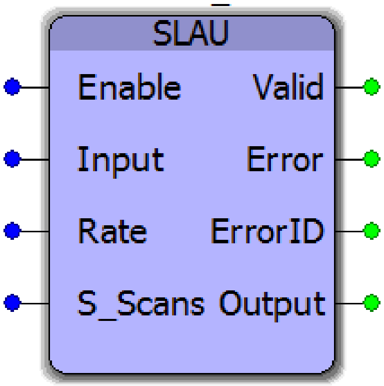
Error Description

ErrorID	Meaning
0	No error.
10075	Calibration Error: Cal_X2 must be greater than Cal_X1.

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

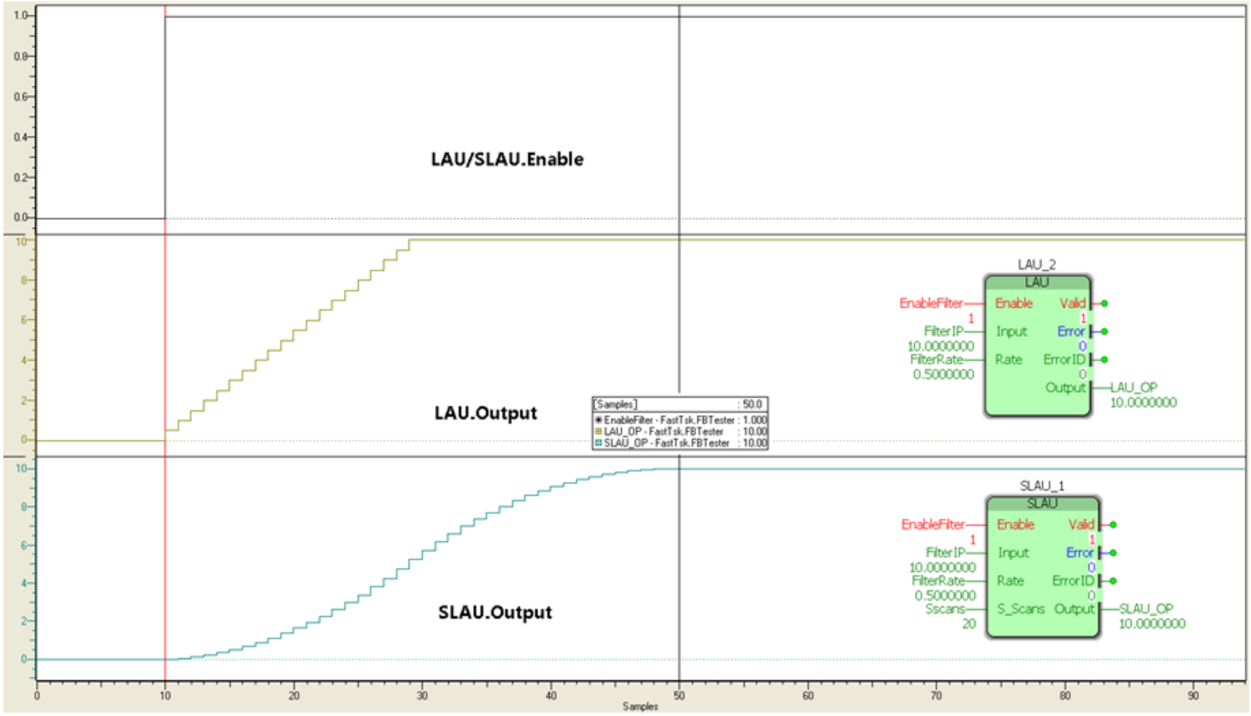
*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	Input	LREAL	Target Value.	LREAL#0.0
V	Rate	LREAL	Acceleration/Deceleration per scan. The time required for the Output to become the Input value profile depends on the Rate 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	Output value.	

Error Description

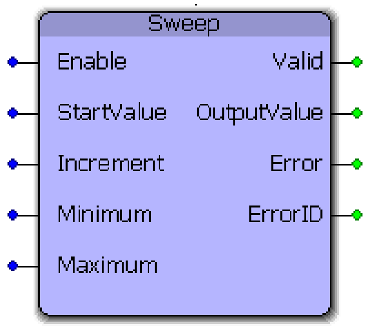
ErrorID	Meaning
10093	Rate cannot be less than or equal to 0.
10094	S_Scans cannot be less than 2 or greater than 30000.

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.



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

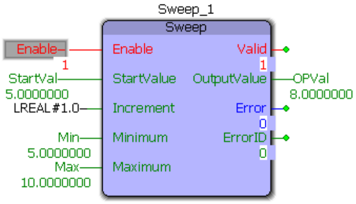
Parameters

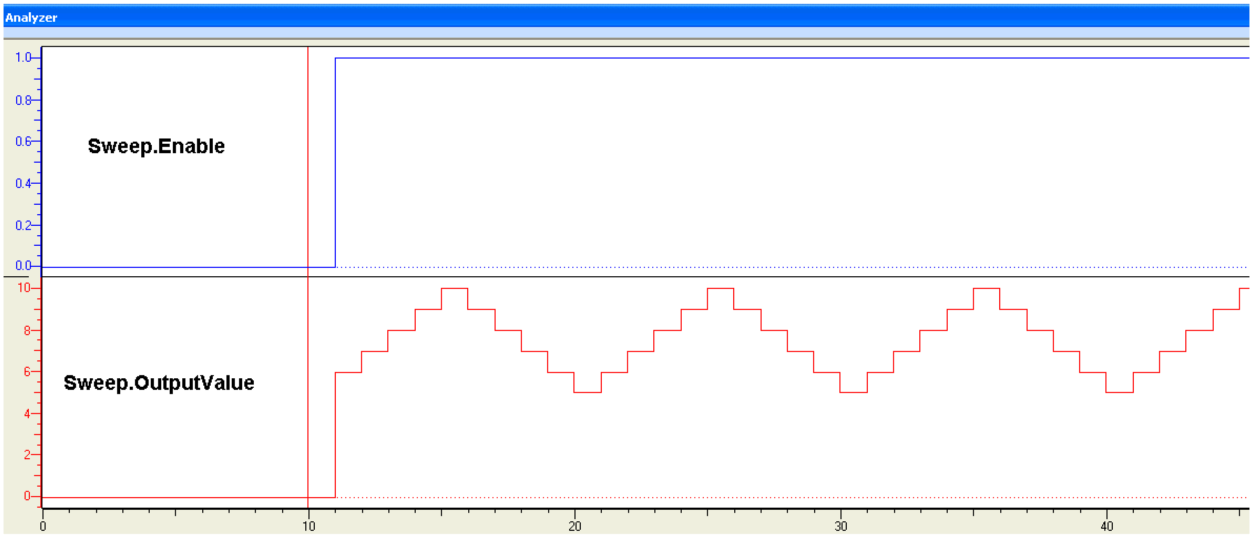
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
B	StartValue	LREAL	The OutputValue will start from this value	LREAL#0.0
B	Increment	LREAL	The amount by which the Outputvalue is changed each scan	LREAL#0.0
B	Minimum	LREAL	The minimum value output	LREAL#0.0
B	Maximum	LREAL	The maximum value output	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates if the function is operating normally	
B	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.	
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

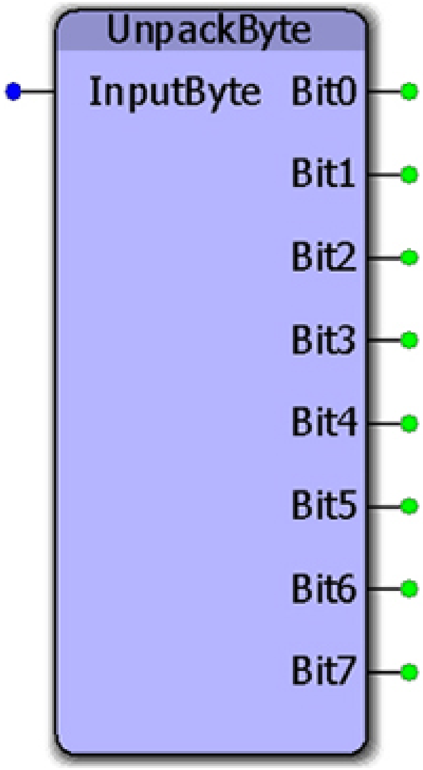
No errors will be generated.

Example:


Sweep_1
Sweep
Enable
1
StartVal
5.0000000
LREAL#1.0
Min
5.0000000
Max
10.0000000
Enable
StartValue
Increment
Minimum
Maximum
Valid
1
OutputValue
OPVal
8.0000000
Error
0
ErrorID
0


Analyzer
1.0
0.8
0.6
0.4
0.2
0.0
Sweep.Enable
10
8
6
4
2
0
Sweep.OutputValue
0
10
20
30
40

UnpackByte



This function block converts a byte into discrete bits.

Library

Yaskawa Toolbox

Parameters

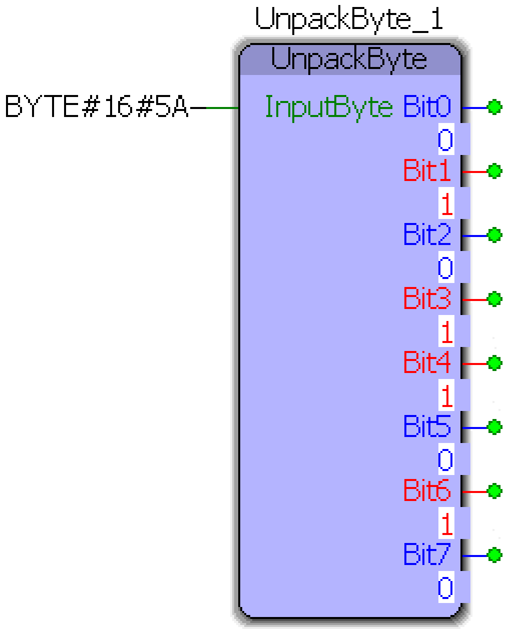
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	InputByte	BYTE	The input data to be separated into bits.	BYTE#0
VAR_OUTPUT				
V	Bit0	BOOL	Bit 0 of the InputByte	
V	Bit1	BOOL	Bit 1 of the InputByte	
V	Bit2	BOOL	Bit 2 of the InputByte	
V	Bit3	BOOL	Bit 3 of the InputByte	
V	Bit4	BOOL	Bit 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

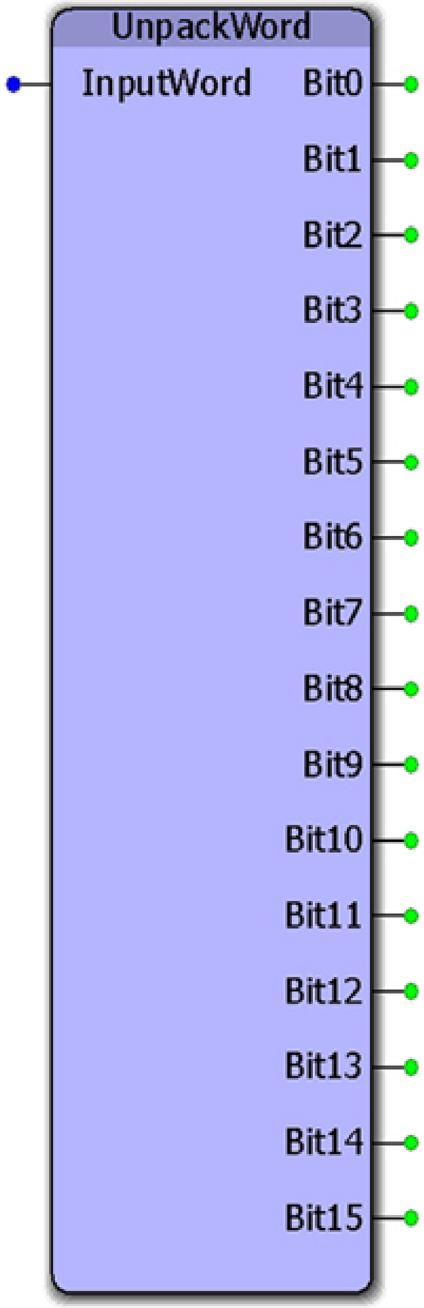
Error Description

No errors will be generated.

Example



UnpackWord



This function block separates a word into individual bits.

Library

Yaskawa Toolbox

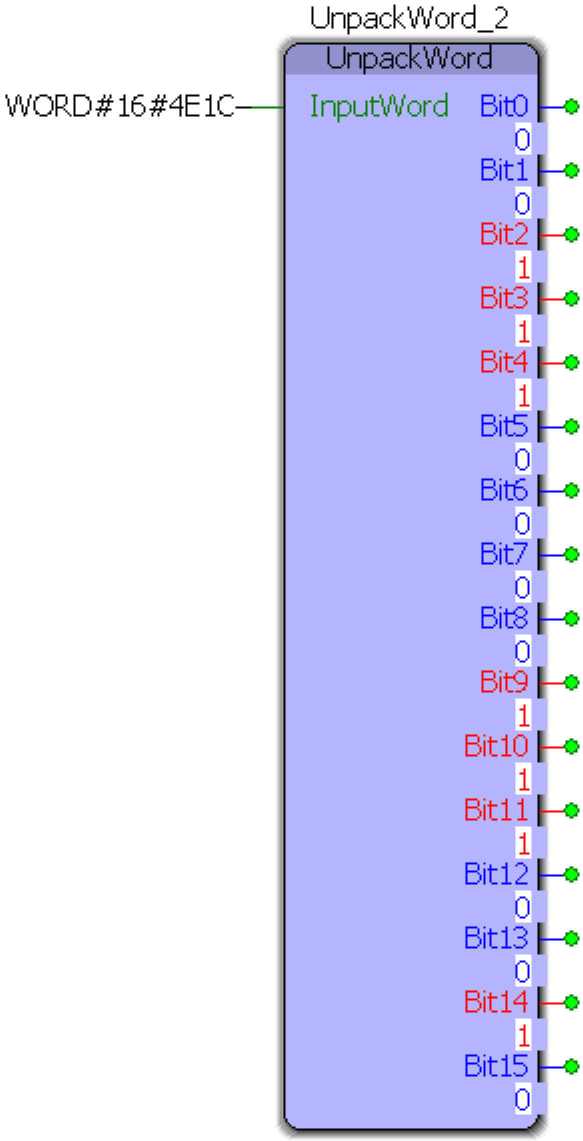
Parameters

*	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	

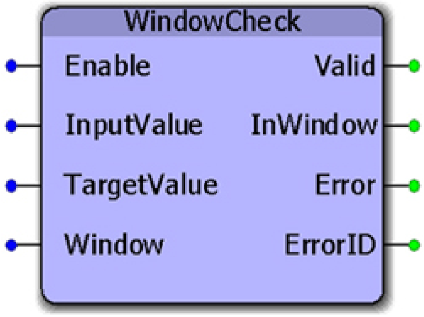
Error Description

No errors will be generated.

Example


UnpackWord_2
UnpackWord
WORD#16#4E1C
InputWord
Bit0
0
Bit1
0
Bit2
1
Bit3
1
Bit4
1
Bit5
0
Bit6
0
Bit7
0
Bit8
0
Bit9
1
Bit10
1
Bit11
1
Bit12
0
Bit13
0
Bit14
1
Bit15
0

WindowCheck



This function block sets the InWindiow 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

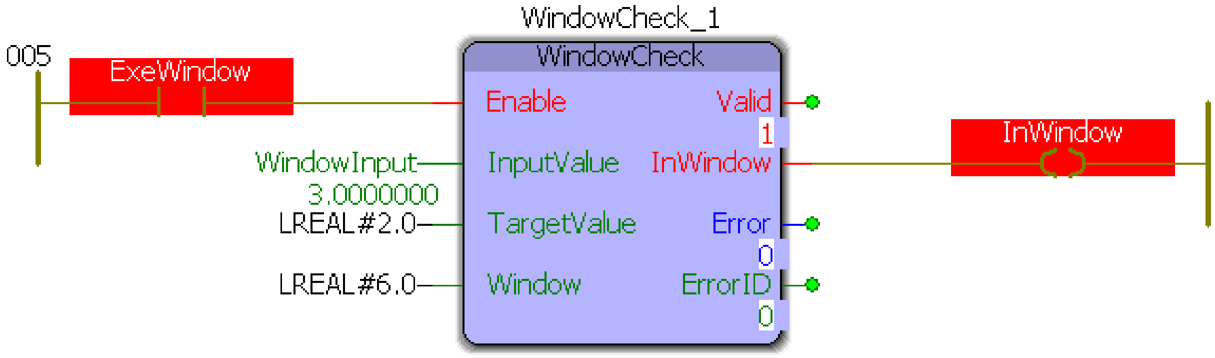
Parameters

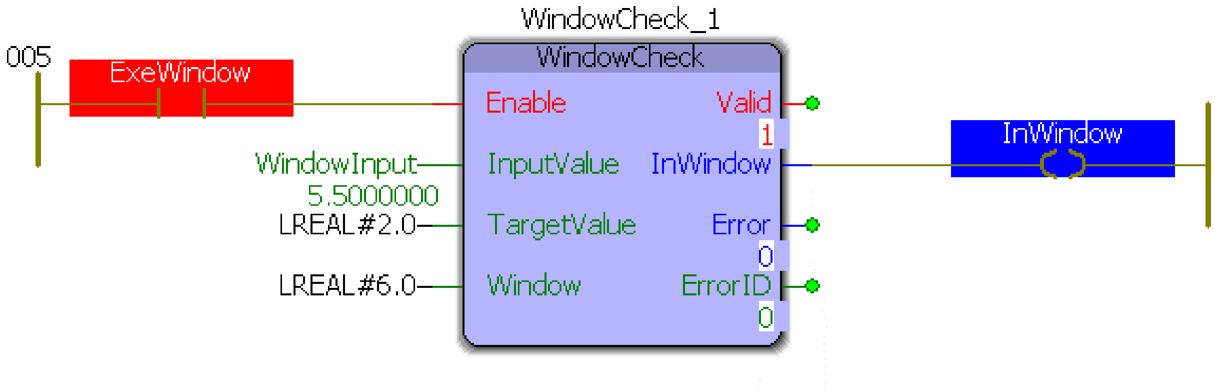
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute every scan while Enable is held high and there are no errors.	FALSE
V	InputValue	LREAL	The data to be tested against the TargetValue	LREAL#0.0
V	TargetValue	LREAL	The desired data to be compared 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

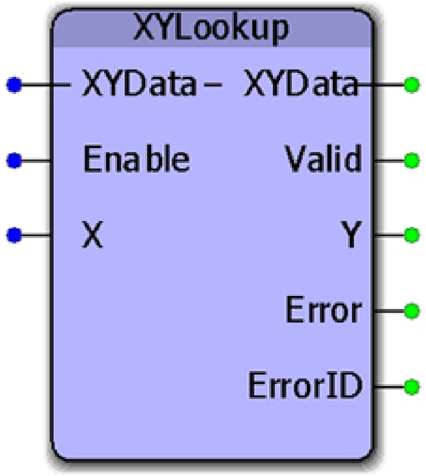
ErrorID	Meaning
0	No error.
10076	WindowSize must be greater than zero.

Example


WindowCheck_1
005
ExeWindow
WindowCheck
Enable
Valid
1
WindowInput
3.0000000
InputValue
InWindow
InWindow
LREAL#2.0
TargetValue
Error
0
LREAL#6.0
Window
ErrorID
0


WindowCheck_1
005
ExeWindow
WindowCheck
Enable
Valid
1
WindowInput
5.5000000
InputValue
InWindow
InWindow
LREAL#2.0
TargetValue
Error
0
LREAL#6.0
Window
ErrorID
0

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

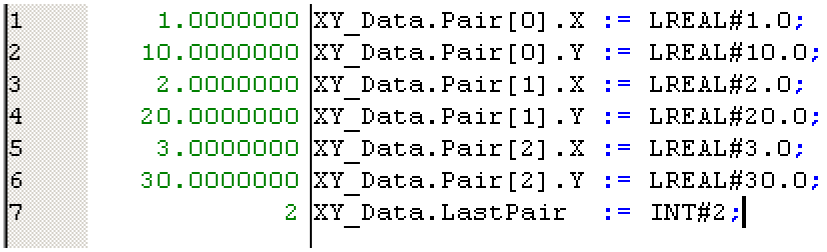
- Works for sets where the X value is always increasing or always decreasing, but to use decreasing values of X requires v205 or higher.

Error Description

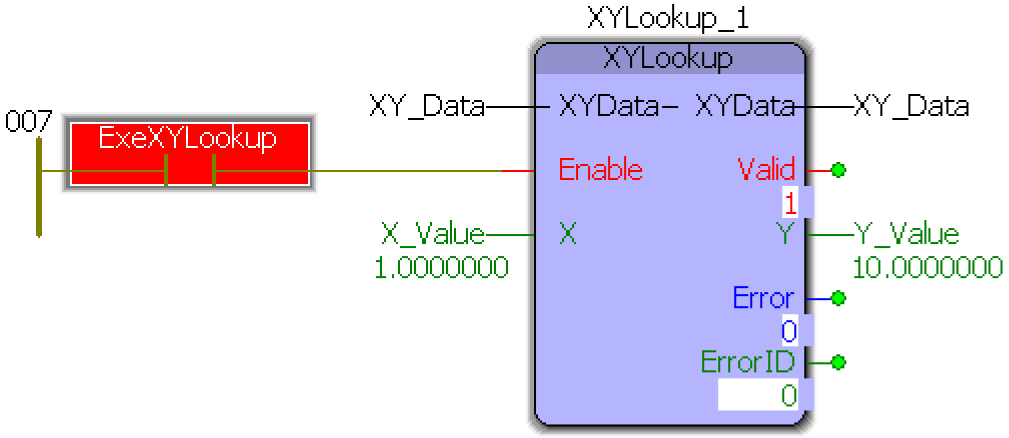
ErrorID	Meaning
0	No error.
10038	CamData.LastSegment must be greater than 0 and less than 400, or whatever value has been declared as the ARRAY size in the CTB_Types file.

Example

The XY_Data structure was initialized as:



```
 1.0000000 XY_Data.Pair[0].X := LREAL#1.0;
10.0000000 XY_Data.Pair[0].Y := LREAL#10.0;
 2.0000000 XY_Data.Pair[1].X := LREAL#2.0;
20.0000000 XY_Data.Pair[1].Y := LREAL#20.0;
 3.0000000 XY_Data.Pair[2].X := LREAL#3.0;
30.0000000 XY_Data.Pair[2].Y := LREAL#30.0;
         2 XY_Data.LastPair  := INT#2;
```



Yaskawa DataTypes

Data Type: MovingAverageArray

Data Type: PIDStruct

Data Type: RTCStruct

Data Type: XYData

Data Type: XYDataStruct

Toolbox Help Documentation

Help version created 1/25/2017

Enumerated Types for Yaskawa 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
Analog_Inputs	**<ENUM Type Description>**		
	0	<Enumeration_Text_Value>	<Enumeration_functionality_description>
	1		
	2		
FBErrorDetails	**<ENUM Type Description>**		
	0		
	1		
	2		
FBErrorHistory	**<ENUM Type Description>**		
	0		
	1		
	2		
FBErrorStruct	**<ENUM Type Description>**		
	0		
	1		
	2		

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

ExplicitData

For use with the Explicit_Message function block.

Data Type Declaration

```
TYPE
ExplicitData : ARRAY[0..503] OF BYTE;
END_TYPE
```

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]

Service

For use with the Explicit_Message function block.

Data Type Declaration

```
TYPE
Service : ARRAY[0..7] OF BYTE;
END_TYPE
```

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]

SenderContext

For use with the Explicit_Message function block.

Data Type Declaration

```
TYPE
SenderContext : 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		MyUnRegSes-

				sionRequestStruct.USR_Options1
U	USR_Options2	BYTE		MyUnRegSessionRequestStruct.USR_Options2
U	USR_Options3	BYTE		MyUnRegSessionRequestStruct.USR_Options3
U	USR_Options4	BYTE		MyUnRegSessionRequestStruct.USR_Options4

Function Block ErrorID List

ErrorID	Description
0	No error.
1	Time limit exceeded.
2	Distance limit exceeded.
3	Torque limit exceeded.
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. 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 can not 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 aborted due to axis alarm. It is also possible that a software limit has been exceeded.
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.
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.
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.
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.
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.
4415	Reboot is already in progress.
4416	Add IP Address already in progress
4417	Remove IP Address already in progress
4418	[[[Undefined variable Primary.ErrorID_44184_Description]]]
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).
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 POUs.
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.
4647	The synch mode does not correspond to a valid enumeration value.
4648	The parameter number does not exist for the specified axis.
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.
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.
4659	Acceleration is less than or equal to zero.
4660	Deceleration is less than or equal to zero.
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.
4666	Denominator is zero.
4667	Jerk is less than or equal to zero.
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 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.
Invalid Input Data	
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.
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 can not 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	Cound 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.
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.
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.
Group ErrorIDs	
8960	Invalid axes group. Confirm that the AxesGroup variable has the correct %M address as automatically assigned by the Hardware Configuration.
8961	An axis is already owned by another group.
8962	Group activation is blocked. Ownership can not be changed while Mechatrolink reset is in progress.
8963	Invalid coordinate system.
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.
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 circular path direction.
8971	Invalid circle geometry.
8972	A grouped axis is disabled.
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.
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.
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 .
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.
MLX200 ErrorIDs	
9216	Invalid Host_ID. Supported Host_IDs are (0 = MECHATROLINK group, 1 = MLX hosted 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.
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. A command for an invalid state transition was issued.
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 RecordedPosition is already past the TrackOptions.MaxPosition
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. Possibly the EIP_Output structure is not mapped to the correct hardware address, or there is a timing problem between the MPiec and the MLX. If you are sure the EIP_Output variable is mapped correctly, try slowing the cyclic task interval 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
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	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 compaitible 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. A confirmational check of the expected TCP position was out of range of 0.1 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 liveman switch is preventing manual mode operation.
9253	The Group's Safety circuit is preventing motion.
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	Gripper Open Error (Timeout).
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..
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.
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 path has an invalid Segment Type. Valid Segments Types are defined in Group Toolbox GroupTypes file as enumeration GTB_SegmentType.
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.
10064	Track Number outside the 0-31 range.
10065	FirstOnPosition is not equal to 0.
10066	LastOnPosition is not equal to 0.
10067	AxisDirection is not equal to 0.
10068	CamSwitchMode is not equal to 0.
10069	Duration is set to 0 or a negative value.
10070	OnCompensationScaler is set to an invalid value.
10071	OffCompensationScaler is set to an invalid value.
10072	ImproperOnPos_SetError.
10073	OnOffPosition_Error.

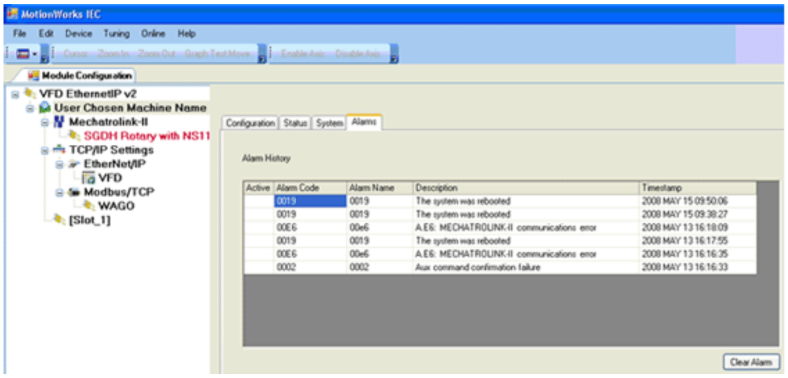
10074	Direction must be 0 for positive, or 2 for negative.
10075	Calibration Error: Cal_X2 must be greater than Cal_X1.
10076	WindowSize must be greater than zero.
10077	Cubic Spline maximum number of consecutive segments exceeded. DataType definition for the Matrix could be increased if necessary.
10078	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.
10086	MaxPosCorrection must be zero or positive, MaxNegCorrection must be or zero or negative.
10089	Bezier Error. There should be a straight line segment before and after the bezier segment.
10093	Rate is less than or equal to 0.
10094	S_Scans is less than 2 or greater than 30000.
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.
10100	Both axes must be configured for the same axis type (Rotary / Linear) and if Rotary, they must have the same Machine Cycle.
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. 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.
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.
10140	Must be greater than zero and less than 20.
10150	Theta1 Below Minimum.
10151	Theta1 Above Maximum.
10152	Theta2 Below Minimum.
10153	Theta2 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.
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 assiged when the motion function block was executed.
10608	CompTypeError - 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 GCode ypes 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 struture.
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
10619	Invalid file name. File names must only contain alphanumeric characters. The first character must not be numeric.
11050	Cam correction (shift/offset) has been aborted by another function block.
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.
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 '.'.
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.
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 servo drive failed.
45335	Failed to initialize 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	
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.

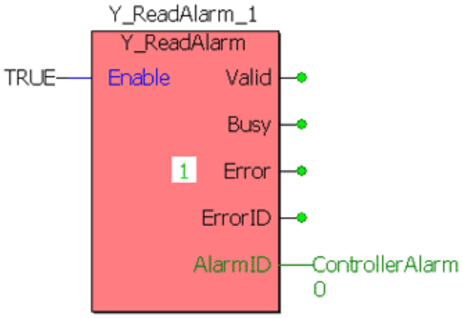
Controller AlarmID List

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.



Alarm History

Active	Alarm Code	Alarm Name	Description	Timestamp
	0019	0019	The system was rebooted	2008 MAY 15 09:50:06
	0019	0019	The system was rebooted	2008 MAY 15 09:38:27
	00E6	00e6	A.E6: MECHATROLINK-II communications error	2008 MAY 13 16:18:09
	0019	0019	The system was rebooted	2008 MAY 13 16:17:55
	00E6	00e6	A.E6: MECHATROLINK-II communications error	2008 MAY 13 16:16:35
	0002	0002	Aux command confirmation failure	2008 MAY 13 16:16:33



	Hex Code		Description
	ErrorClass (UINT)	AxisErrorID (UINT) GroupErrorID (UINT)	ErrorClass+AxisErrorID output from MC_ReadAxisError
	AlarmID (UDINT)		AlarmID output from Y_ReadAlarm
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.
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.
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.
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. 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.
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.
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.
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.
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	0001	reserved
Mechatrolink	3312	0002	reserved
Mechatrolink	3312	0003	reserved
Mechatrolink	3312	0004	reserved
Mechatrolink	3312	0005	reserved
Mechatrolink	3312	0006	reserved
Mechatrolink	3312	0007	reserved
Mechatrolink	3312	0008	reserved
Mechatrolink	3312	0009	reserved
Mechatrolink	3312	000A	reserved
Mechatrolink	3312	000B	reserved
Mechatrolink	3312	000C	reserved
Mechatrolink	3312	000D	reserved
Mechatrolink	3312	000E	reserved
Mechatrolink	3312	000F	reserved
Mechatrolink	3312	0010	reserved
Mechatrolink	3312	0011	reserved
Mechatrolink	3312	0012	reserved
Mechatrolink	3312	0013	reserved
Mechatrolink	3312	0014	reserved
Mechatrolink	3312	0015	reserved
Mechatrolink	3312	0016	reserved
Mechatrolink	3312	0018	reserved
Mechatrolink	3312	0019	reserved
Mechatrolink	3312	001A	reserved
Mechatrolink	3312	001B	reserved
Mechatrolink	3312	001C	reserved
Mechatrolink	3312	001D	reserved
Mechatrolink	3312	001E	reserved
Mechatrolink	3312	001F	reserved
Mechatrolink	3312	0020	reserved
Mechatrolink	3312	0021	reserved
Mechatrolink	3312	0025	reserved
Mechatrolink	3312	0026	reserved
Mechatrolink	3312	0027	reserved
Mechatrolink	3312	0028	reserved
Mechatrolink	3312	0029	reserved
Mechatrolink	3312	002A	reserved
Mechatrolink	3312	002B	reserved
Mechatrolink	3312	002C	reserved
Mechatrolink	3312	002D	reserved

Mechatrolink	3312	002E	reserved
Mechatrolink	3312	002F	reserved
Mechatrolink	3312	0031	reserved
Mechatrolink	3312	0083	reserved
Mechatrolink	3312	0084	reserved
Mechatrolink	3312	0085	reserved
Mechatrolink	3312	0086	reserved
Mechatrolink	3312	0087	reserved
Mechatrolink	3312	0088	reserved
Mechatrolink	3312	0089	reserved
Mechatrolink	3312	008A	reserved
Mechatrolink	3312	008B	reserved
Mechatrolink	3312	0091	reserved
Mechatrolink	3312	0092	reserved
Mechatrolink	3312	0093	reserved
Mechatrolink	3312	0094	reserved
Mechatrolink	3312	00E6	reserved
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.
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.
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.
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.
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
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 lees 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.
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.

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_VerifyParmeters 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.
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
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 iputs 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.
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.
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.
Memory			
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.
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.
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).
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 com-

			munications.
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
Mechatrolink	4312	0000	The base code for inverter warnings. The inverters warning value is bitwise OR'd in with this base value.
Mechatrolink	4312	0001	Reserved
Mechatrolink	4312	0002	Reserved
Mechatrolink	4312	0003	Reserved
Mechatrolink	4312	0004	Reserved
Mechatrolink	4312	0005	Reserved
Mechatrolink	4312	0006	Reserved
Mechatrolink	4312	0007	Reserved
Mechatrolink	4312	0008	Reserved
Mechatrolink	4312	0009	Reserved
Mechatrolink	4312	000A	Reserved
Mechatrolink	4312	000B	Reserved
Mechatrolink	4312	000C	Reserved
Mechatrolink	4312	000D	Reserved
Mechatrolink	4312	000E	Reserved
Mechatrolink	4312	0010	Reserved
Mechatrolink	4312	0011	Reserved
Mechatrolink	4312	0012	Reserved
Mechatrolink	4312	0013	Reserved
Mechatrolink	4312	0014	Reserved
Mechatrolink	4312	0017	Reserved
Mechatrolink	4312	0018	Reserved
Mechatrolink	4312	001A	Reserved
Mechatrolink	4312	001B	Reserved
Mechatrolink	4312	001C	Reserved
Mechatrolink	4312	001D	Reserved
Mechatrolink	4312	001E	Reserved
Mechatrolink	4312	001F	Reserved
Mechatrolink	4312	0022	Reserved
Mechatrolink	4312	0023	Reserved
Mechatrolink	4312	0024	Reserved
Mechatrolink	4312	0025	Reserved

Mechatrolink	4312	0026	Reserved
Mechatrolink	4312	0094	Reserved
Mechatrolink	4312	0095	Reserved
Mechatrolink	4312	0096	Reserved
Mechatrolink	4312	00E5	Reserved
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.
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 fimware 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.