MotionWorks IEC Toolboxes Rev: 2013-09-013

Table Of Contents

Yaskawa's IEC61131-3 Toolboxes: 2013-09-13

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.
- **Kinematics** toolbox contains forward and inverse kinematics for selected mechanisms.
- **Math** Toolbox provides compatibility with the built in function that include EN and ENO outputs, and also provides other tools such as ATAN2, and Floating Point Remainder (REM).
- **PackML** is both a Template and Toolbox for designing applications to take advantage of the PackML specification. It emphasizes machine state and transition logic and provides predefined PackML data structures.
- **PLCopen** toolbox contains functions that build upon the PLCopen standard functions. It can serve as a starting point for every project.

- **Yaskawa** toolbox contains functions that add basic functionality, such as PID Control, or a Moving Average Filter.

Others coming soon!

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 MotionWorksIEC and many of our toolboxes.

Cam Toolbox

Cam Toolbox

Cam Toolbox contains functions which provide enhanced support of the PLCopen function blocks:

Y_CamStructSelect

Y_CamIn

Y_CamOut

Y_CamShift

Y_SlaveOffset

Y_CamScale

Y_ReleaseCamTable

See the Cam_Toolbox eLearning Modules on Yaskawa's YouTube Channel for video tutorials and examples.

Requirements:

PLCopen Toolbox, Math Toolbox. Some functions such as CamSlaveFeedToLength require the YMotion firmware library.

The Cam Toolbox consists of the following:

Data Types:

Data Type	Description
AXIS_REF	Identifies an axis
AxisParameterStruct	For use with the CamSlave_FeedToLength and CamSlave_WindowCheck function blocks.
BlendStruct	For use with the CamBlend function block
CamPairs	Used by the CamGenerator function block
CamParameters	Supporting structure for CamSegmentStruct. For use with the CamGenerator function block
CamSegmentArray	Supporting structure for CamSegmentStruct. For use with the CamGenerator function block
CamSegmentStruct	For use with the CamGenerator function block

CamStruct	For use with Y_CamIn and Y_CamOut function blocks
CamSyncStruct	For use with the CamControl and CamShift_Control function blocks
Matrix	For use by the CamGenerator for Cubic Spline calculations
SubMatrix	For use by the CamGenerator for Cubic Spline calculations
TablesIDStruct	For use with the CamTableUpdate function block
UINTArray	For use with the CamTableManager Function Block
Y_MS_CAM_STRUCT	For use with the CamGenerator, CamTableUpdate, CamMaster_Lookup, and SlaveIndex_Lookup function blocks.

Enumerated Types:

Enumerated Type	Description
TB_CurveType	For use with the CamSegmentStruct when using the CamGenerator function block
TB_Mode	ENUM Type for CamShift_Control to select rotary or linear slave motion

Function Blocks:

Function Block	Description
CamAnalyzer	Designed to calculate peak velocity, accel/decel and jerk for a given cam profile and master velocity
CamBlend	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
CamControl	Used to make a decision on when to engage and disengage a cam for applications where product length or frequency can be variable
CamGenerator	Designed to replicate the functionality of Yaskawa's CamTool software
CamMaster_Lookup	Provides the master position given a slave position by searching the referenced CamTable
CamShift_Control	Calculates shift parameters and performs shifting on the master position
CamSlave_FeedToLength	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
CamSlave_FeedToLength2	CamSlave_FeedToLength2 is an enhancement of CamSlave_FeedtoLength and uses Y_ProbeContinuous making use of the continuous latch feature of the Sigma-5
CamSlave_Lookup	Returns the array index value corresponding to the given slave position
CamSlave_PullToLength	Designed for applications where the slave mechanism pulls material forward but the mechanism has a reciprocating stroke
CamSlave_Recover	Used to bring the slave axis back in sync with the master axis after

	camming was interrupted unexpectedly
CamSlave_WindowCheck	Used inside CamSlave_FeedToLength to determine when the MC_TouchProbe output is valid and should be used for correction
CamTableManager	Serves as a FIFO buffer for CamTableID's
CamTableUpdate	Aids with cam file management when on the fly changes to the table data are required
SlaveIndex_Lookup	Returns the array index value corresponding to the given slave position
SlaveRegistrationCheck	Uses 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

Getting Started: Cam

Requirements for v204

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

Firmware libraries:

- YMotion (only if using CamSlave_FeedToLength2)

User libraries:

- Math_Toolbox (v202 or higher)
- DataTypes_Toolbox (v200 or higher)
- PLCopen_Toolbox (v205 or higher)

Using the Cam Toolbox

See Yaskawa's Youtube video - Camming Demonstration with Yaskawa MP2300Siec for more info.

Cam Revision History

Current Version:

New for Cam 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

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

Previous Versions:

(************* 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) Improved CamGenerator to support wrap around cubic spline segments at the beginning and the end of the cam.

(YEU) 7 spline categories tested.

2) 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

Creating Cam Tables

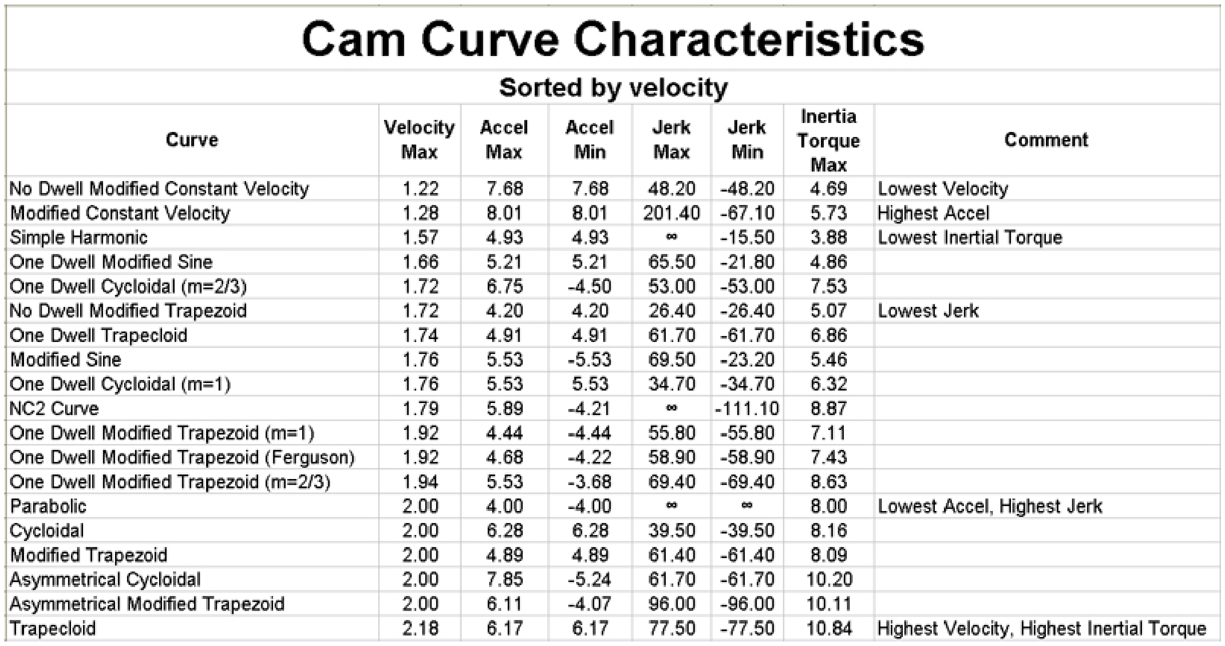
Cam Curve Characteristics

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:

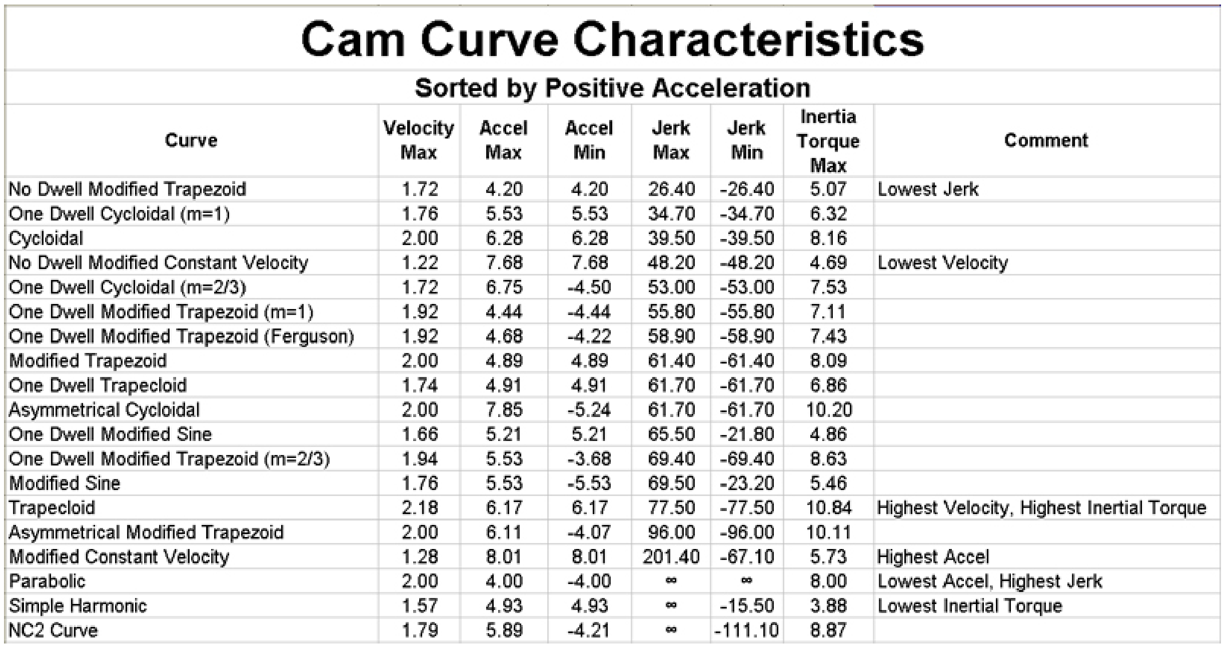
1. Velocity V and Acceleration A are to be continuous
2. Low values of Vm and Qm are needed in low speed and heavy load applications.
3. Low values of Am and Jm are needed in high speed and light load applications.
4. Asymmetrical curve having the longer period of deceleration than acceleration should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

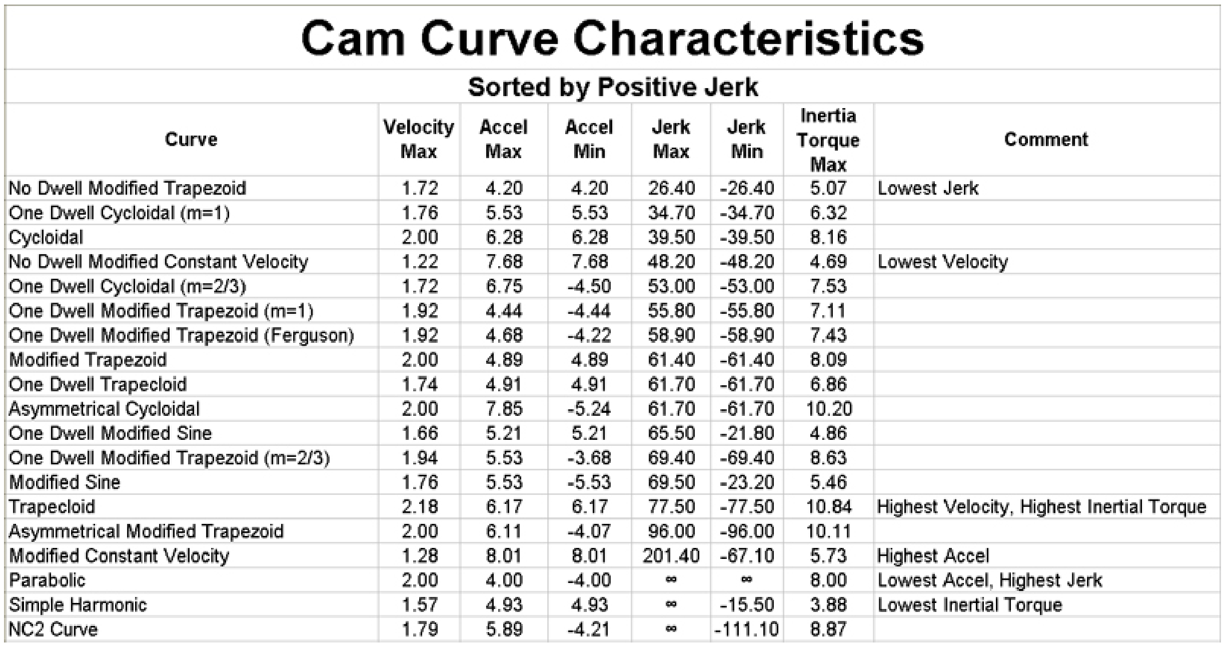
Curve	Velocity Max	Accel Max	Accel Min	Jerk Max	Jerk Min	Inertia Torque Max	Comment
No Dwell Modified Constant Velocity	1.22	7.68	7.68	48.20	-48.20	4.69	Lowest Velocity
Modified Constant Velocity	1.28	8.01	8.01	201.40	-67.10	5.73	Highest Accel
Simple Harmonic	1.57	4.93	4.93	∞	-15.50	3.88	Lowest Inertial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

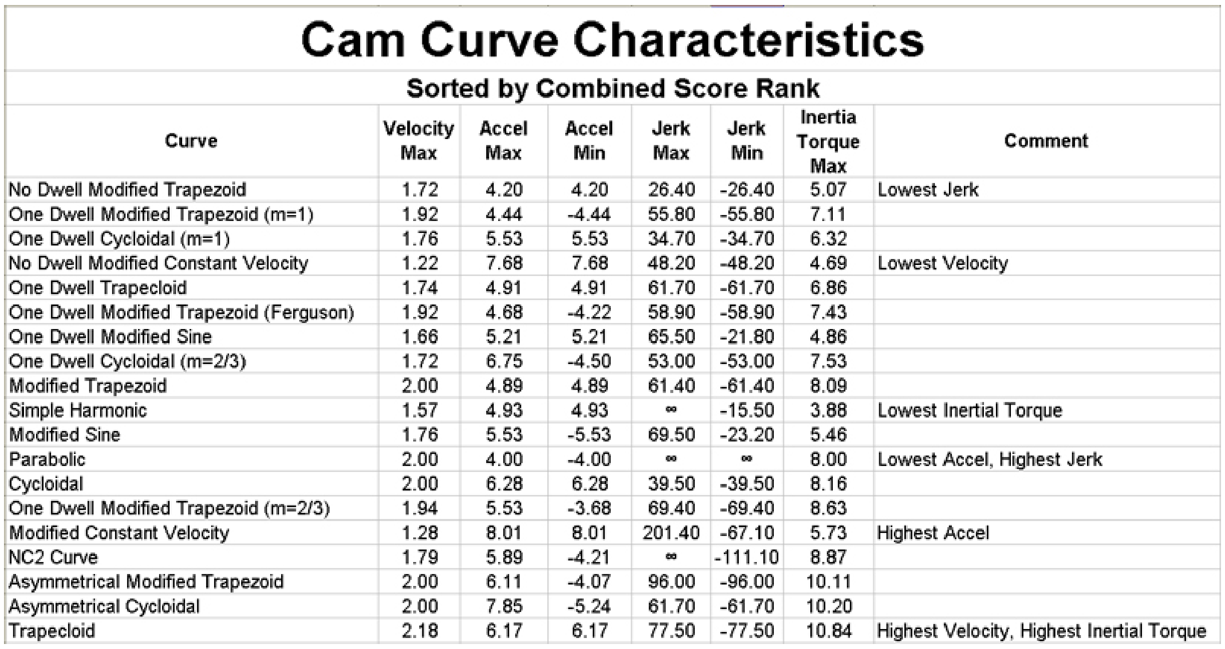
Curve	Velocity Max	Accel Max	Accel Min	Jerk Max	Jerk Min	Inertia Torque Max	Comment
No Dwell Modified Trapezoid	1.72	4.20	4.20	26.40	-26.40	5.07	Lowest Jerk
One Dwell Cycloidal (m=1)	1.76	5.53	5.53	34.70	-34.70	6.32	
Cycloidal	2.00	6.28	6.28	39.50	-39.50	8.16	
No Dwell Modified Constant Velocity	1.22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

Cam Curve Types

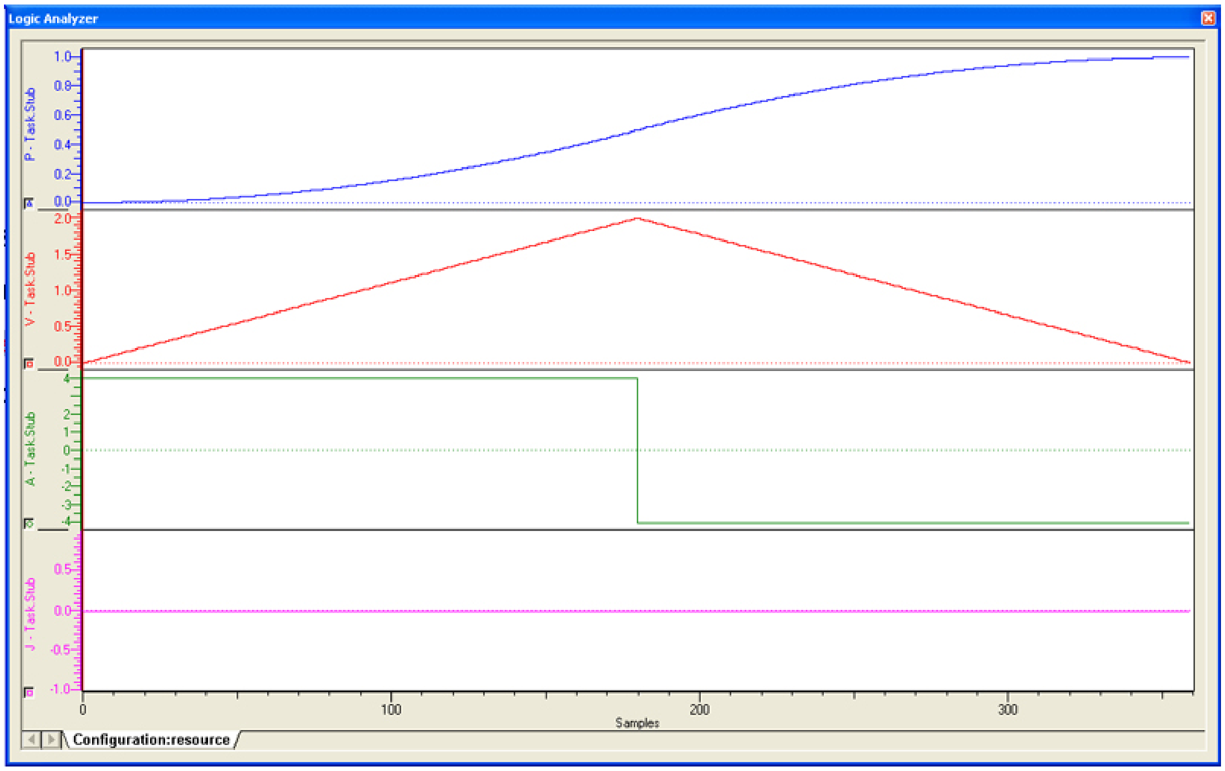
Cam Curve Types

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

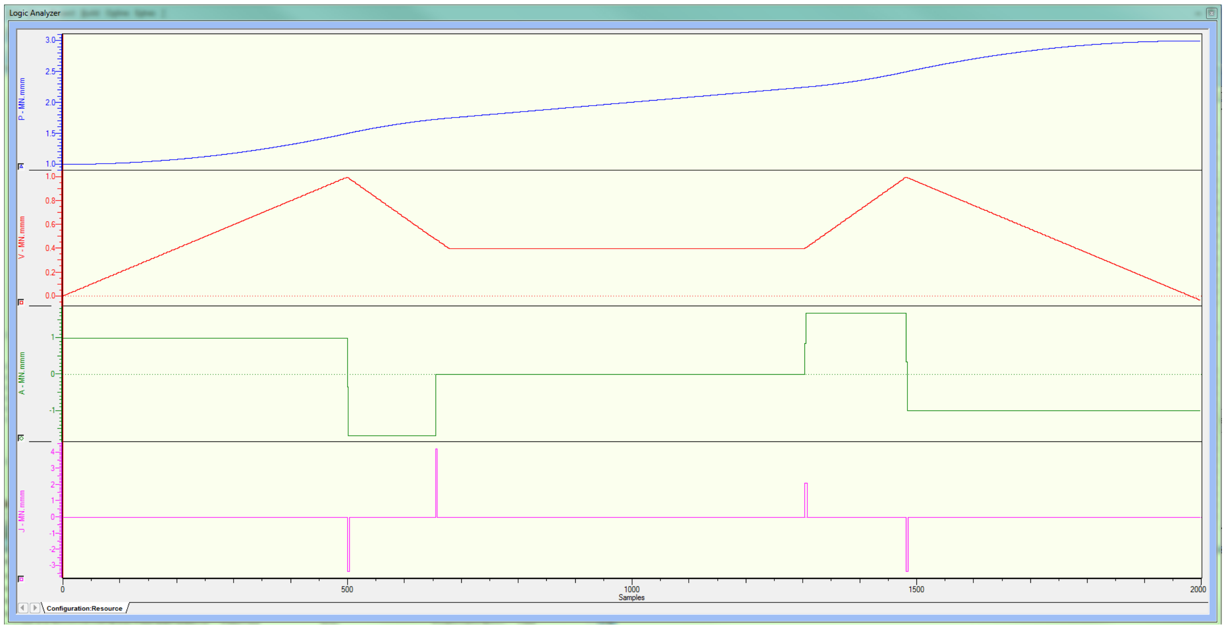
- ParabolicVelocityBlend

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.

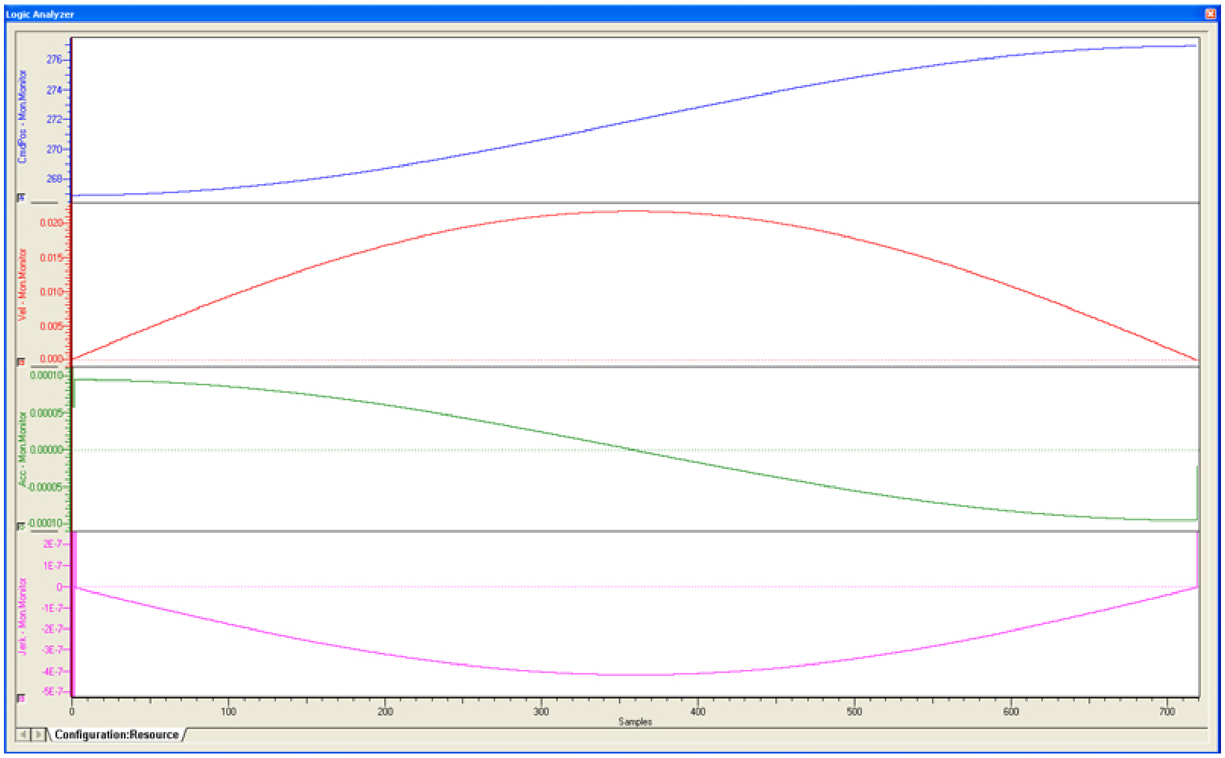


ParabolicVelocityBlend

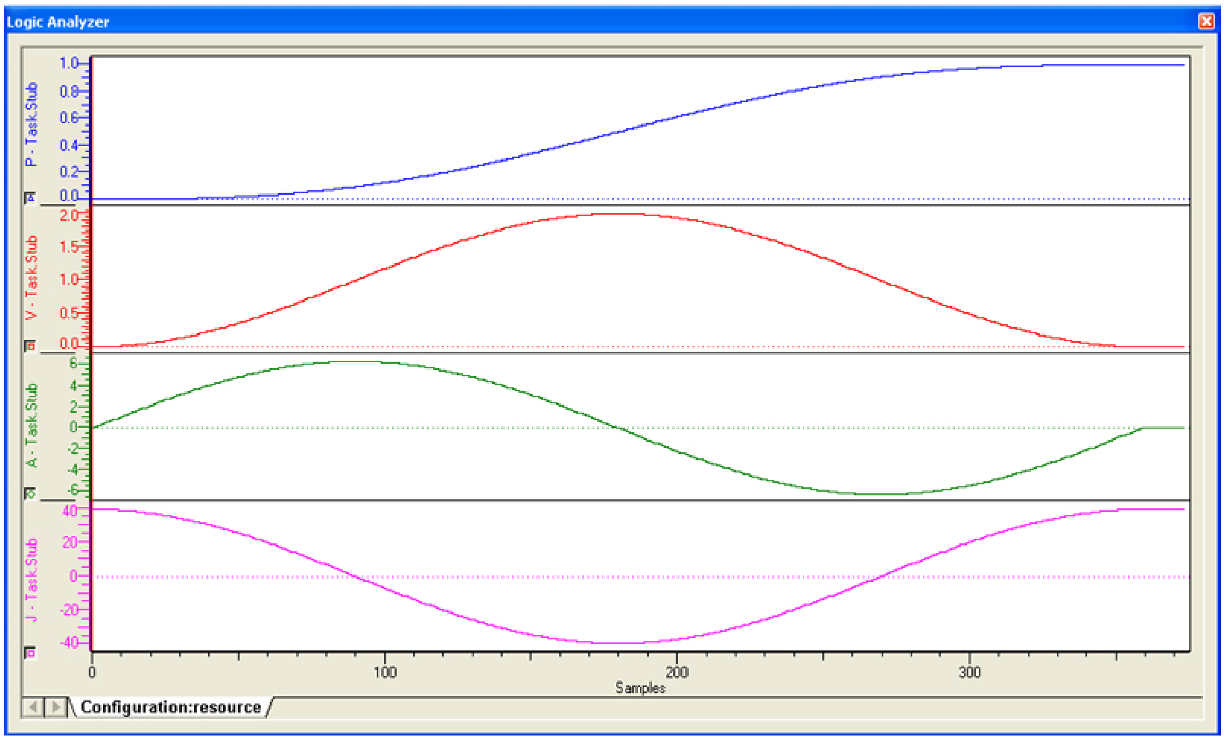


Simple Harmonic

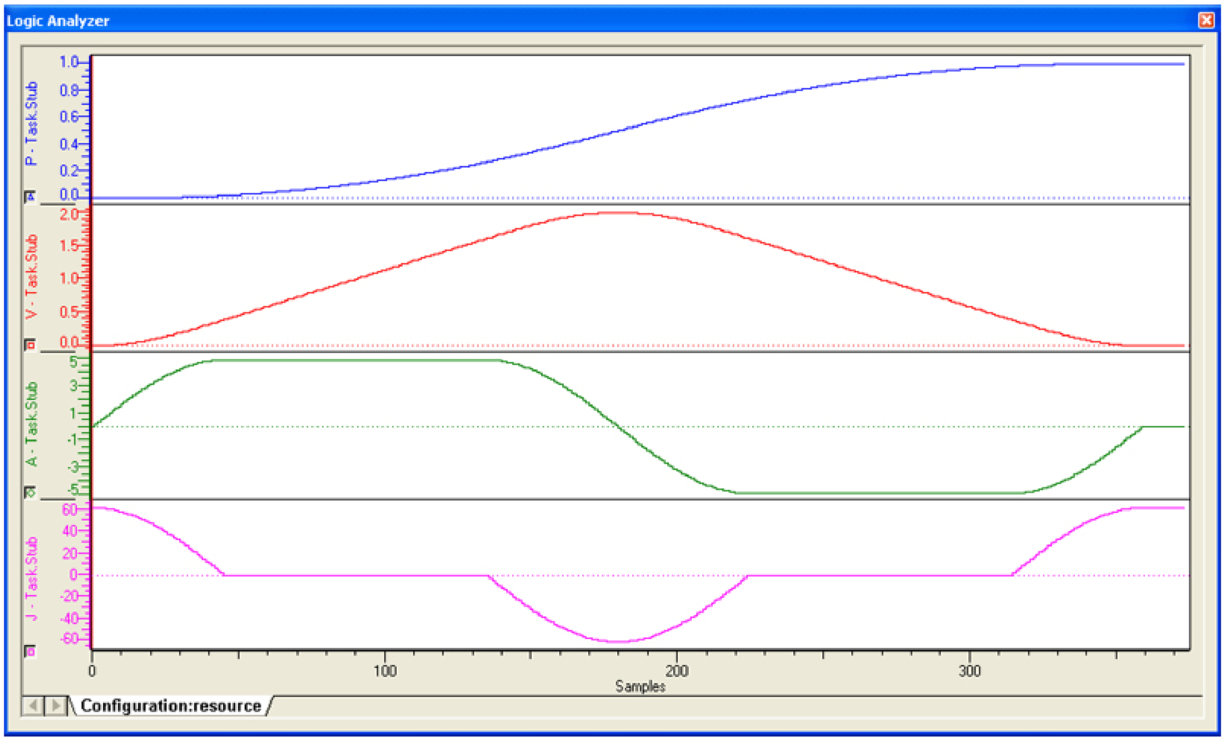
This curve is also one of the discontinuous curves that easily causes vibration, but since it has smooth and good (low) properties, it can be used for low speed applications. When this curve is used for no-dwell applications, (out & back) the discontinuity of acceleration at the starting and end points 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.



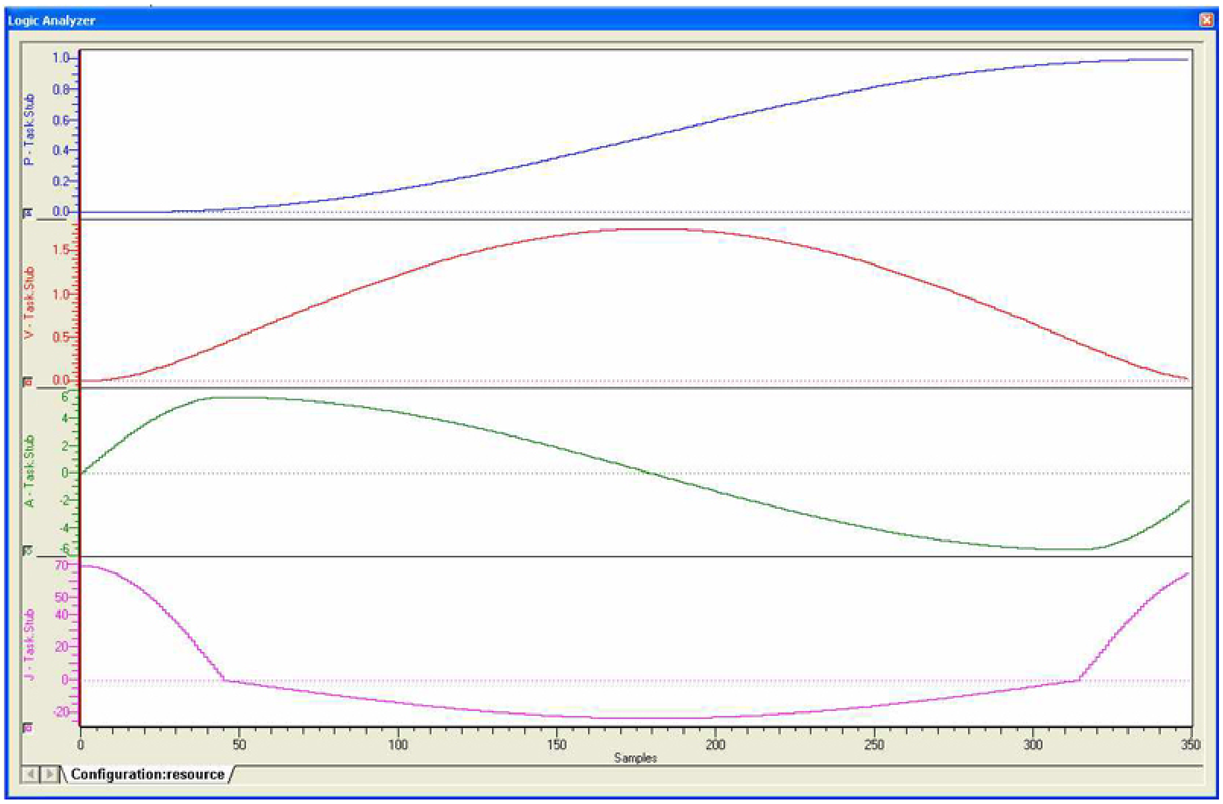
Cycloidal


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

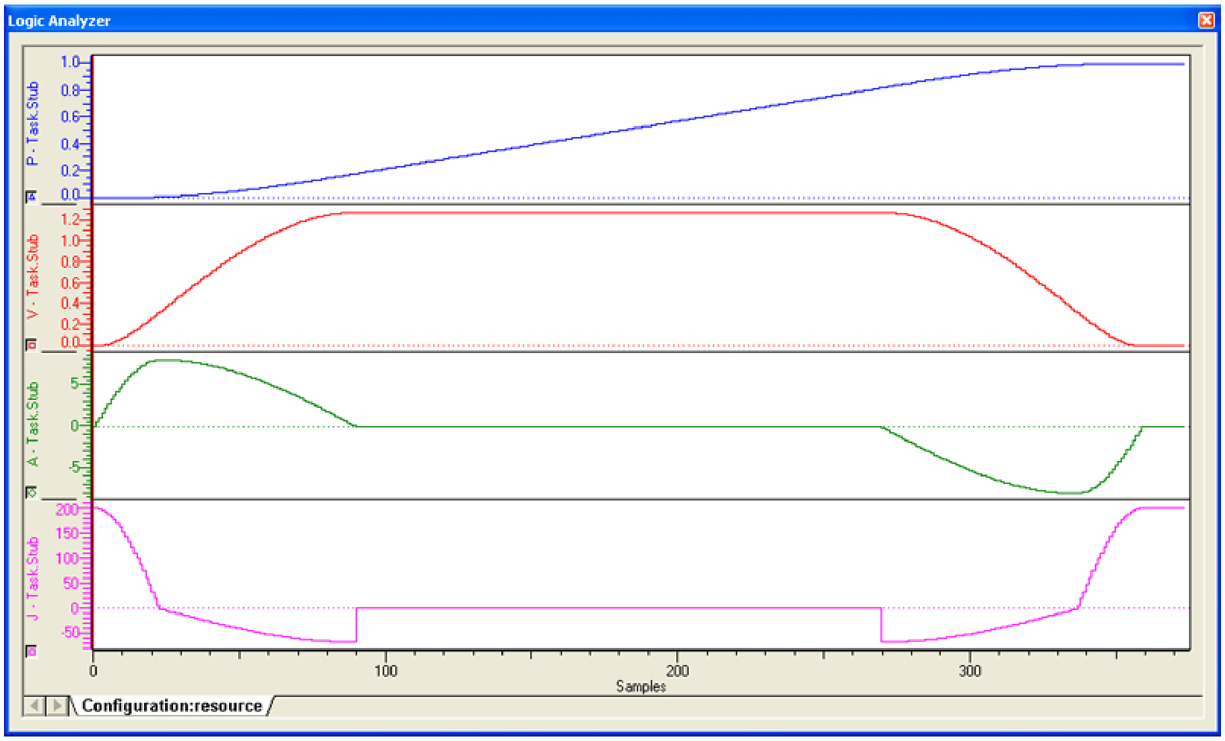
Modified Trapezoid



Modified Sine

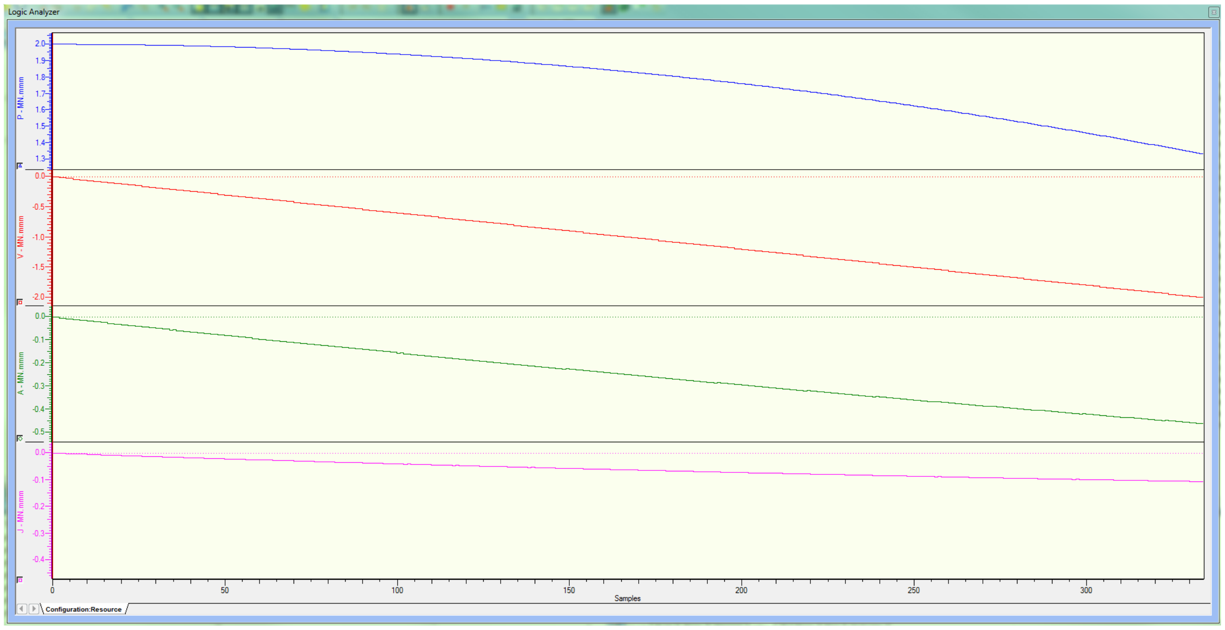


Modified Constant Velocity

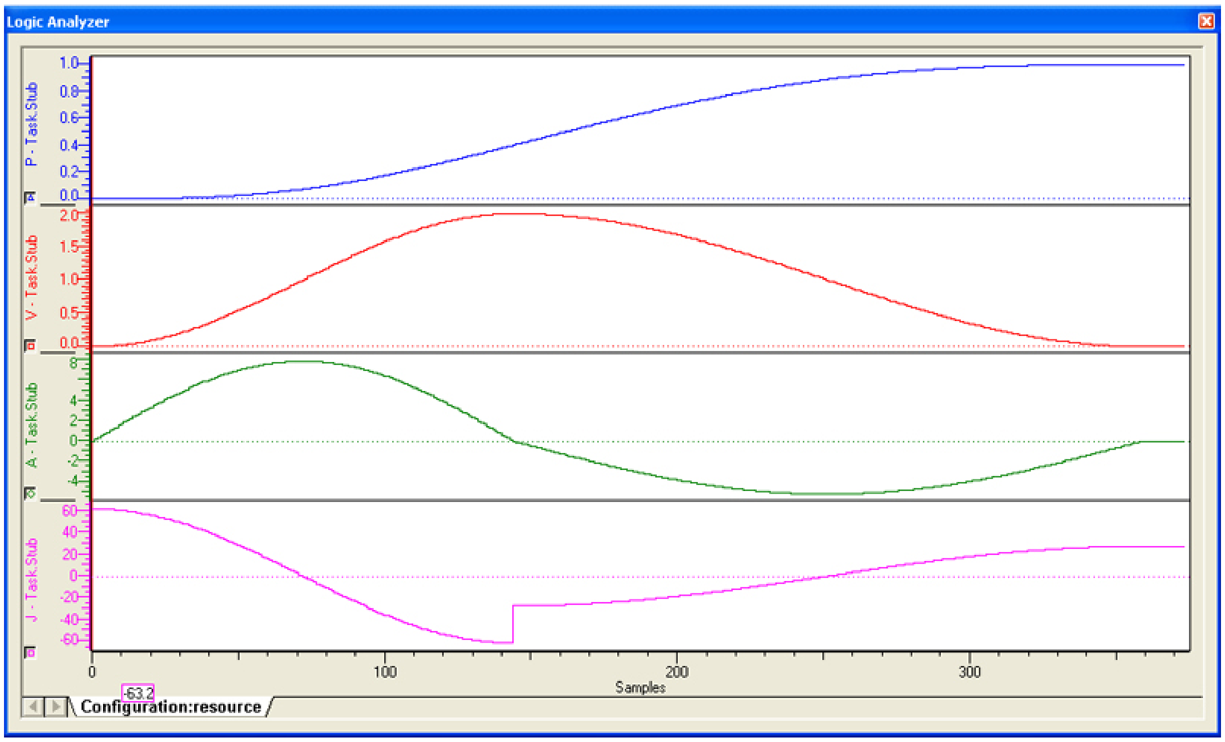


Arc

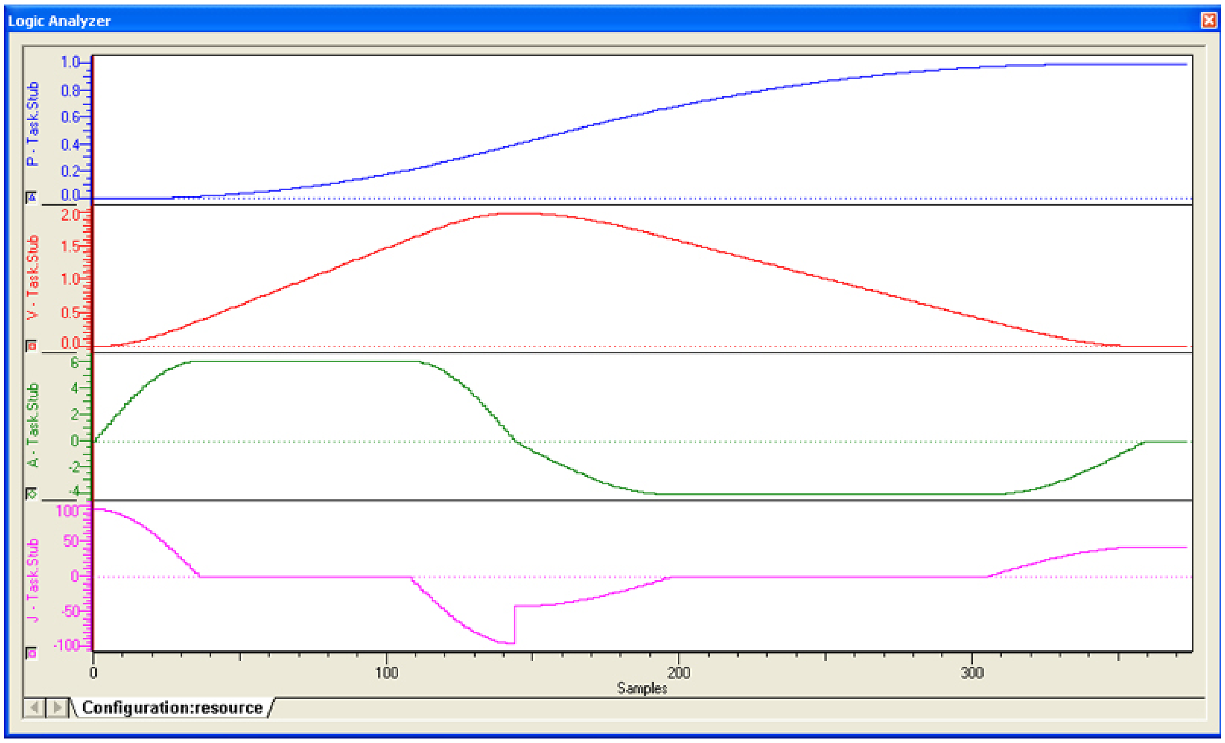
The CamSegmentStruct elements ArcRadius and ArcDirection must be declared for proper usage of this curve type.



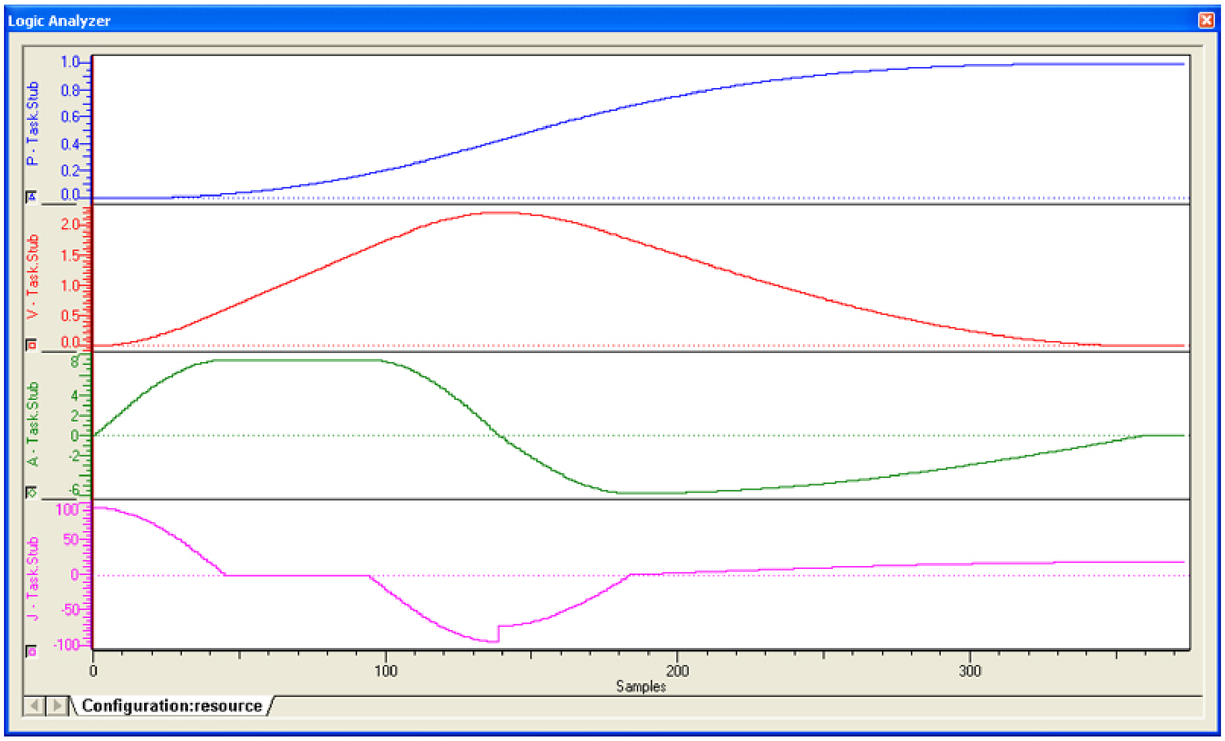
Asymmetrical Cycloidal



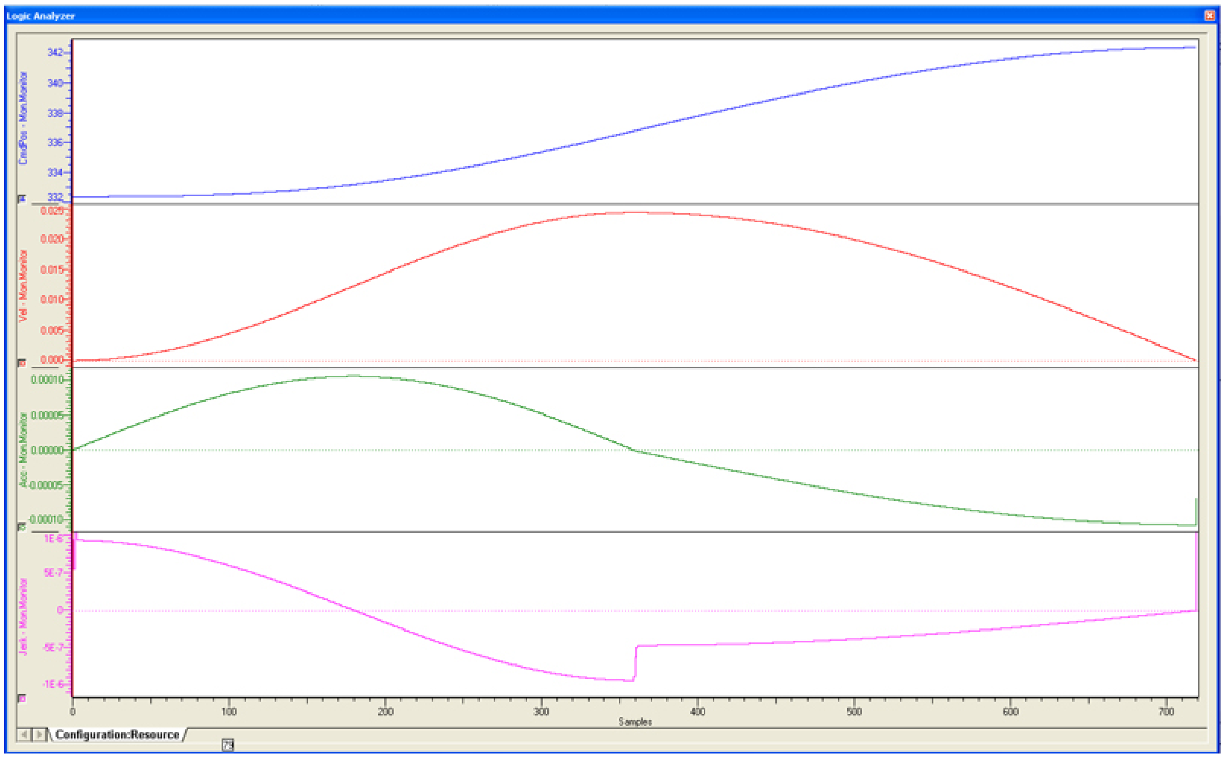
Asymmetrical Modified Trapezoid



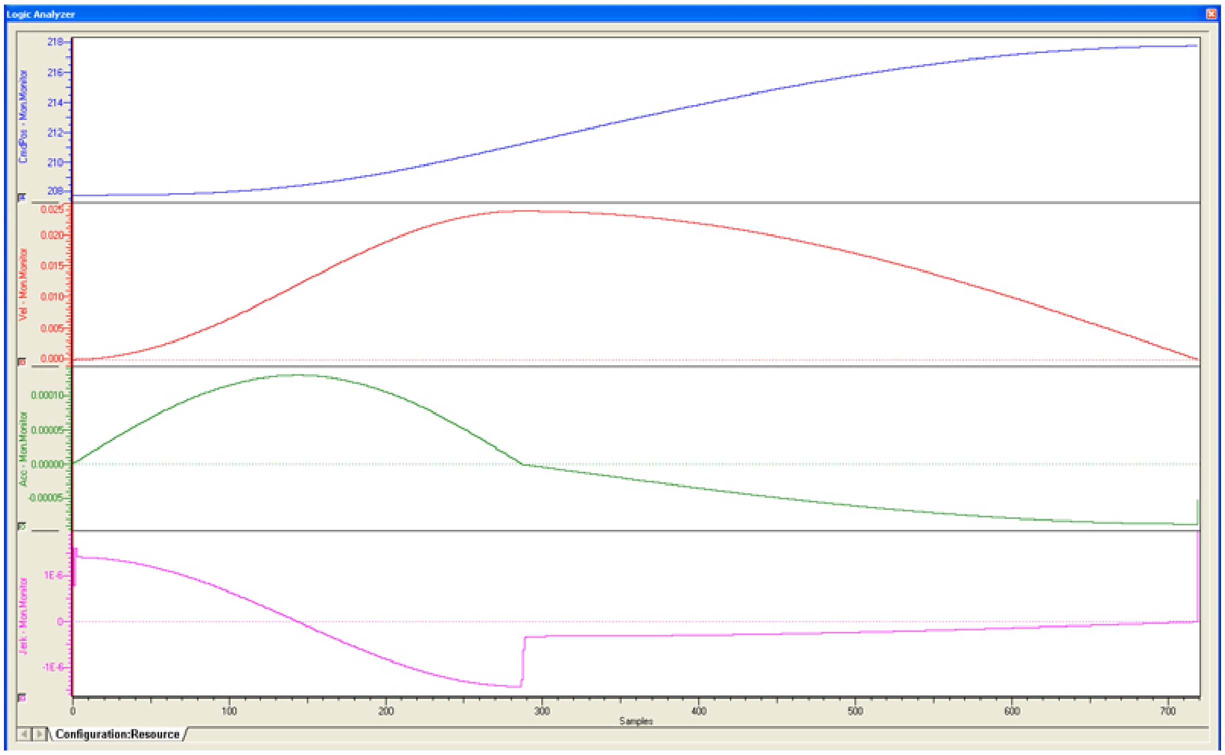
Trapecloid


Logic Analyzer
P - Task.Stub
V - Task.Stub
A - Task.Stub
J - Task.Stub
Samples
Configuration:resource

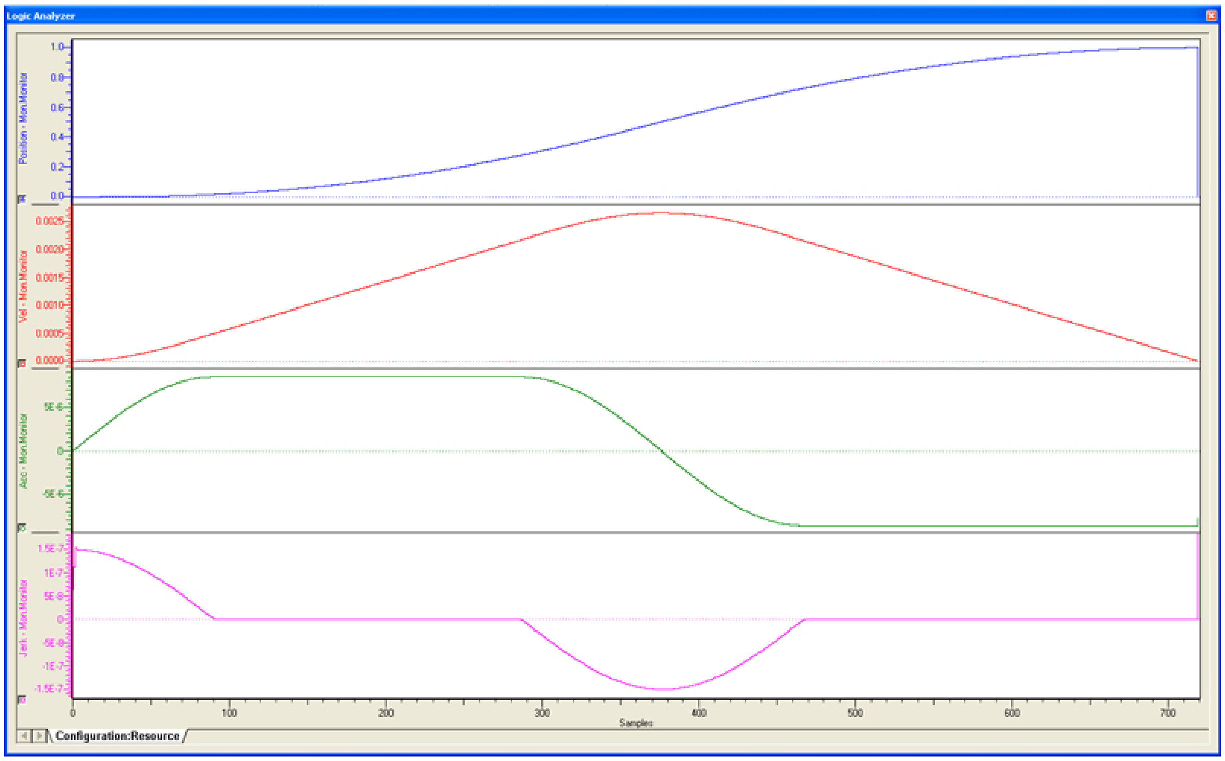
One Dwell Cycloidal_1



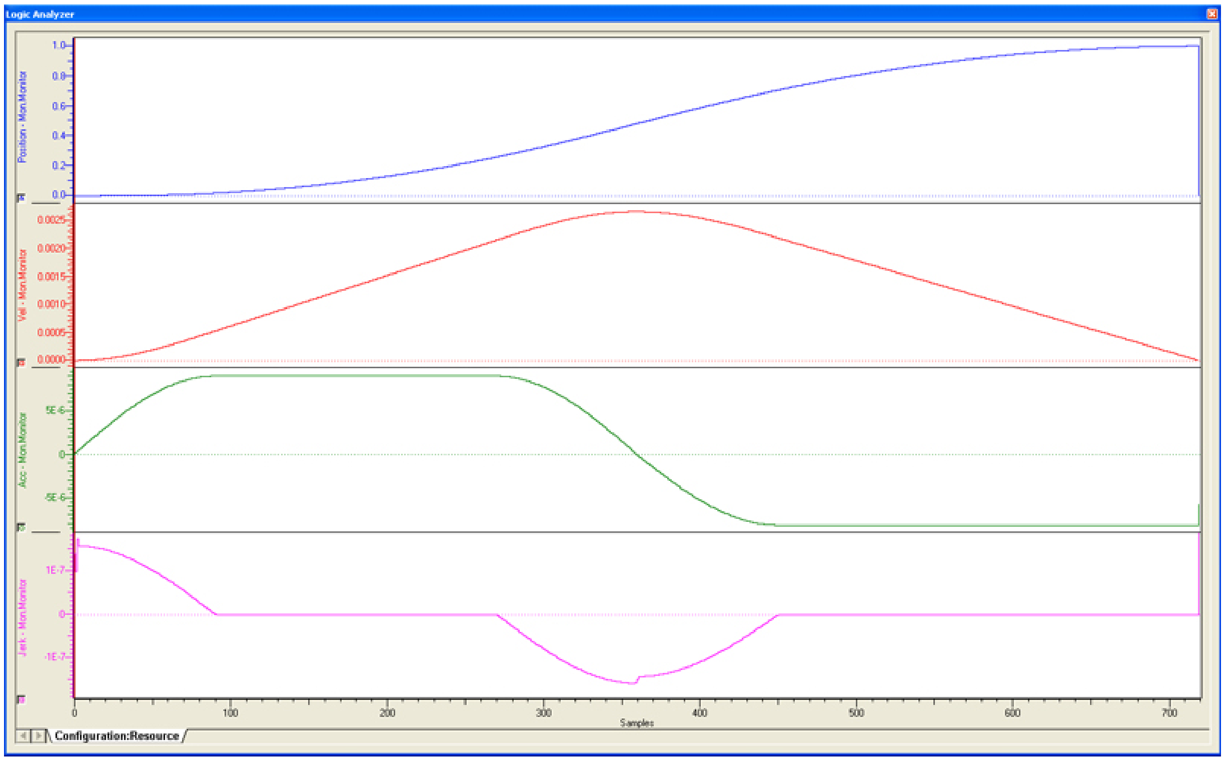
One Dwell Cycloidal_2_3



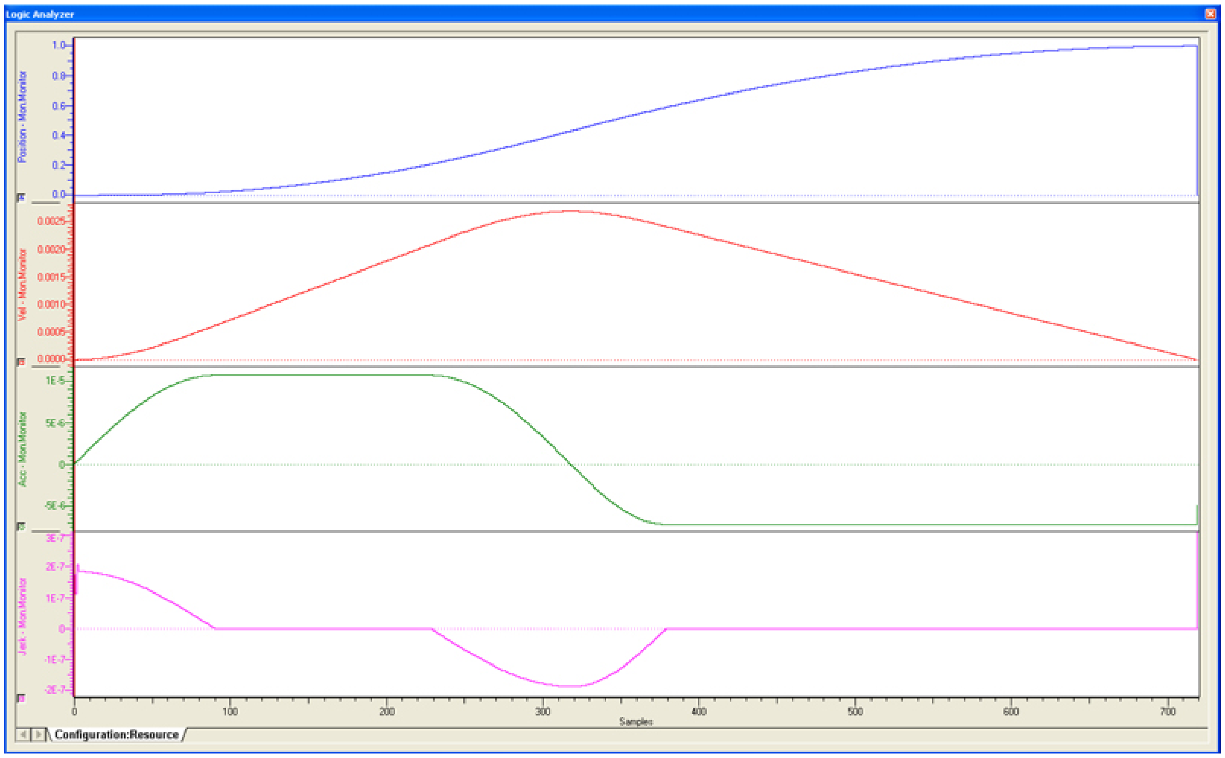
One Dwell Trapezoid_1



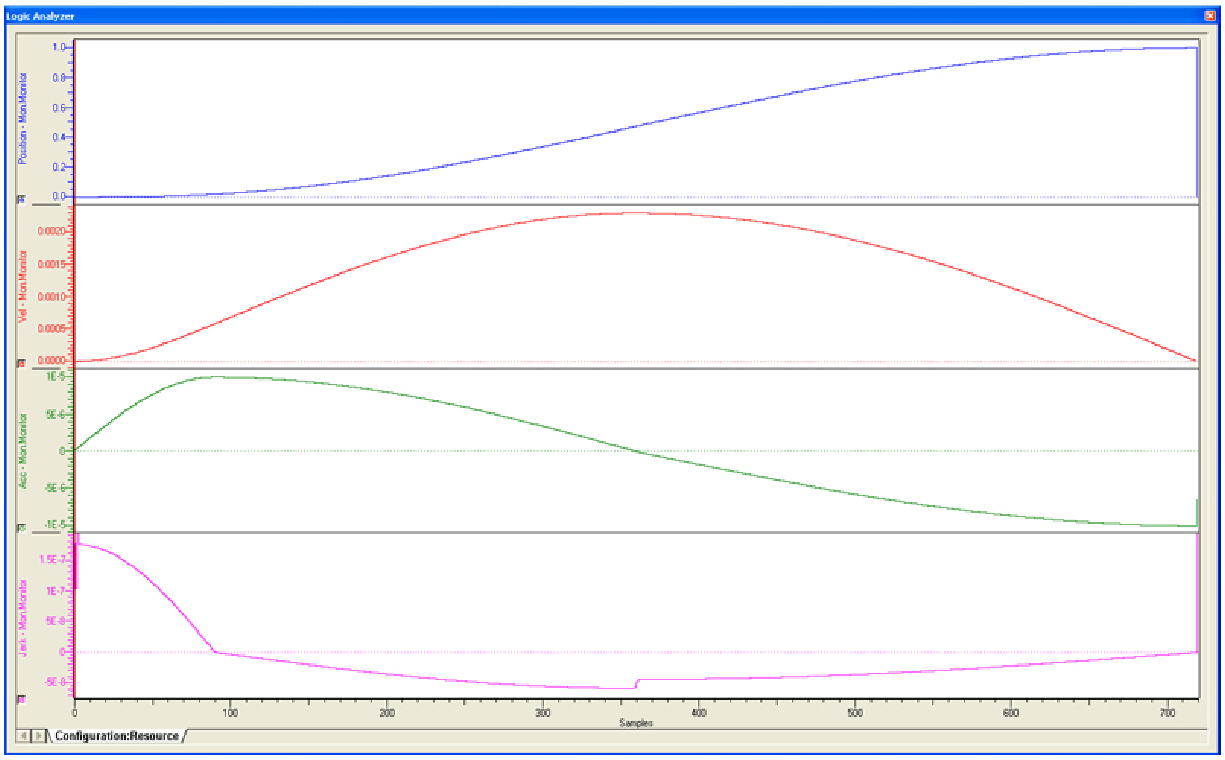
One Dwell Trapezoid



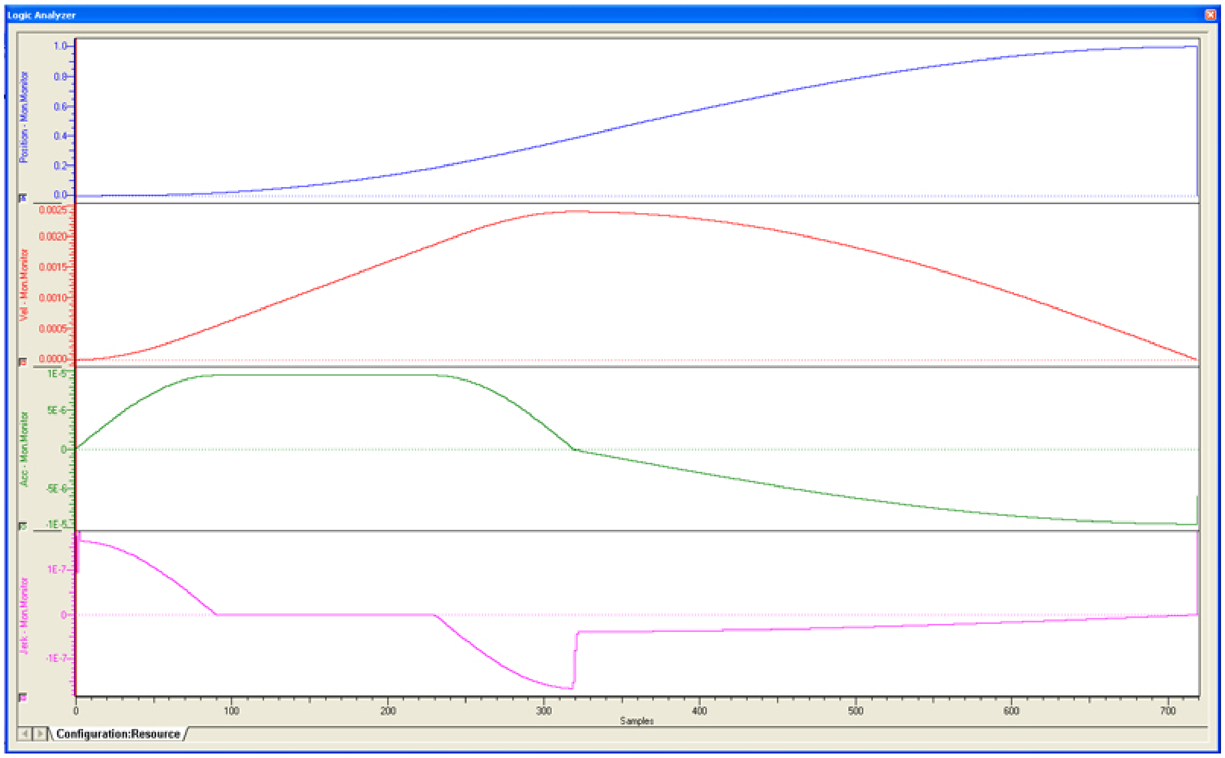
One Dwell Trapezoid_2_3



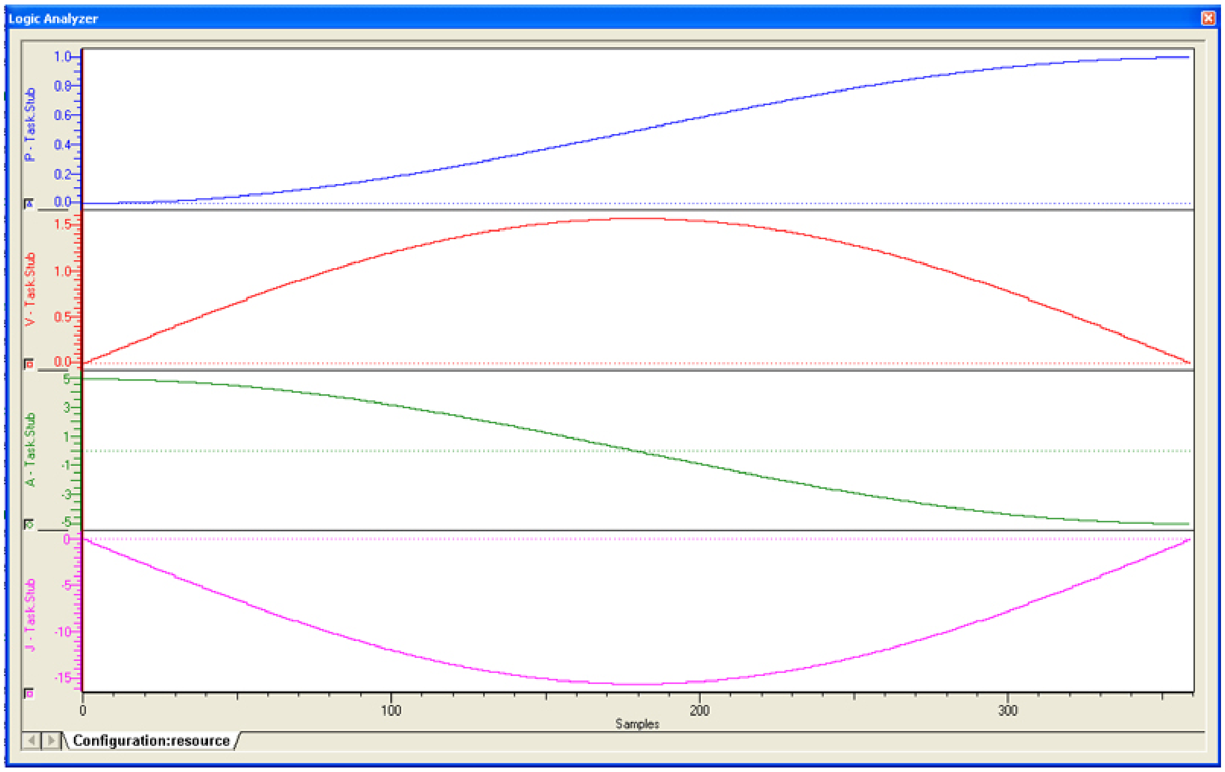
One Dwell Modified Sine



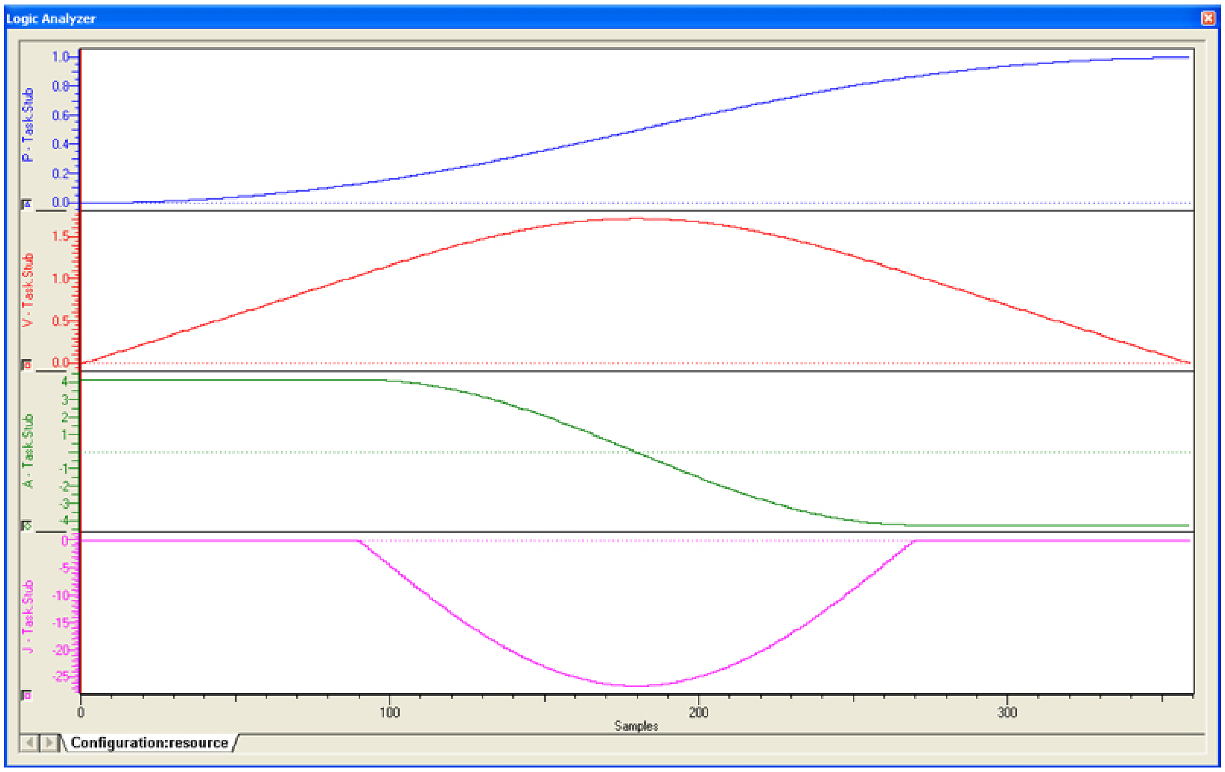
One Dwell Trapecloid



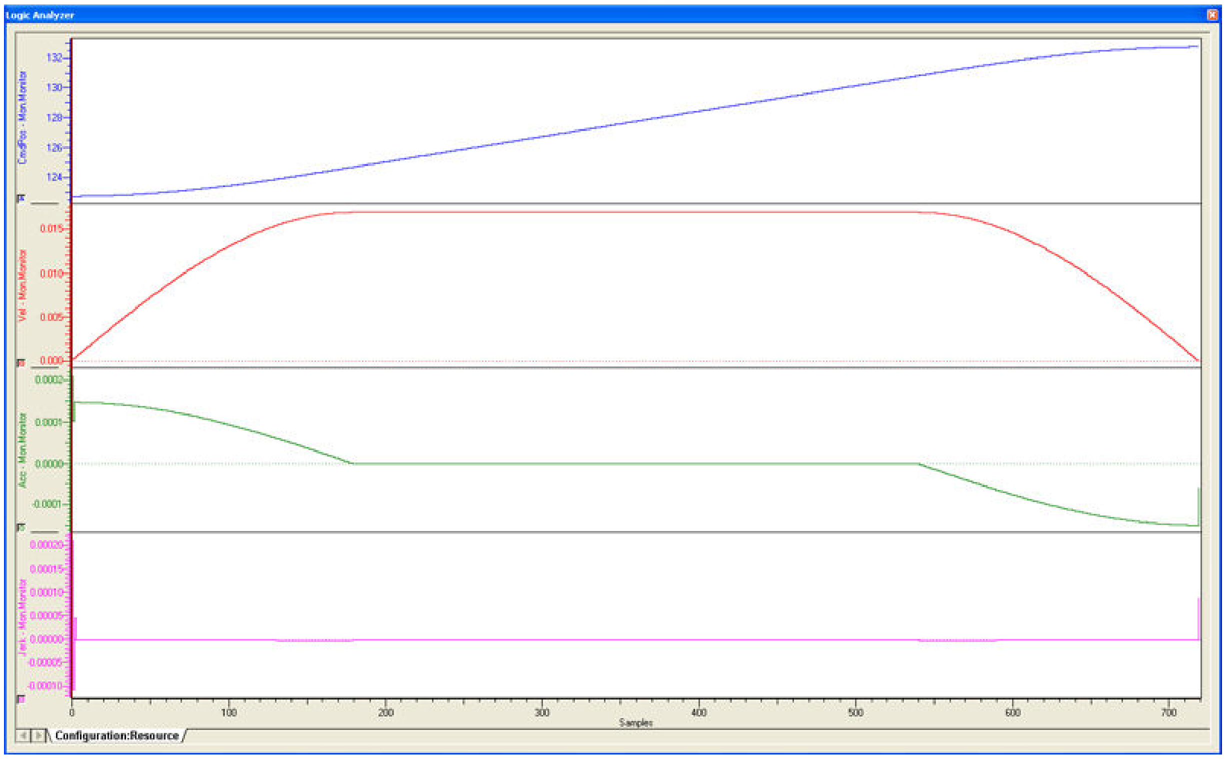
No Dwell Simple Harmonic



No Dwell Modified Trapezoid

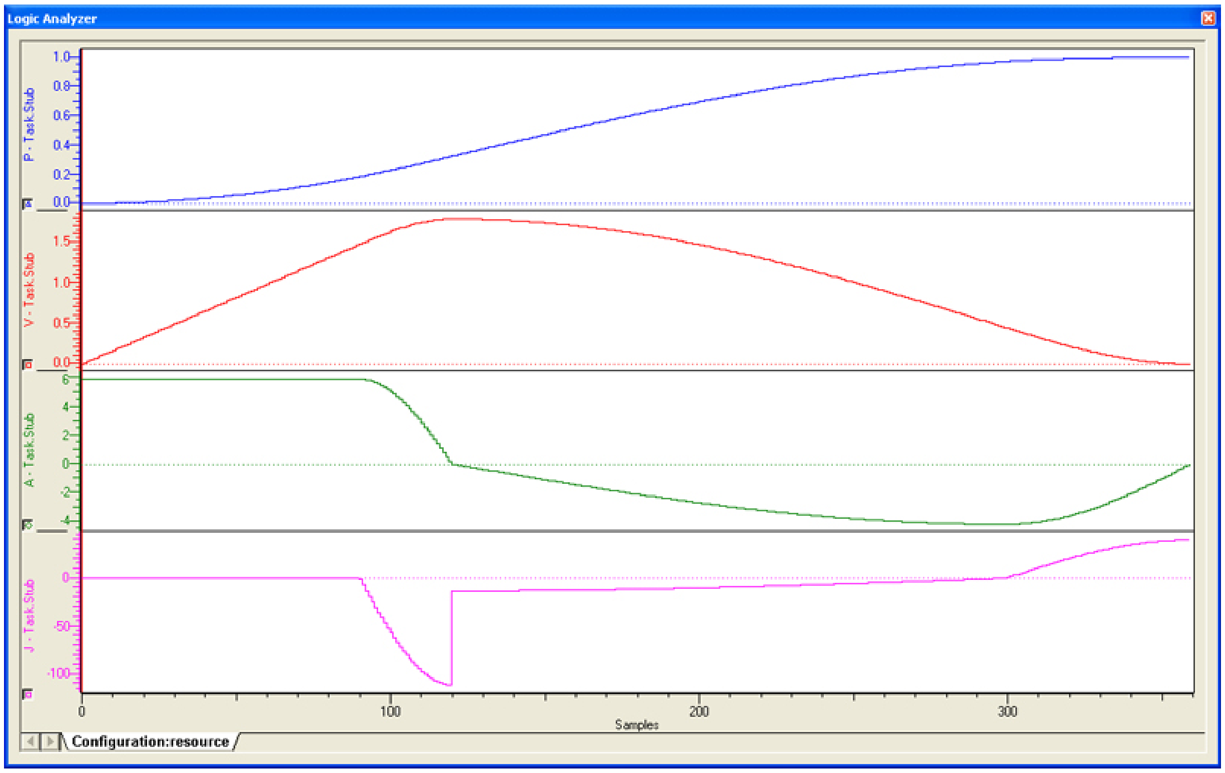


No Dwell Modified Constant Velocity



NC2 Curve

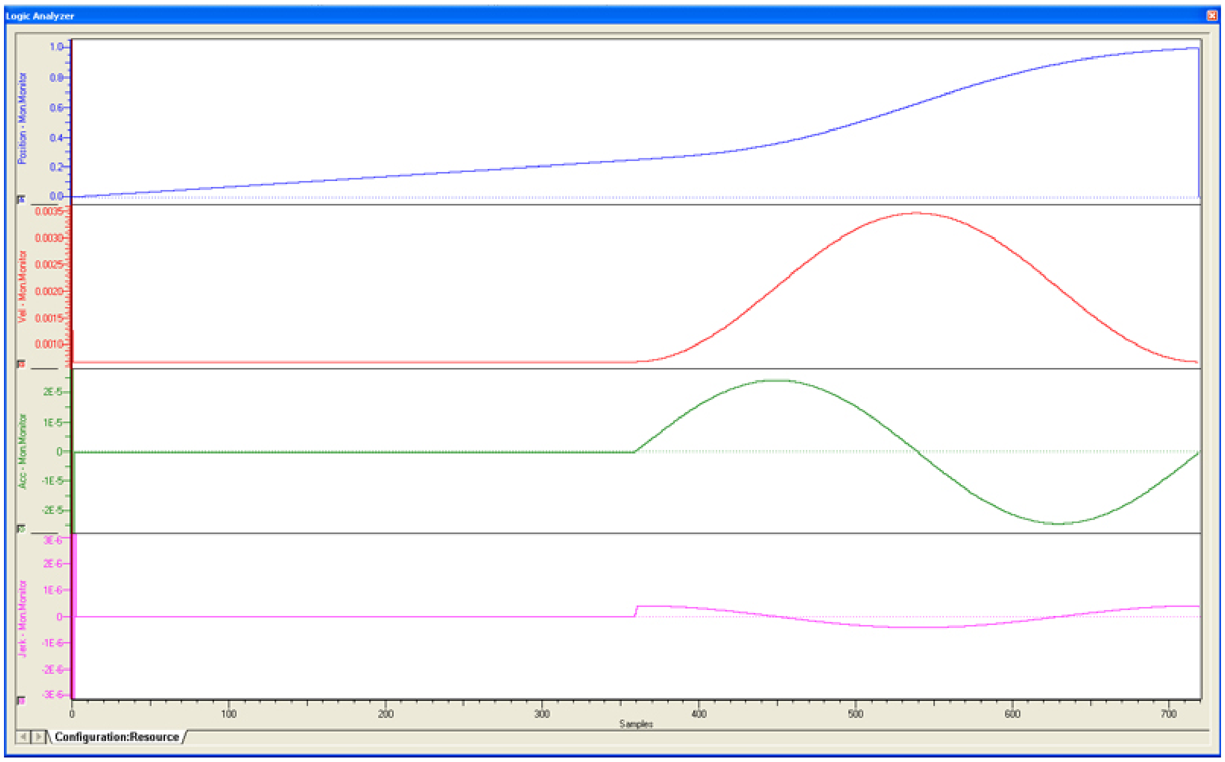
Notes: Deceleration is twice as long as acceleration, which provides the effect of restricting vibration.



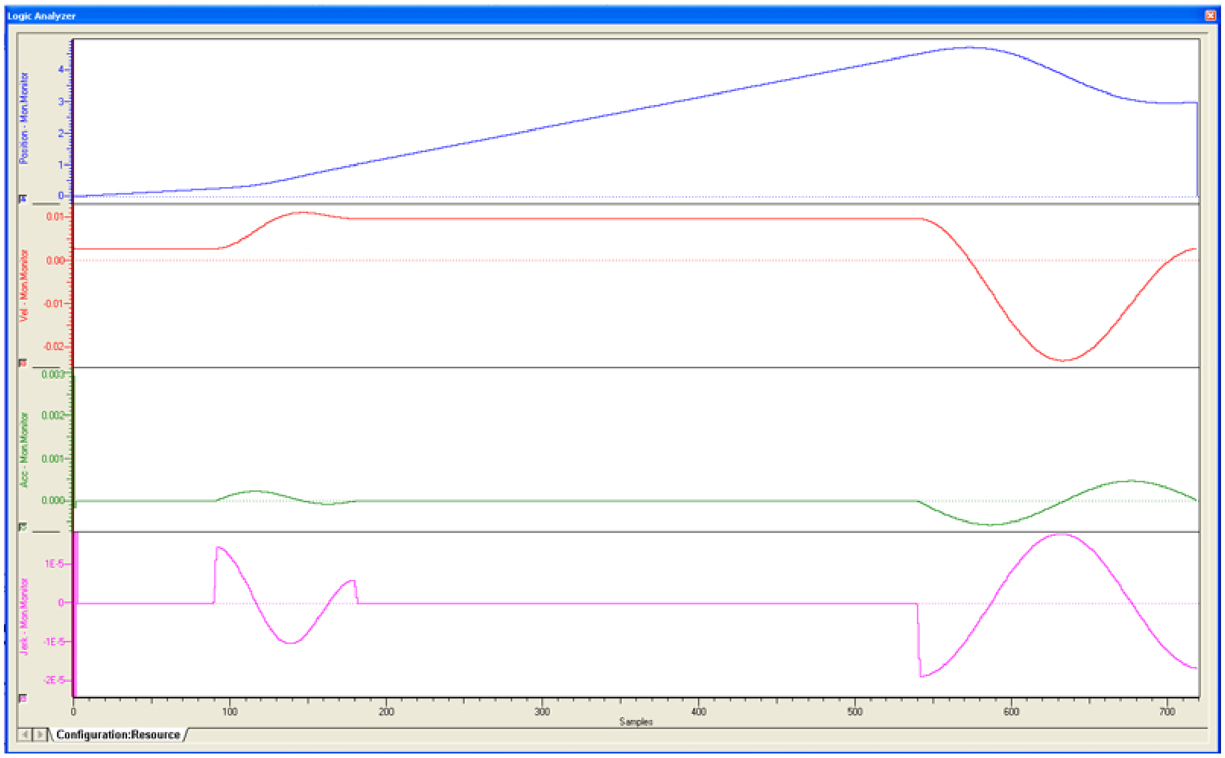
Tangent Matching

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;
```



Reverse Trapecloid

This cam curve type is not currently supported.

Double Harmonic

This cam curve type is not currently supported.

Reverse Double Harmonic

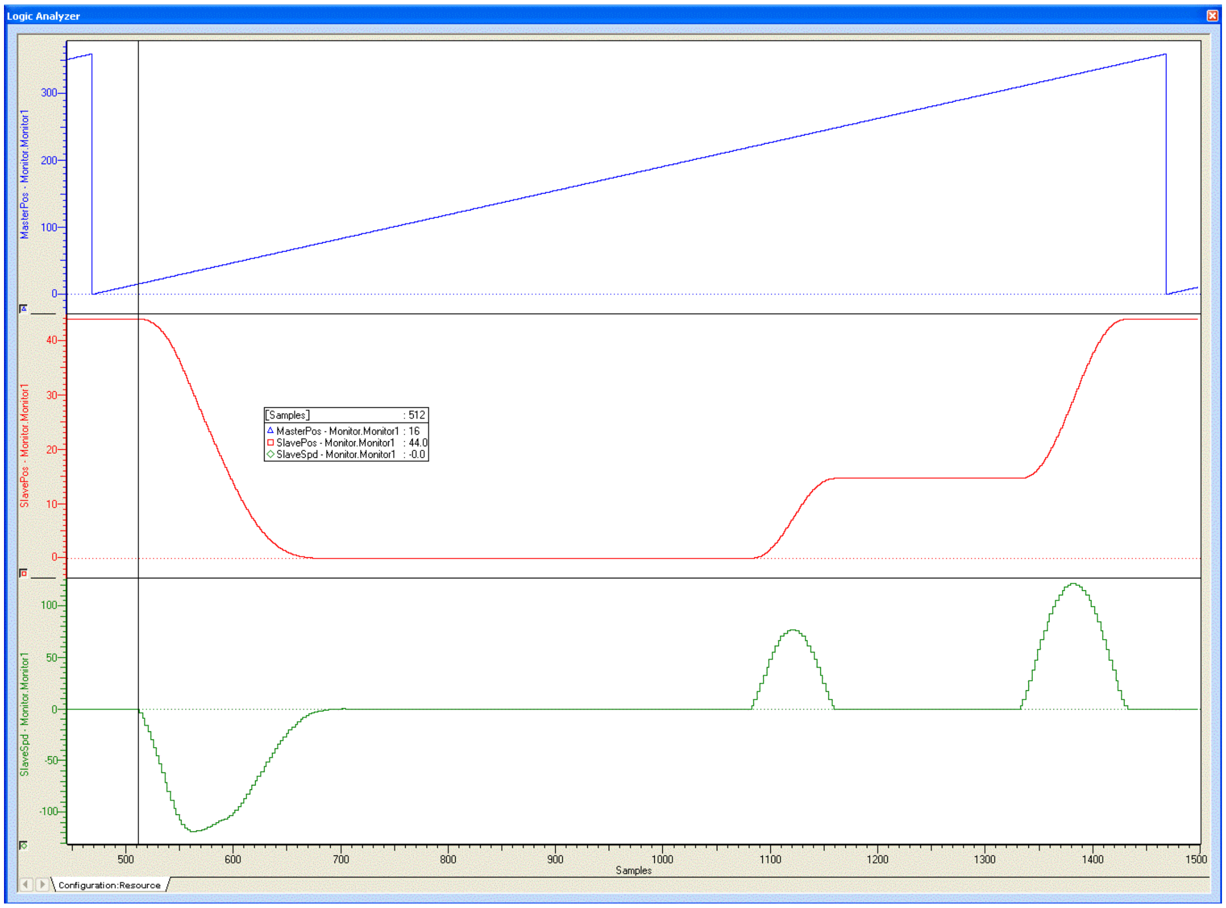
This cam curve type is not currently supported.

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" or blend out profile is created.

See CamBlend function block for application examples

Cubic Spline



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

The CamData values are shown below:

```
(*  test cubic spline *)

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

seg:=INT#1;

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

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

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

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

```
seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#17.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#43.9614;

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


seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#25.5;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#40.3036;

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


seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#34.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#30.4425;

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


seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#42.5;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#19.6003;

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


seg:=Seg + INT#1;
```

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#43.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#19.0;

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



seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#51.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#10.0305;

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



seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#59.5;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#3.5477;

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



seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#68.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#0.6464;

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



seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#76.5;
```

```
Profile4.CamParameters[Seg].SlaveEnd:=LREAL#0.005;

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



seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#85.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#0.0;

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



seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#220.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#0.0;

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



seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#250.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#14.7;

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



seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#310.0;

Profile4.CamParameters[Seg].SlaveEnd:=LREAL#14.7;

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

```
seg:=Seg + INT#1;

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

Profile4.CamParameters[Seg].MasterEnd:=LREAL#348.0;

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

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


seg:=Seg + INT#1;

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

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

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

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


Profile4.LastSegment:=Seg;
```

Data Types

Data Types for Cam Toolbox

The following is a complete list of all DataTypes included in the Cam Toolbox. The list is arranged to separate those that are used internally, and not useful outside of their particular function, and those that an application program must incorporate when the programmer wishes to use the associated Function Block.

Data Type	Usage
DataTypes for use with function blocks in the PLCopen Plus firmware library	
CamStruct	For use with Y_CamIn and Y_CamOut function blocks
DataTypes for external use with Cam Toolbox function blocks	
AXIS_REF	Identifies an axis
AxisParamStruct	For use with the CamSlave_FeedToLength and CamSlave_WindowCheck function blocks.
BlendStruct	For use with the CamBlend function block
CamSegmentStruct	For use with the CamGenerator function block
CamSyncStruct	For use with CamControl and CamShift_Control function blocks
TableIDStruct	For use with the CamTableUpdate function block
TB_CurveType	ENUM type for describing the cam formula to be used.
UINTArray	For use with the CamTableManager Function Block
Y_MS_CAM_STRUCT	For use with the CamGenerator, CamTableUpdate, CamMaster_Lookup, and SlaveIndex_Lookup function blocks.
DataTypes that support other DataTypes (no need for direct use by the application programmer)	
CamParameters	Supporting structure for CamSegmentStruct. For use with the CamGenerator function block
CamSegmentArray	Supporting structure for CamSegmentStruct. For use with the CamGenerator function block
DataTypes used internally by Cam Toolbox function blocks	
CamPairs	Used by the CamGenerator function block
Matrix	For use by the CamGenerator for Cubic Spline calculations
SubMatrix	For use by the CamGenerator for Cubic Spline calculations

Data Type: AXIS_REF

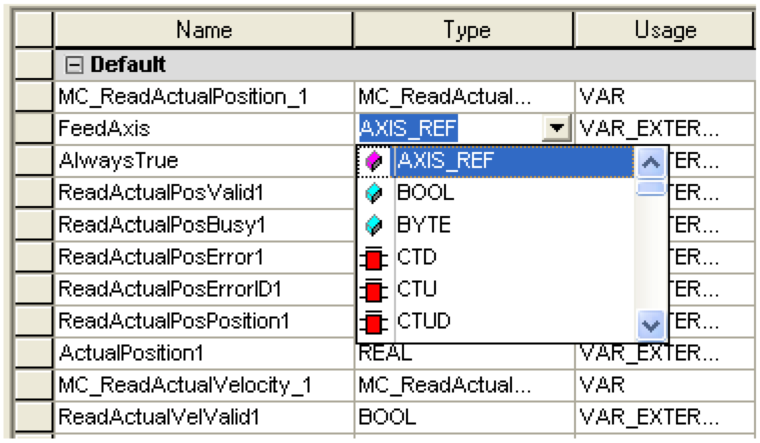
The AXIS_REF data type identifies an axis and thus provides the interface to the hardware or virtual axes. AXIS_REF is used as VAR_IN_OUT in all Motion Control Function Blocks described in this Online help. It is represented as an input and an output connected by a horizontal line in the graphical representation of a function block.

The value of AxisNum is determined by the logical axis number assigned in the Hardware Configuration. See the Configuration tab under each axis.

Data Type Declaration

```
TYPE

AXIS_REF:STRUCT

AxisNum:UINT;    (*  Logical axis number  *)

END_STRUCT;

END_TYPE
```

Variable Declaration Example



Name	Type	Usage
⊟ **Default**		
MC_ReadActualPosition_1	MC_ReadActual...	VAR
FeedAxis	AXIS_REF	VAR_EXTER...
AlwaysTrue	AXIS_REF	...ER...
ReadActualPosValid1	BOOL	...ER...
ReadActualPosBusy1	BYTE	...ER...
ReadActualPosError1	CTD	...ER...
ReadActualPosErrorID1	CTU	...ER...
ReadActualPosPosition1	CTUD	...ER...
ActualPosition1	REAL	VAR_EXTER...
MC_ReadActualVelocity_1	MC_ReadActual...	VAR
ReadActualVelValid1	BOOL	VAR_EXTER...

Code Example

```
AxisX.Number:=UINT#0;
MCMoveAbsoluteX(Axis:=AxisX, Execute:=FALSE);
AxisX:=MCMoveAbsolutX.Axis;
AxisY.Number:=UINT#0;
```

```
MCMoveAbsoluteY(Axis:=AxisY, Execute:=FALSE);
AxisX:=MCMoveAbsolutY.Axis;
```

Data Type: AxisParameterStruct

For use with the CamSlave_FeedToLength and CamSlave_WindowCheck function blocks.

Data Type Declaration

```
TYPE

AxisParameterStruct:STRUCT

ActualPosition:LREAL;               (*  1000   *)

ActualPositionCyclic:LREAL;         (*  1005   *)

ActualPositionNonCyclic:LREAL;      (*  1006   *)

ActualTorque:LREAL;                 (*  1004   *)

ActualVelocity:LREAL;               (*  1001   *)

AtVelocity:BOOL;                    (*  1141   *)

BufferedMotionBlocks:LREAL;         (*  1600   *)

CamMasterCycle:LREAL;               (*  1512   *)

CamMasterPosition:LREAL;            (*  1500   *)

CamMasterShiftedCyclic:LREAL;       (*  1502   *)

CamMasterShiftedPosition:LREAL;     (*  1501   *)

CamMasterScale:LREAL;               (*  1510   *)

CamMasterShift:LREAL;               (*  1511   *)

CamOffset:LREAL;                    (*  1531   *)

CamScale:LREAL;                     (*  1530   *)

CamShiftRemaining:LREAL;            (*  1513   *)

CamState:LREAL;                     (*  1540   *)

CamTableIDEngaged:LREAL;            (*  1541   *)

CamTableOutput:LREAL;               (*  1520   *)

CommandedAcceleration:LREAL;        (*  1012   *)
```

```
CommandedPosition:LREAL;           (*  1010   *)

CommandedPositionCyclic:LREAL;      (*  1015   *)

CommandedPositionNonCyclic:LREAL;   (*  1016   *)

CommandedTorque:LREAL;              (*  1014   *)

CommandedVelocity:LREAL;           (*  1011   *)

InPosition:BOOL;                   (*  1140   *)

LatchPositionNonCyclic:LREAL;       (*  1031   *)

PositionError:LREAL;               (*  1130   *)

PositionWindow:LREAL;              (*  1120   *)

END_STRUCT;

END_TYPE
```

Data Type: BlendStruct

Used by the CamBlend function block

Data Type Declaration

```
TYPE

RampInTableID:UINT;        (*  TableID of the Cam profile that is used to ramp up (accelerate) to the master
speed  *)

RampInSwitchOverPos: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  *)

RunningTableID:UINT;        (*  TableID of the Cam profile that is used in normal operation  *)

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 *)

RampOutTableID:UINT;        (*  TableID of the Cam profile that is used to ramp out (decelerate) to zero speed
 *)

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  *)

Window:LREAL;               (*  Switchover / Engage window  *)

END_TYPE
```

Data Type: CamPairs

Used by the CamGenerator function block

Data Type Declaration

```
TYPE

CamPairs: ARRAY[0..20] OF UDINT;

END_TYPE
```

Data Type: CamParameters

Supporting structure for CamSegmentStruct. For use with the CamGenerator function block.

Data Type Declaration

```
TYPE

CamParameters:STRUCT

MasterEnd:LREAL;  (*  Location of the master at the end of the current segment  *)

SlaveEnd:LREAL;   (*  Location of the slave at the end of the current segment  *)

CurveType:INT;    (*  Formula code to indicate the motion profile for this segment  *)

Resolution:REAL;  (*  Determines how many data points are calculated along this segment.  *)

                  (*  If the master delta from the previous segment is 10.0, and the resolution  *)

                  (*  is set to 0.5, then 20 datapoints will be generated for this segment.  *)

END_STRUCT;

END_TYPE
```

Data Type: CamSegmentArray

Supporting structure for CamSegmentStruct. For use with the CamGenerator function block.

Data Type Declaration

```
TYPE

CamSegmentArray: ARRAY[0..200] OF CamParameters;

END_TYPE
```

Data Type: CamSegmentStruct

For use with the CamGenerator function block.

Data Type Declaration

```
TYPE

CamSegmentStruct:  STRUCT

CamParameters:  CamSegmentArray;

SlaveStart:  LREAL;

LastSegment:  INT;

ArcRadius:  LREAL;    (*  Only used with 'Arc' CurveType  *)

ArcDirection:  INT;    (*  1: ccw,  -1: cw  *) (*  Only used with 'Arc' CurveType  *)

END_STRUCT;

END_TYPE
```

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

```
TYPE

CamStruct:  STRUCT

FileName:STRING;         (*  Filename that will be used by Y_CamFileSelect    *)

TableType:INT;          (*  0=Undefined, 1=M/S pair, 2=reserved, 3=reserved  *)

TableSize:UDINT;         (*  The size of the cam table in bytes (Don't forget, 16 bytes per M/S pair)   *)

TableID:UINT;           (*  Number returned from Y_CamFileSelect  *)

EngagePosition:LREAL;    (*  Master location where slave must start synchronization

(Reference prm 1502 - CamMasterShiftedCyclic  *)

EngageData:Y_ENGAGE_DATA;

DisengagePosition:LREAL; (*  Master location where slave must stop synchronization

(Reference prm 1502 - CamMasterShiftedCyclic  *)

DisengageData:Y_DISENGAGE_DATA;

Window:LREAL;            (*  Size of the window in master units where the engage or disengage

 will take place  *)

MasterCycle:LREAL;

SlaveCycle:LREAL;

END_STRUCT;

END_TYPE
```

Data Type: CamSyncStruct

For use with the CamControl and CamShift_Control function blocks

Data Type Declaration

CamSyncStruct: STRUCT

Mode:INT; (* User Input. 1 = Rotary Knife; 2 = Linear Flying Shear, 3 = Rotary Placer or Reciprocating Drill *)

StartSyncPosition:LREAL; (* User Input. The first master position where the slave must be synchronized with the master *)

EndSyncPosition:LREAL; (* User Input. The final master position where the slave must be synchronized with the master, adjustments can start after. *)

DecisionPosition:LREAL; (* User Input. Key location in the process where the machine must decide to start the disengage process, or continue camming and CamShift to the next product. *)

MaxShift:LREAL; (* User Input. If Mode = 3, this value is used to determine if the slave should advance or retard to synchronize with the next product. *)

SafeEngageDistance:LREAL; (* Distance traveled from the sensor until the product is less than one machine cycle away from the synchronization point. *)

Shifting:BOOL; (* System Output flag set by the CamShift_Control function block *)

Pause:BOOL; (* System Output flag set by the CamControl function block if the system was temporarily disengaged *)

END_STRUCT;

Data Type: Matrix

For use by the CamGenerator for Cubic Spline calculations

Data Type Declaration

```
Matrix : ARRAY [0..20] OF SubMatrix;
```

Data Type: SubMatrix

For use by the CamGenerator for Cubic Spline calculations

Data Type Declaration

```
SubMatrix : ARRAY [0..20] OF LREAL;
```

Data Type: TableIDStruct

For use with the CamTableUpdate function block

Data Type Declaration

```
TYPE

TableIDStruct:STRUCT

Inactive:UINT;  (*  The CamTableID that is NOT currently being accessed to control motion   *)

Active:UINT;    (*  The CamTableID that IS currently being accessed to control motion  *)

END_STRUCT;

END_TYPE
```

Data Type: UINTArray

For use with the CamTableManager Function Block

Data Type Declaration

```
UINTArray: ARRAY[0..4] OF UINT;        (*  An array for CamTableIDs that are released from memory in a FIFO method.  *)
```

Data Type: Y_MS_CAM_STRUCT

This data type is for use with the CamGenerator, CamMaster_Lookup, CamTableUpdate, and SlaveIndex_Lookup function blocks.

Data Type Declaration

```
TYPE

    Y_CAM_HEADER:STRUCT

        TableType:INT;            (*  INT#1 = Master/Slave pair  *)

        Reserved1:UINT;

        DataSize:UDINT;           (*  Size of cam table in bytes, 16 bytes per Y_MS_PAIR  *)

    END_STRUCT;


    Y_MS_PAIR: STRUCT

        Master:LREAL;             (*  Master position  *)

        Slave:LREAL;              (*  Slave position   *)

    END_STRUCT;


    Y_MS_HEADER:STRUCT

        SlaveIncremental:BOOL;    (*  If TRUE, then the slave data from pair to pair is relative.  *)

        MasterIncremental:BOOL;   (*  If TRUE, then the master data from pair to pair is relative.  *)

        Reserved1:UINT;

        Reserved2:UINT;

        Reserved3:UINT;

    END_STRUCT;


    MS_Array_Type:ARRAY[0..2880] OF Y_MS_PAIR;
    Y_MS_CAM_STRUCT:STRUCT

        Header:Y_CAM_HEADER;

        MS_Header:Y_MS_HEADER;

        MS_Data:MS_Array_Type;

    END_STRUCT;

END_TYPE
```

Enumerated Types

Enumerated Type: TB_CurveType

ENUM type for describing the cam formula to be used.

Data Type Declaration

```
(*  ENUM Type for CurveType  *)

TB_CurveType:

(

na,                                    (*   INT#0   - Not a valid CurveType   *)

StraightLine,                 (*  INT#1  - Straight Line *)

Parabolic,                    (*  INT#2  - Parabolic *)

SimpleHarmonic,               (*  INT#3  - Simple Harmonic *)

Cycloidal,                    (*  INT#4  - Cycloidal *)

ModifiedTrapezoid,            (*  INT#5  - Modified Trapeziod *)

ModifiedSine,                 (*  INT#6  - Modified Sine *)

ModifiedConstVelocity,        (*  INT#7  - Modified Constant Velocity *)

AsymmetricalCycloidal,        (*  INT#8  - Asymmetrical Cycloidal *)

AsymmetricalModifiedTrapezoid, (*  INT#9  - Asymmetrical Modified Trapezoid *)

Trapecloid,                   (*  INT#10 - Trapecloid *)

OneDwellCycloidal_1,          (*  INT#11 - One Dwell Cycloidal m=1 *)

OneDwellCycloidal_2_3,        (*  INT#12 - One Dwell Cycloidal m=2/3 *)

OneDwellTrapezoid_1,          (*  INT#13 - One Dwell Trapezoid m=1 *)

OneDwellTrapezoid,            (*  INT#14 - One Dwell Trapezoid *)

OneDwellTrapezoid_2_3,        (*  INT#15 - One Dwell Trapezoid m=2/3 *)

OneDwellModifiedSine,         (*  INT#16 - One Dwell Modified Sine *)
OneDwellTrapecloid,           (*  INT#17 - One Dwell Trapecloid *)

NoDwellSimpleHarmonic,        (*  INT#18 - No Dwell Simple harmonic *)

NoDwellModifiedTrapezoid,     (*  INT#19 - No Dwell Constant Velocity *)

NoDwellModifiedConstVelocity, (*  INT#20 - No Dwell Modified Constant Velocity *)

NC2Curve,                     (*  INT#21 - NC2 Curve *)

TangentMatching,              (*  INT#22 - Tangent Matching *)

ReverseTrapecloid,            (*  INT#23 - Reverse Trapecloid *)

DoubleHarmonic,               (*  INT#24 - Double Harmonic *)

ReverseDoubleHarmonic,        (*  INT#25 - Reverse Double Harmonic *)

TangentBlending,              (*  INT#26 - Tangent Blending *)

Unsupported27                 (*  INT#27 - Unsupported  *)

Unsupported28                 (*  INT#28 - Unsupported  *)

UserModifiedConstVelocity,    (*  INT#29 - User Modified Constant Velocity - To specify the accel / decel
distances *)

Polynomial345                 (*  INT#30 - 5th order polynomial with C3 = 10, C4 = -15, C5 = 6   *)

CubicSpline                   (*  INT#31 - Cubic spline interpolation  *)

Arc                           (*  INT#32 - Arc  *)

ParabolicVelocityBlend        (*  INT#33 - Parabolic curve with velocity blending  *)

);
```

Enumerated Type: TB_Mode

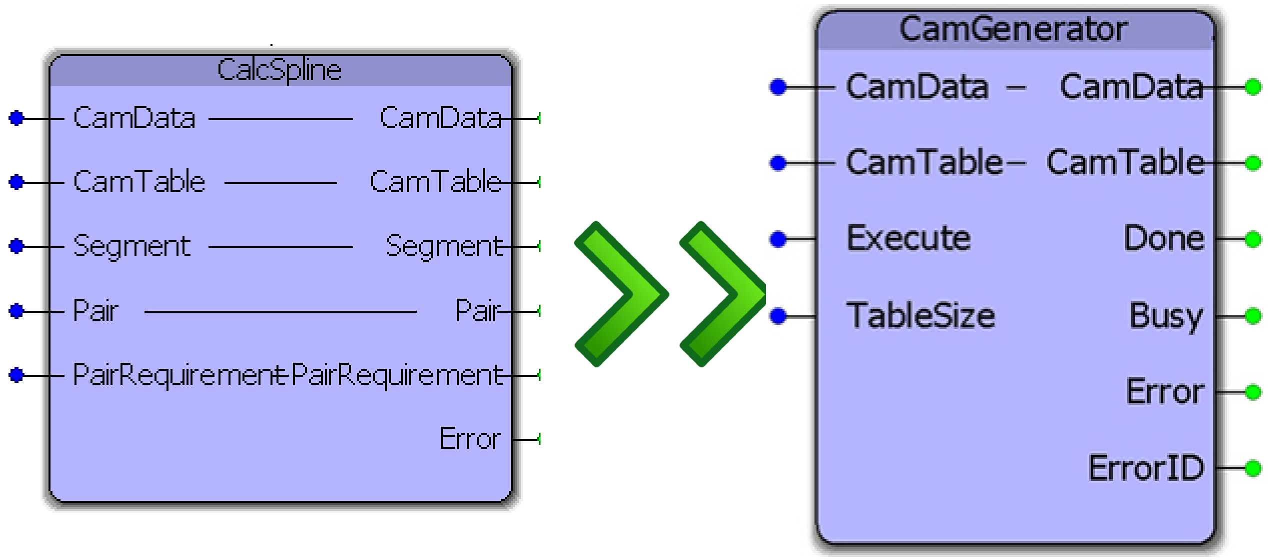
ENUM Type for CamShift_Control to select rotary or linear slave motion

Data Type Declaration

```
TB_Mode:

(

na, (*  INT#0  - Not a valid Mode   *)

Reciprocating, (*  INT#1  - Reciprocating, like Rotary Placer, Rotary Knife, etc.  *)

OutAndBack (*  INT#2  - Out and Back, like linear flying shear, walking beam, bottle filler  *)

);
```

Function Blocks

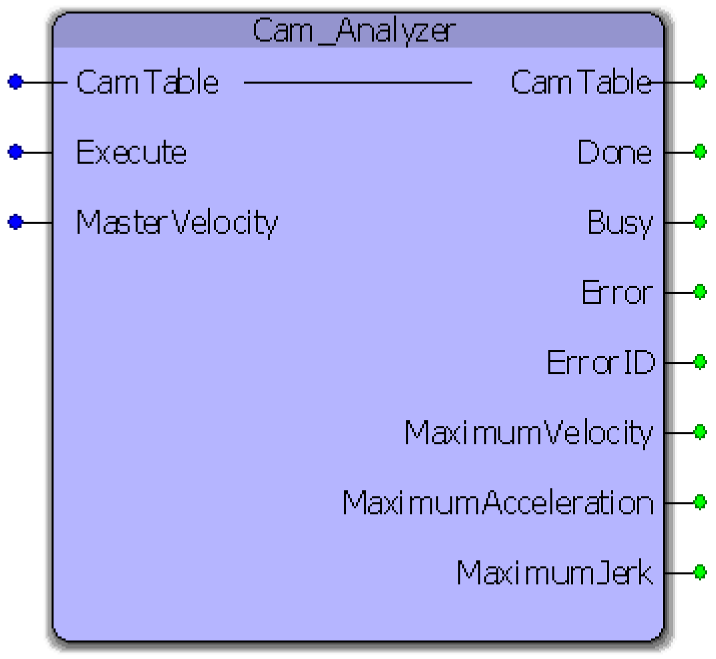
CalcSpline



This function block is for internally calculating the Cubic Spline cam curve type.

Attention: This function block is not intended for end user implementation. Its functionality is a requirement for the Cam Toolbox user library. To use this functionality, please refer to the function block CamGenerator.

Cam_Analyzer



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

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	CamTable	Y_MS_CAM_STRUCT	This structure contains the resulting master/slave information for each datapoint 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.)	0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has	

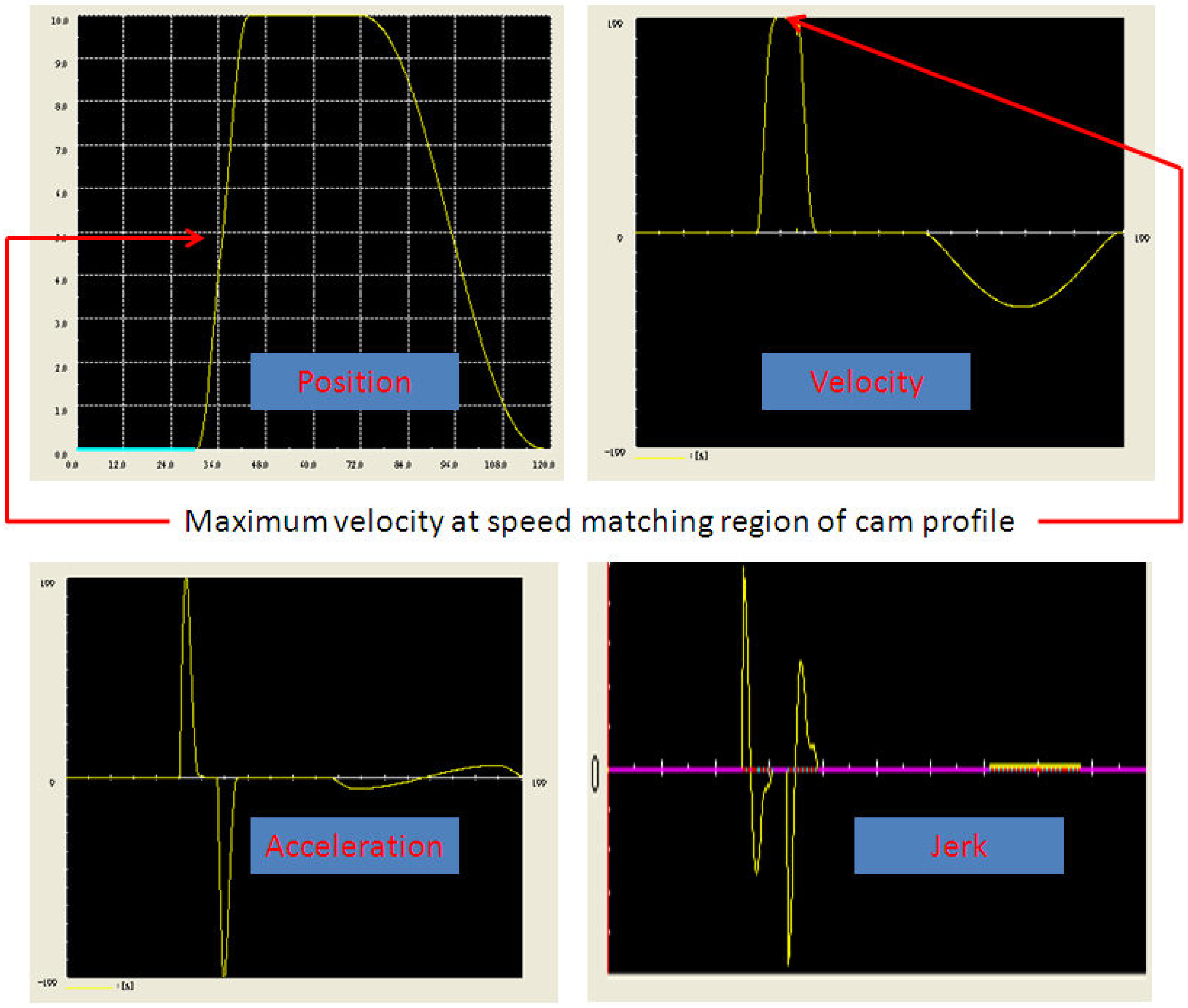
			been completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.
B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

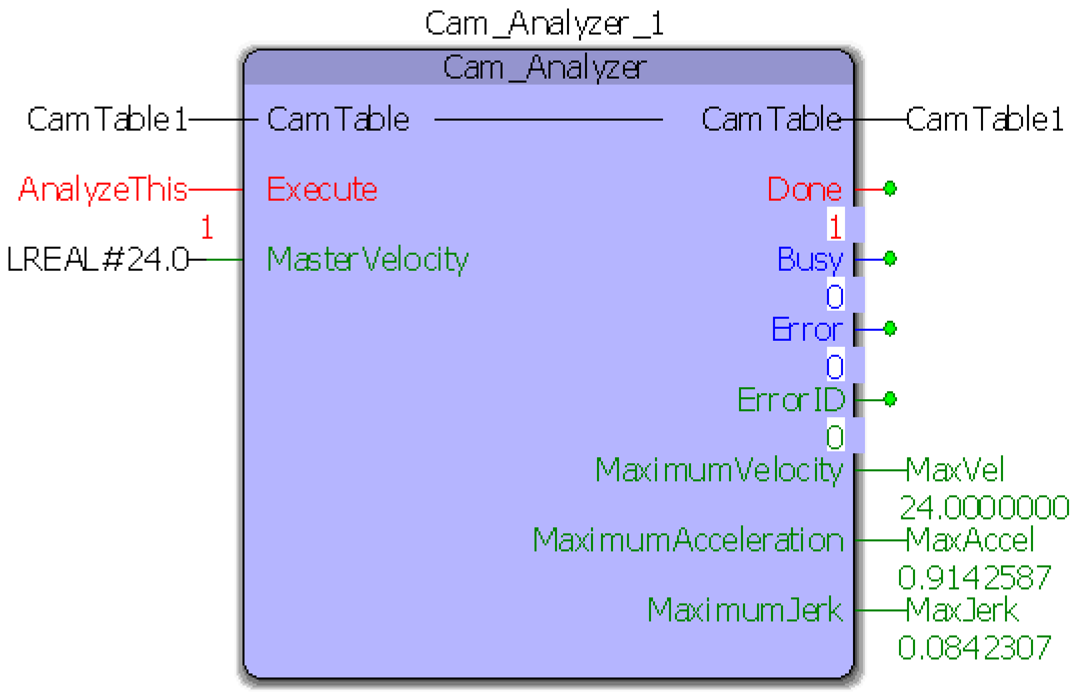
Error Description

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

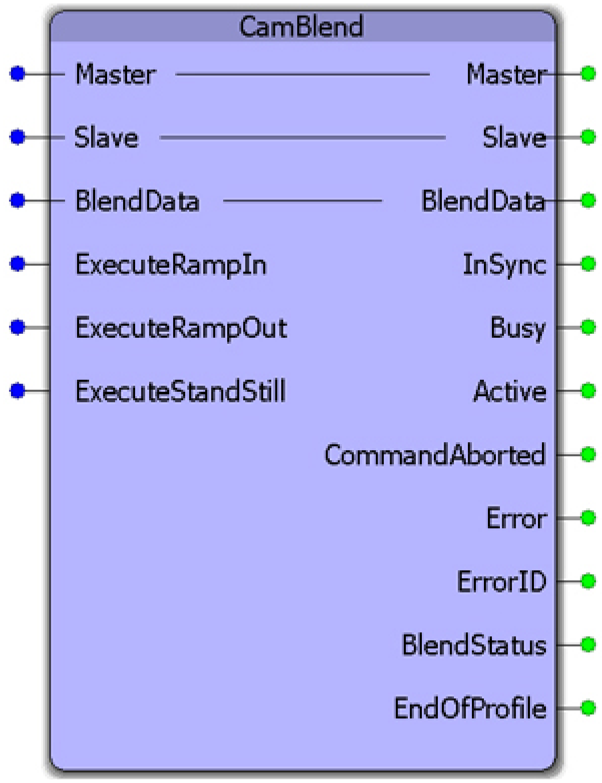
Example

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.


Position
Velocity
Maximum velocity at speed matching region of cam profile
Acceleration
Jerk
10.0
9.0
8.0
7.0
6.0
4.0
3.0
2.0
1.0
0.0
12.0
24.0
36.0
48.0
60.0
72.0
84.0
96.0
108.0
120.0
100
0
-100
[s]


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.

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	FALSE

			prepare to switch to the RampOut cam profile at the SwitchOver position specified in the BlendData structure.	
V	ExecuteStandStill	BOOL	Upon the rising edge, this function block will prepare to engage the slave to the Running cam profile at the StandstillEngage position (calculated after an E-Stop recovery routine) in the BlendData structure	FALSE
VAR_OUTPUT				
E	InSync	BOOL	Set high when the slave first synchronizes with the master (Running cam profile is synchronized). 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	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 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.

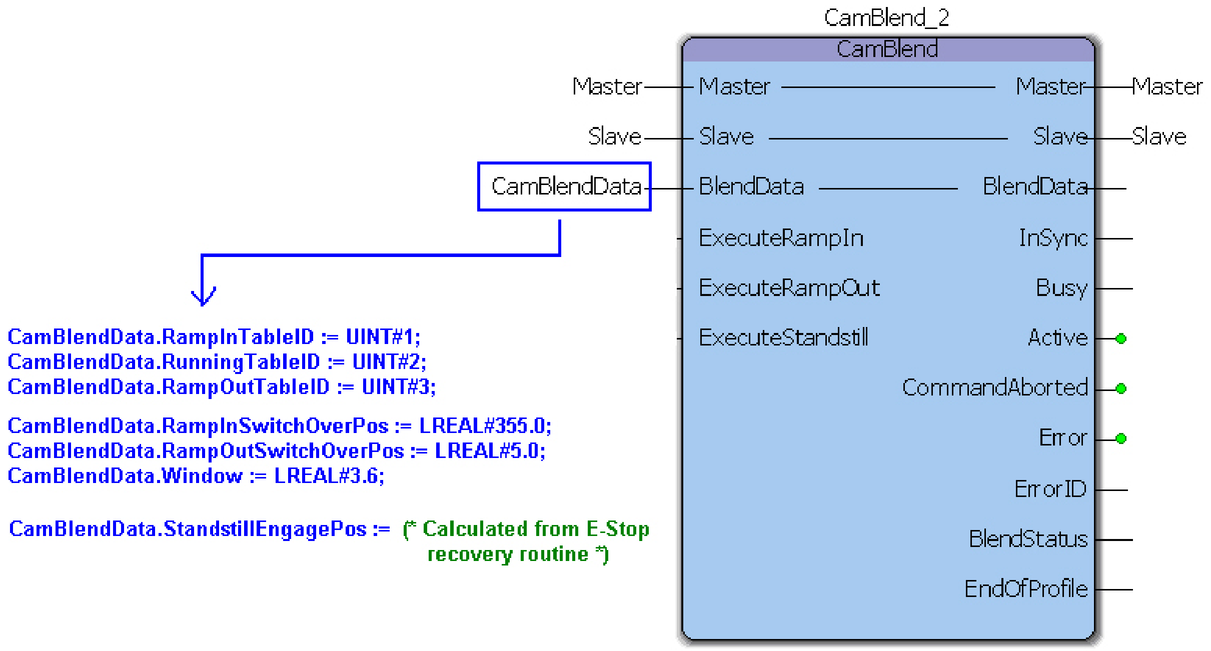
- If using the ExecuteStandStill mode, use the CamMaster_Lookup and CamSlave_Recover function blocks to determine the master position that corresponds to the current slave position, and set BlendData.StandStillEngagePos accordingly to preserve synchronization. The ExecuteStandStill mode was added to provide the capability of re-synchronizing after an E-Stop.

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

Error Description

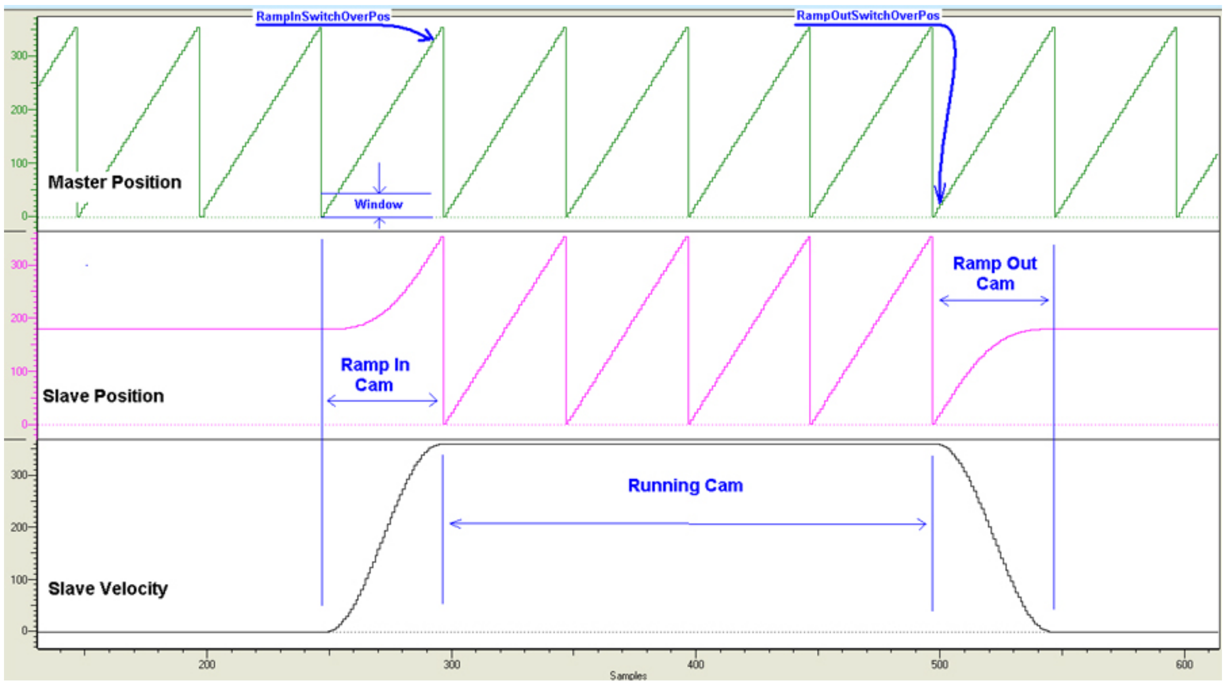
ErrorID	Meaning
0	No error
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4375	CamOut called while not 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 cams Machine Cycle. (0 to Prm1502, the last master position in the active cam table.)
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.
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. 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

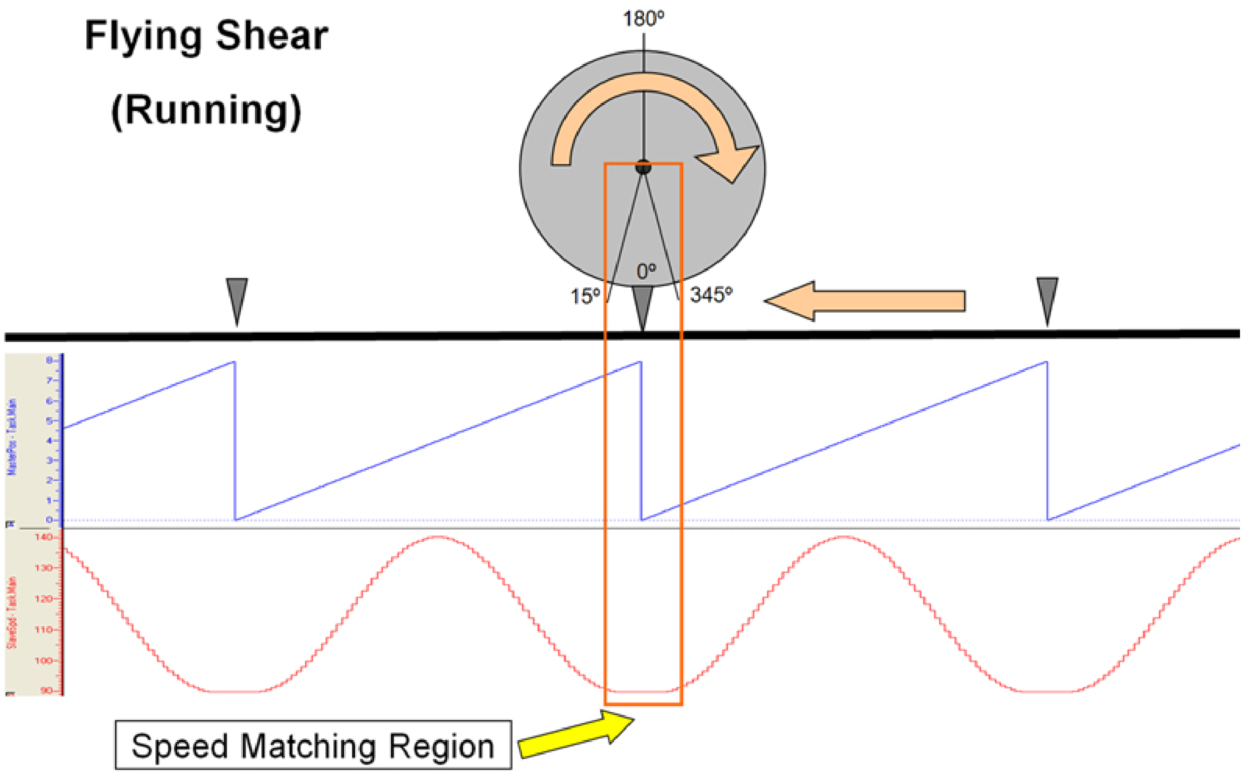


```
CamBlendData.RampInTableID := UINT#1;
CamBlendData.RunningTableID := UINT#2;
CamBlendData.RampOutTableID := UINT#3;

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

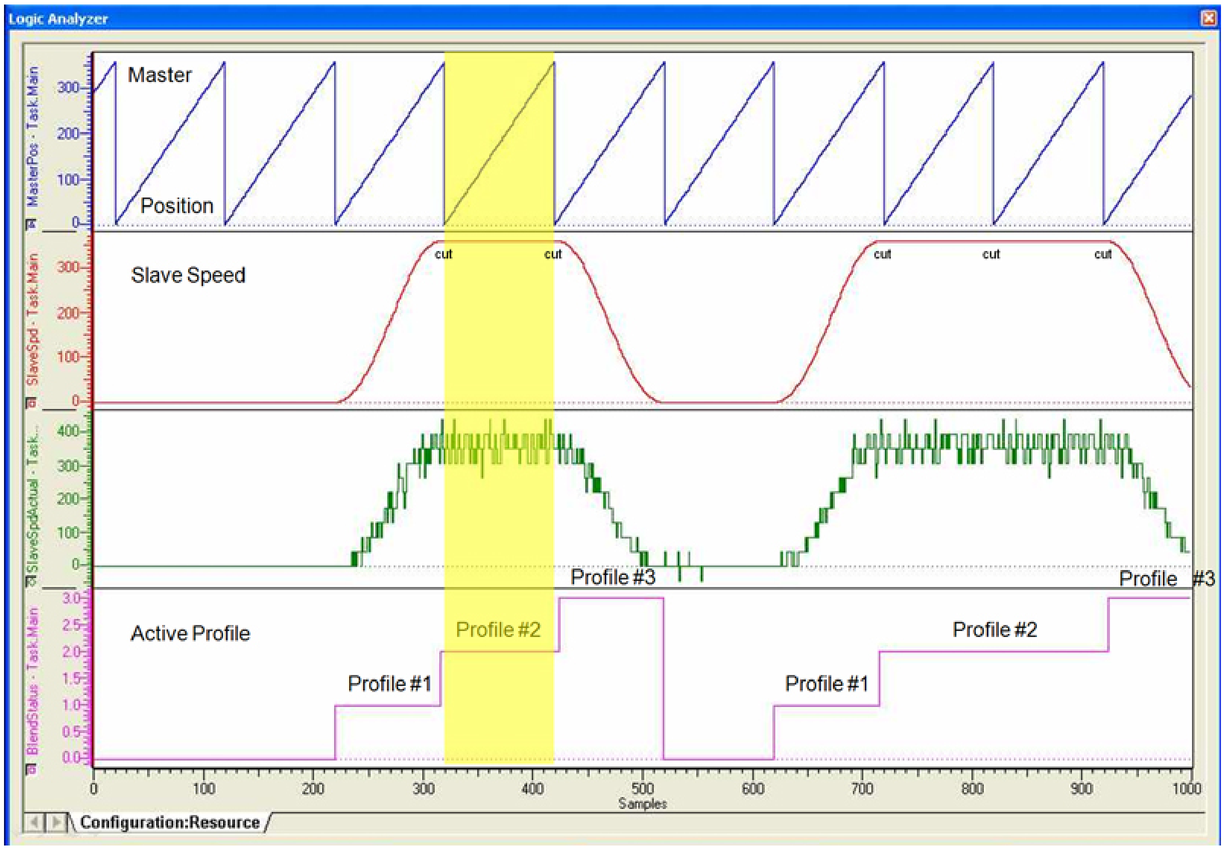


Application Example

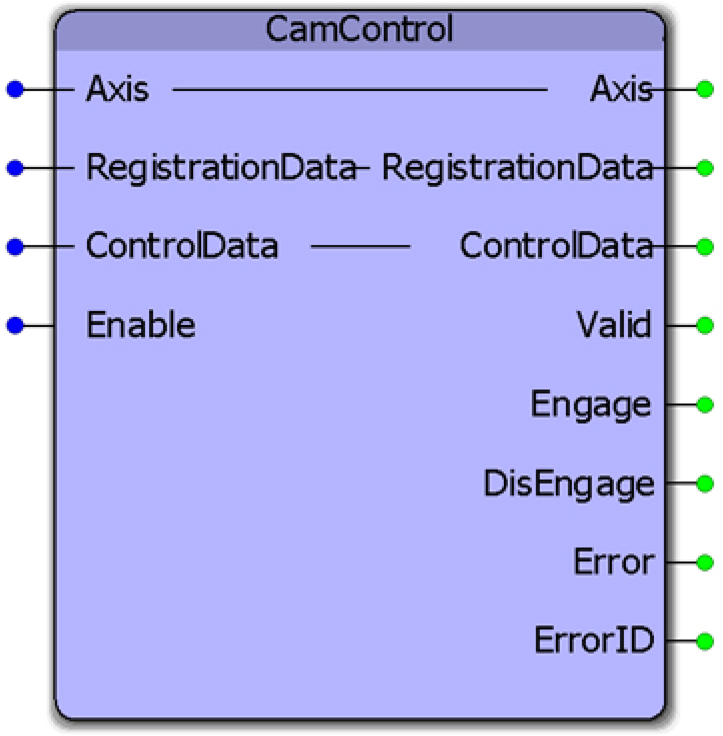


Timing Diagram

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.


Logic Analyzer
Master
Position
Slave Speed
cut
cut
cut
cut
cut
Profile #3
Profile #3
Active Profile
Profile #2
Profile #2
Profile #1
Profile #1
MasterPos - Task.Main
SlaveSpd - Task.Main
SlaveSpdActual - Task...
BlendStatus - Task.Main
Samples
Configuration:Resource

CamControl



The CamControl block makes decisions regarding engage and disengage timing 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 like Linear Flying Shear, Random Rotary Placer, Knife, Drill, etc.

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	FALSE

			while enable is held high.	
VAR_OUTPUT				
B	Valid	BOOL	Indicates that 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 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.

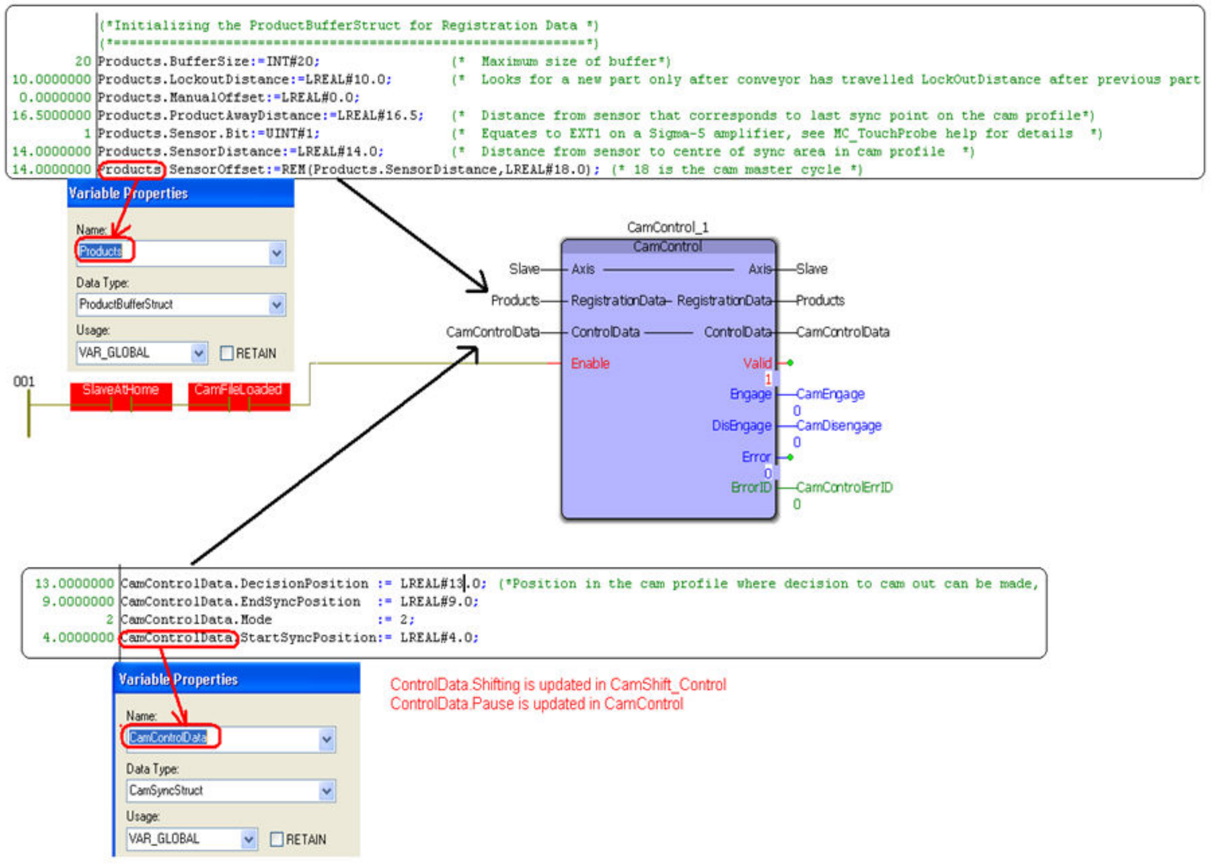
• This function block is designed to work with the CamShift_Control function block. It waits for an initial Camshift will occur before the first Engage event should take place. If the application requires the slave to become synchronized with the master without a Camshift, simply use an R_TRIG of the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

Error Description

ErrorID	Meaning
0	No error
10081	ControlData.DecisionPosition

Code 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.



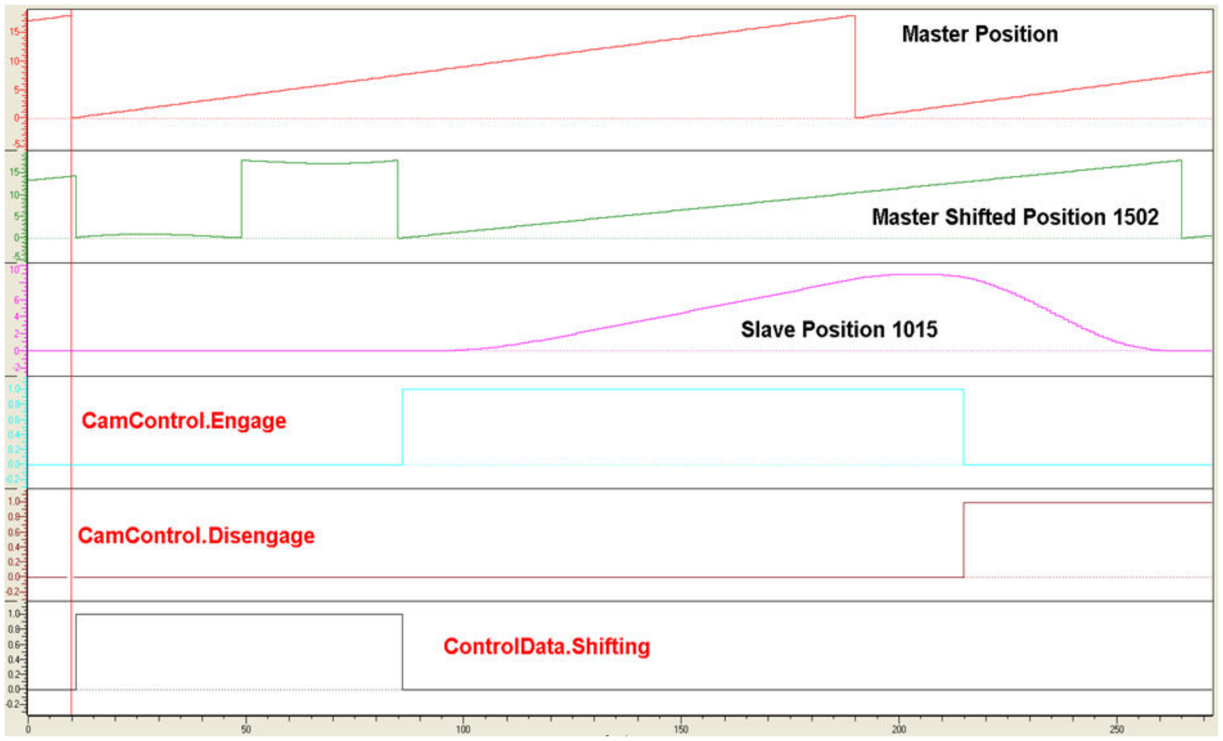
```
          (*Initializing the ProductBufferStruct for Registration Data *)
          (*===========================================================*)
       20 Products.BufferSize:=INT#20;                (*  Maximum size of buffer*)
10.0000000 Products.LockoutDistance:=LREAL#10.0;      (*  Looks for 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:

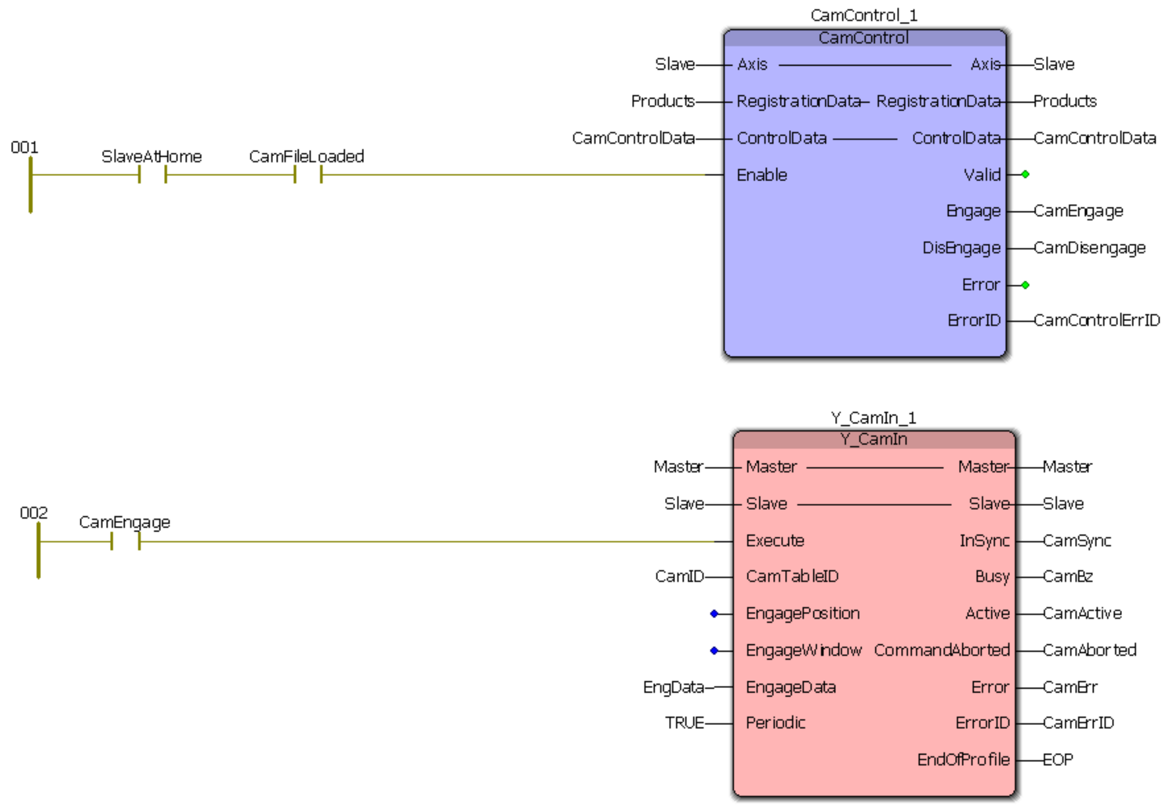
1. The ProductBuffer stores the position of each defect where a cut must be made.

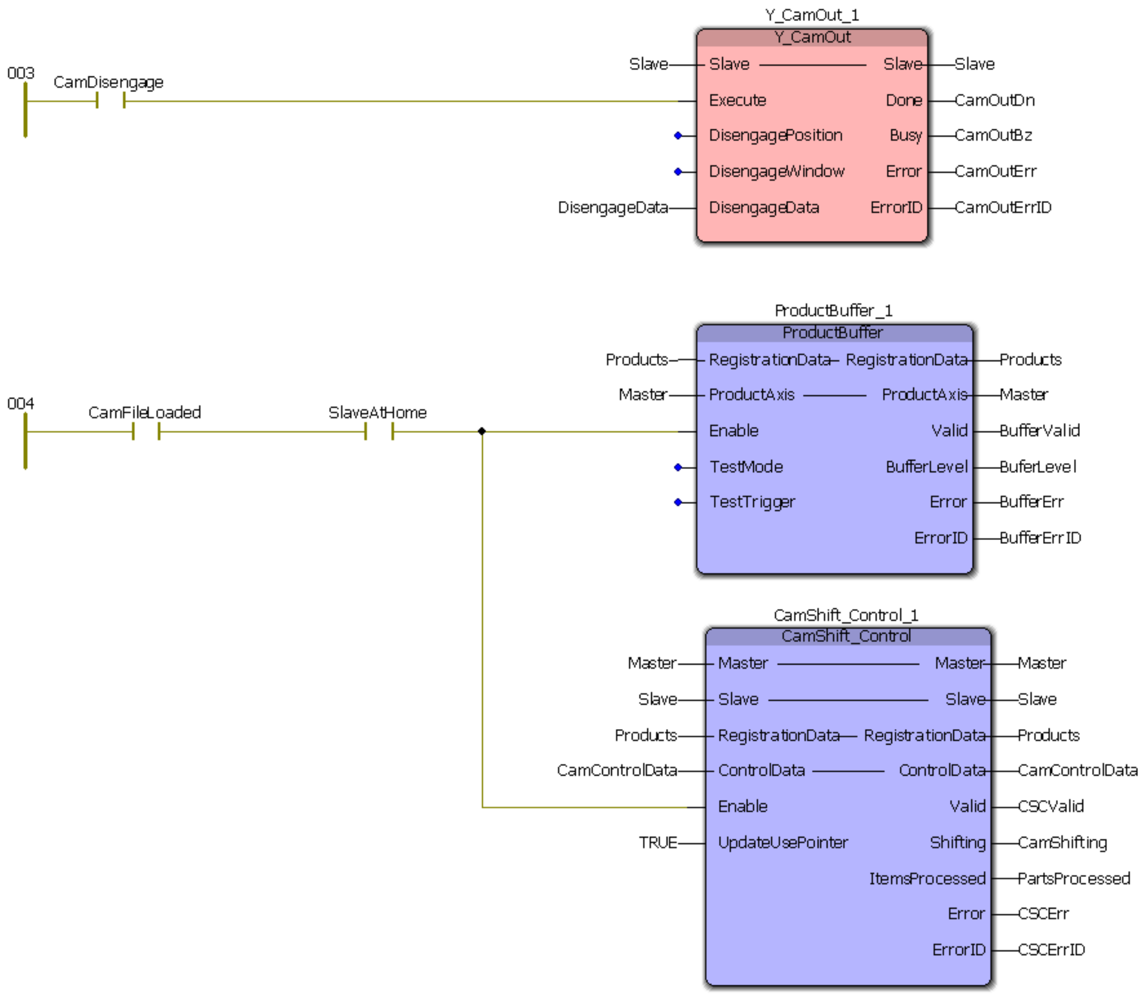
2. The CamShift_Control synchronizes the master (conveyor moving the wood) and slave (saw).

3. The CamControl.Engage output must be connected to Y_CamIn.Execute. (Other logic requirements may be included if necessary.)

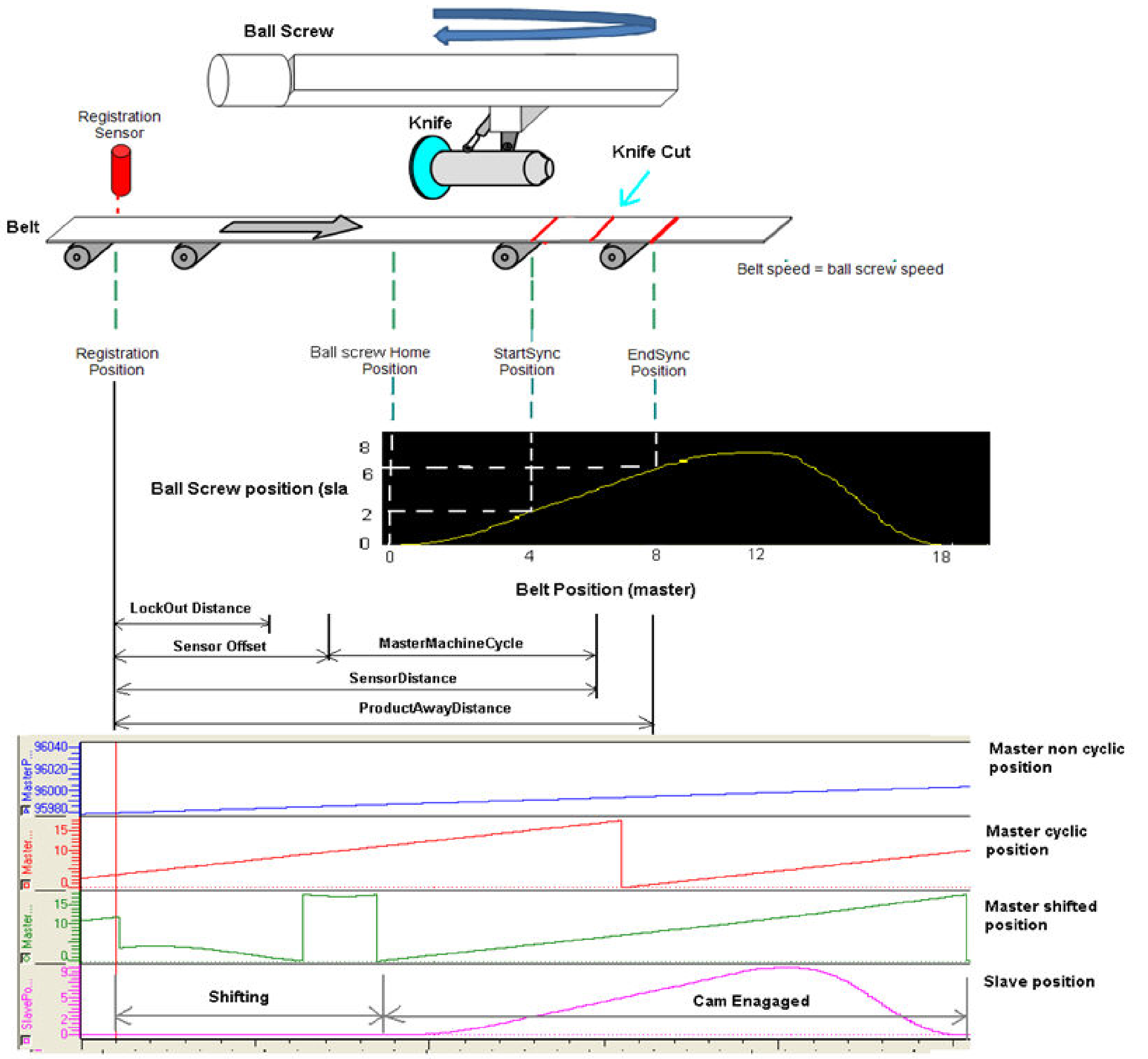
4. Key Point: When defects are close together, the goal is to remain engaged, and use the CamShift function during the slave (saw) retraction stroke while not in contact with the wood to re-synchronize with the next defect (or knot) to be cut.

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.

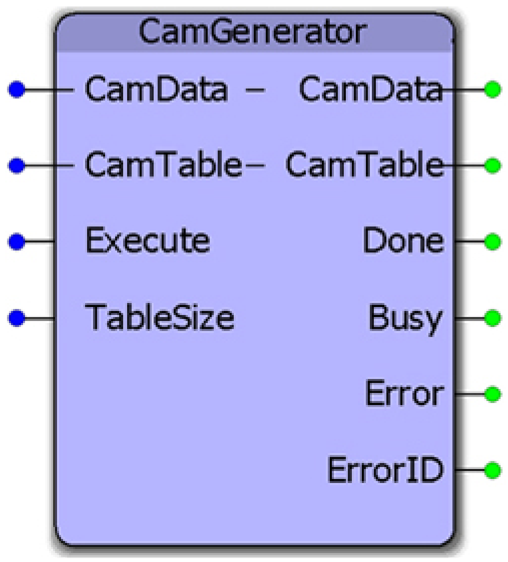




003
CamDisengage
Y_CamOut_1
Y_CamOut
Slave
Execute
DisengagePosition
DisengageWindow
DisengageData
Slave
Done
Busy
Error
ErrorID
Slave
CamOutDn
CamOutBz
CamOutErr
CamOutErrID
Slave
DisengageData
004
CamFileLoaded
SlaveAtHome
ProductBuffer_1
ProductBuffer
Products
Master
RegistrationData
ProductAxis
Enable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


Ball Screw
Registration Sensor
Knife
Knife Cut
Belt
Belt speed = ball screw speed
Registration Position
Ball screw Home Position
StartSync Position
EndSync Position
Ball Screw position (sla
Belt Position (master)
LockOut Distance
Sensor Offset
MasterMachineCycle
SensorDistance
ProductAwayDistance
Master non cyclic position
Master cyclic position
Master shifted position
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 datapoints required by the application, including a formula code which is used to generate a pair of master / slave datapoints 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.

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	ERROR: Variable (Parameter bDescription_Done) is undefined.
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
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 PLCopen Toolbox DataType definition. If you change 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

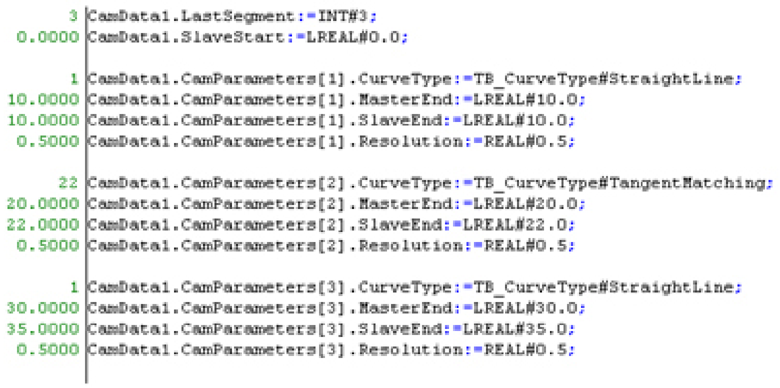
ErrorID	Meaning
0	No error
10038	CamData.LastSegment must be greater than 0 and less than 400, or whatever value has been declared as the ARRAY size in the CTB_Types file.

[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

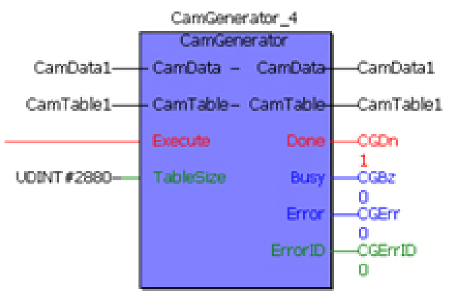
Examples

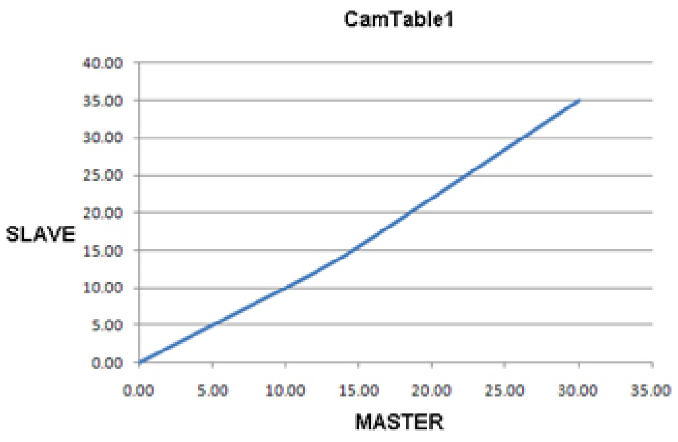
Structured text to load a CamSegmentStruct:

Example 1



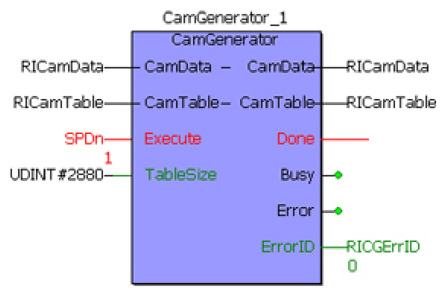
```
      3 CamData1.LastSegment:=INT#3;
 0.0000 CamData1.SlaveStart:=LREAL#0.0;

      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;
```

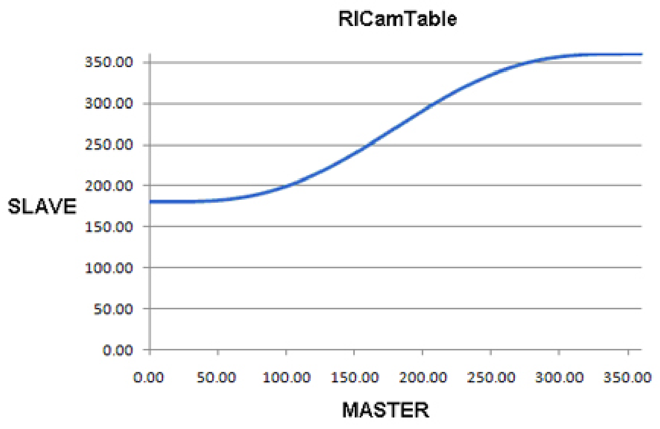




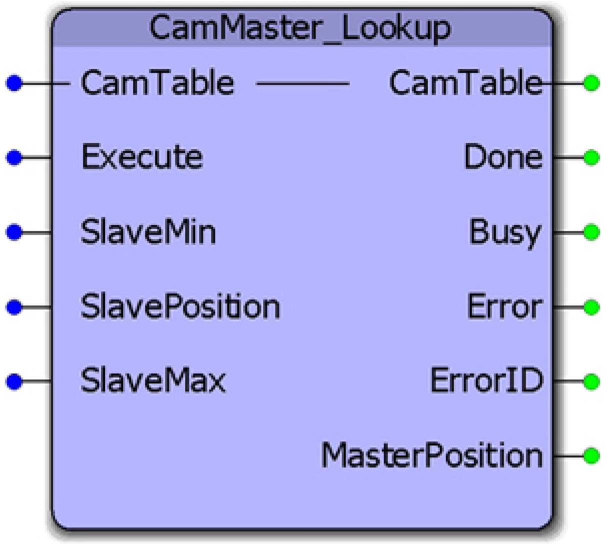
Example 2

```
        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.

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	CamTable	Y_MS_CAM_STRUCT	Cam data structure	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	SlaveMin	LREAL	The smallest slave position to include when searching for the master.	LREAL#0.0
V	SlavePosition	LREAL	The current slave position	LREAL#0.0
B	SlaveMax	LREAL	The largest slave position to include when searching for the master.	LREAL#0.0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has been completed successfully. If another block takes control before the action is completed, the Done output will	

			not be set. This output is reset when execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
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:

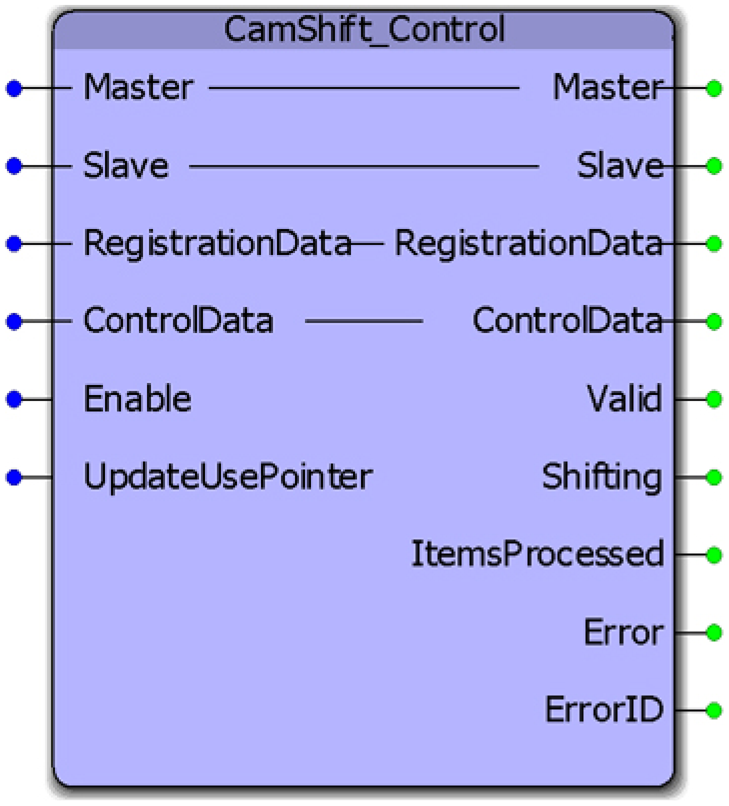
M	S
0	0
10	0
20	5
30	10
40	20

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.

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 while enable is held high.	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
VAR_OUTPUT				
B	Valid	BOOL	Indicates that 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 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.
- This 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 by. 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.

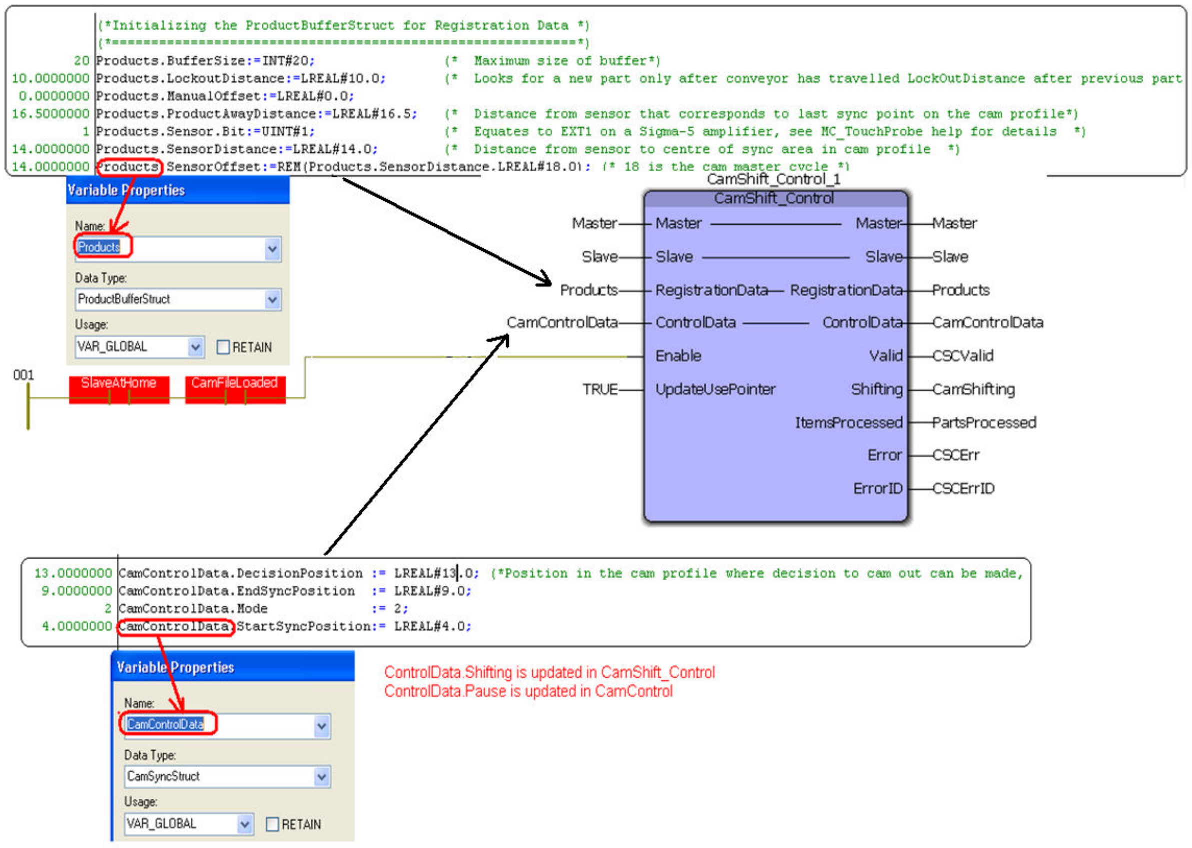
Error Description

ErrorID	Meaning
0	No error

7282	Cam Shift Aborted. Verify that the CamShift_Control function block was not interrupted by another function block that resulted in Y_CamShift.CommandAborted
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.



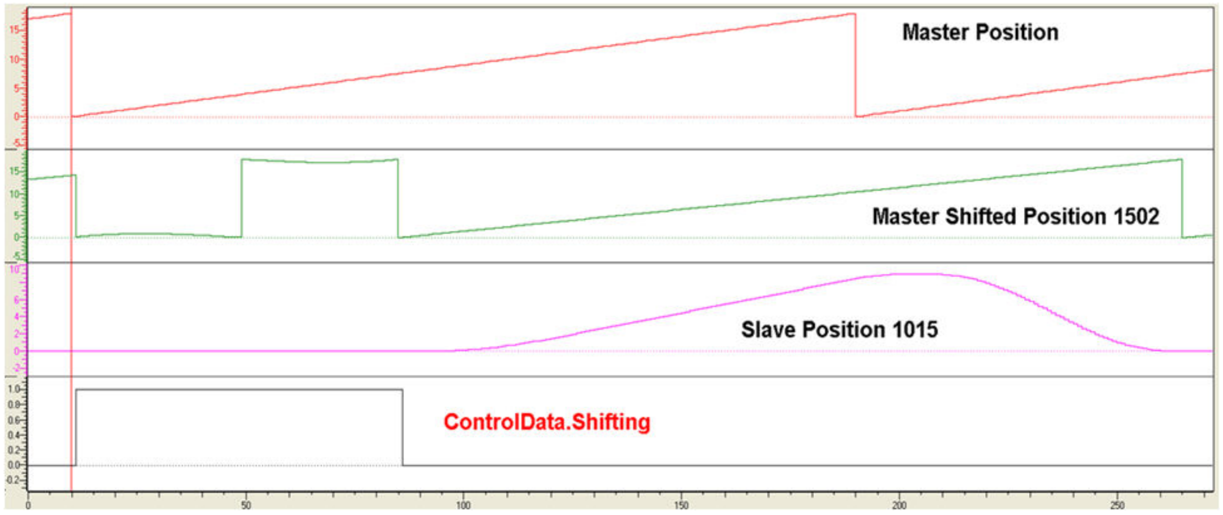
```
          (*Initializing the ProductBufferStruct for Registration Data *)
          (*============================================================*)
       20 Products.BufferSize:=INT#20;                 (*  Maximum size of buffer*)
10.0000000 Products.LockoutDistance:=LREAL#10.0;       (*  Looks 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

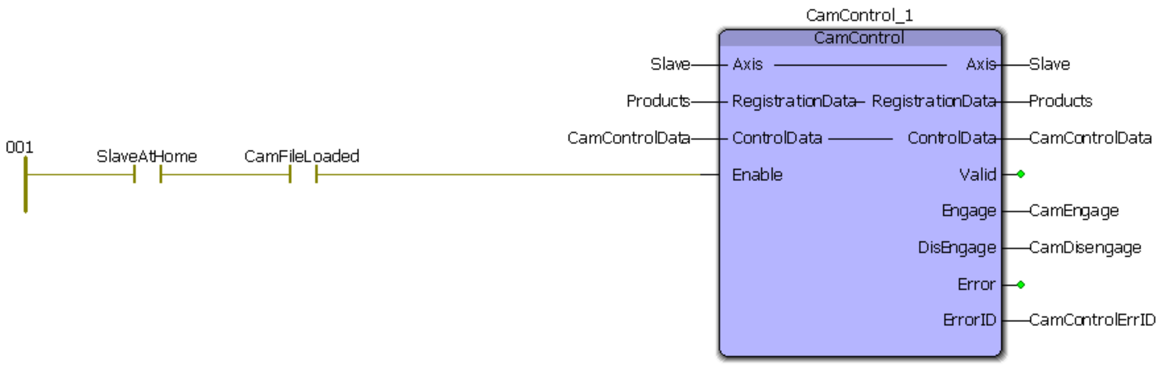
ControlData.Shifting is updated in CamShift_Control
ControlData.Pause is updated in CamControl

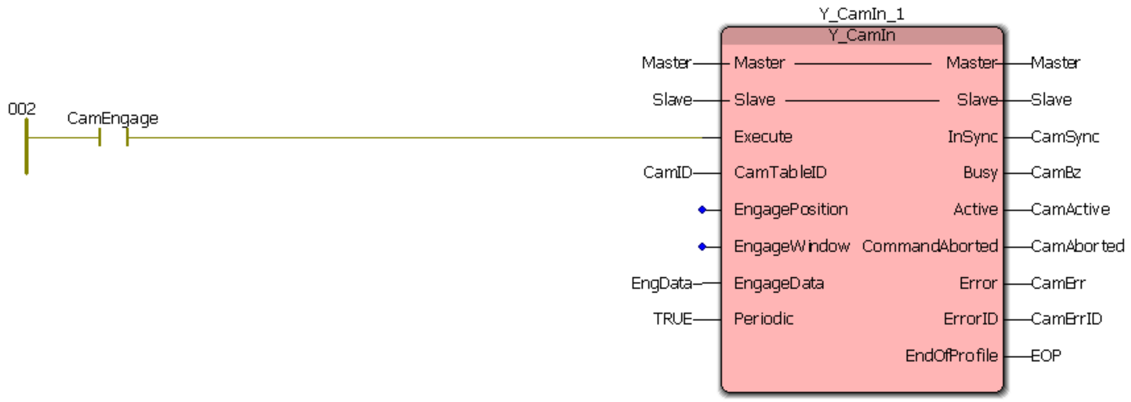


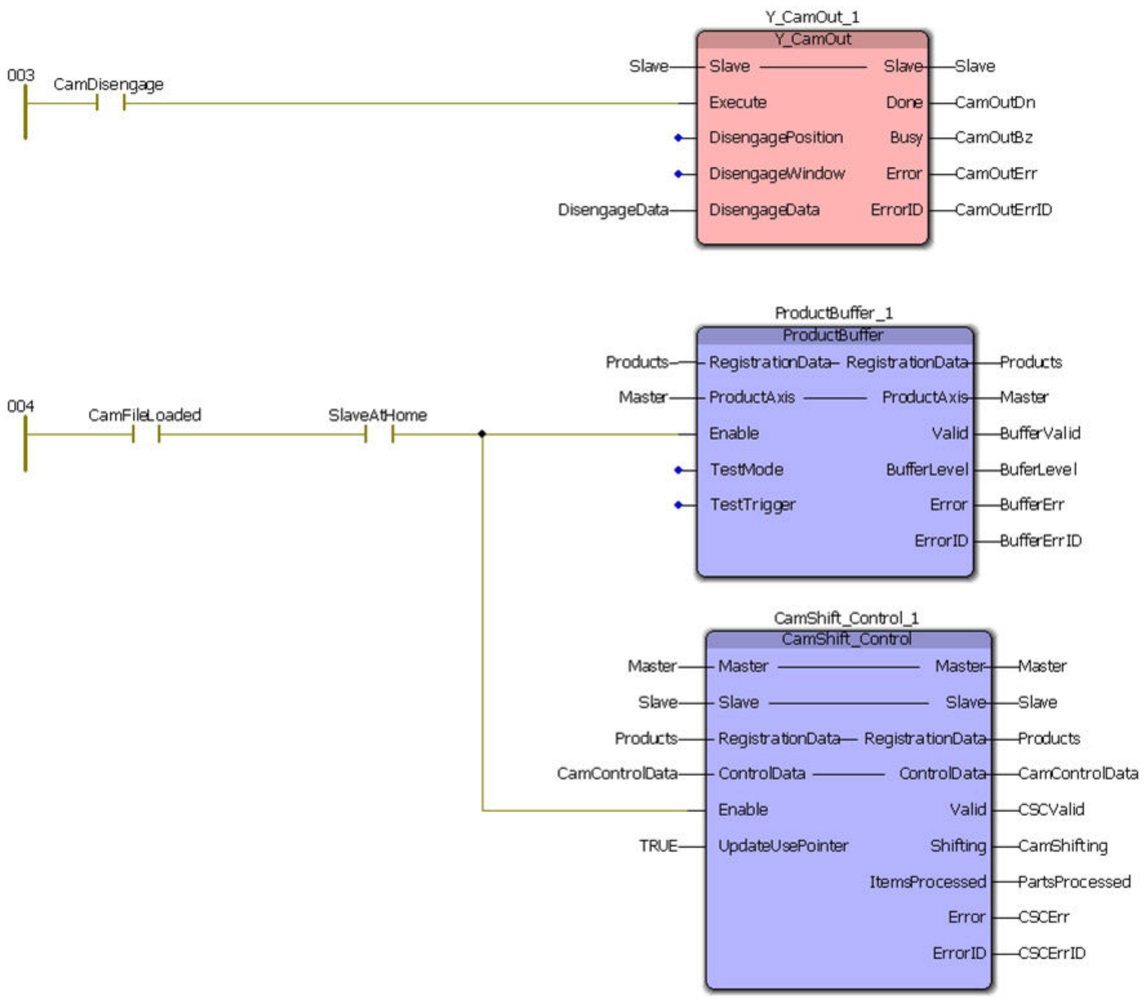
Application Example

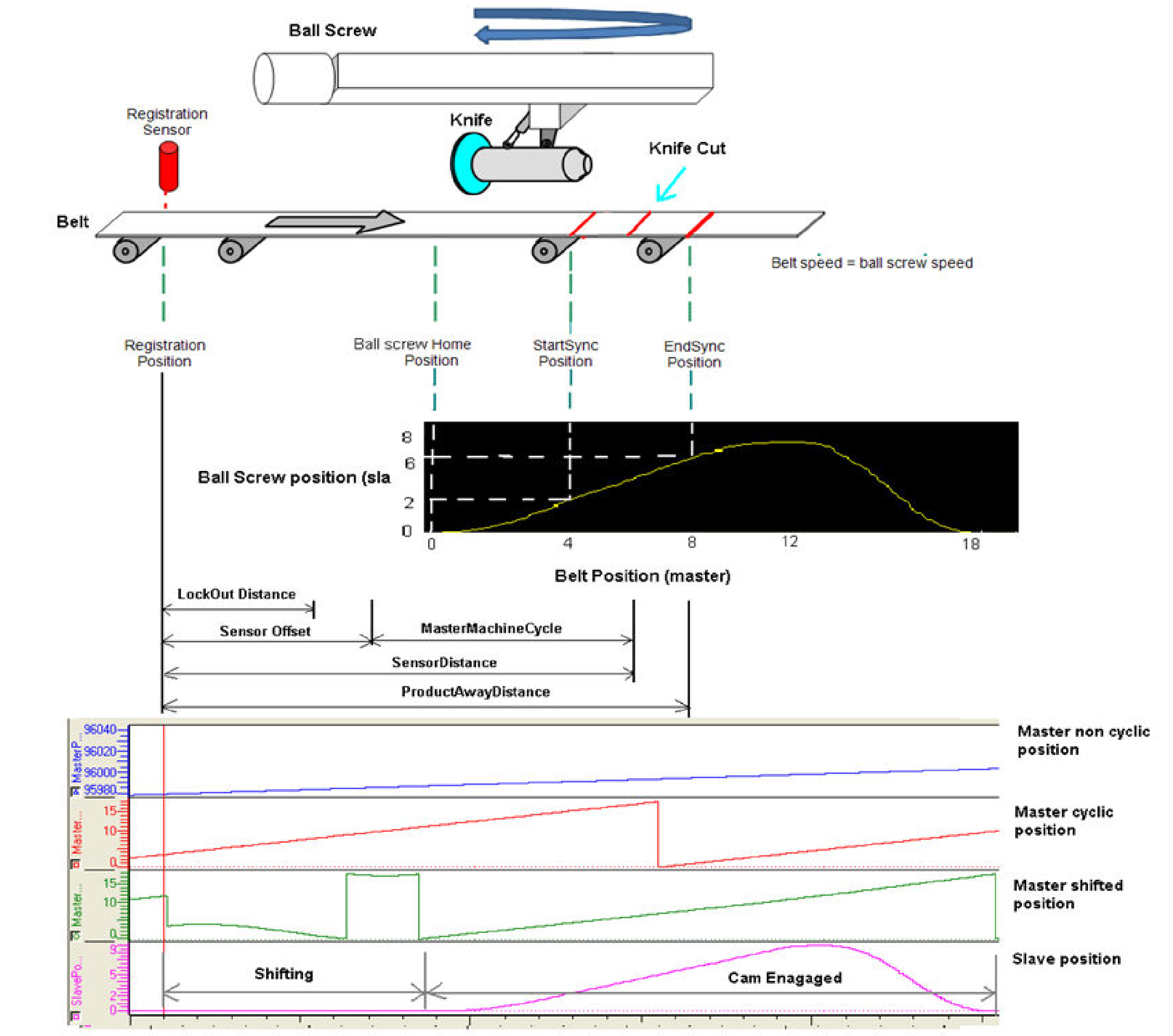
This example illustrates how the CamControl block can be applied in a linear flying shear application. In this application, the items to be cut are defective areas (knots) in a piece of wood. The code shown here performs the following actions:

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.


CamControl_1
CamControl
001
SlaveAtHome
CamFileLoaded
Slave
Products
CamControlData
Axis
RegistrationData
ControlData
Enable
Axis
RegistrationData
ControlData
Valid
Engage
DisEngage
Error
ErrorID
Slave
Products
CamControlData
CamEngage
CamDisengage
CamControlErrID

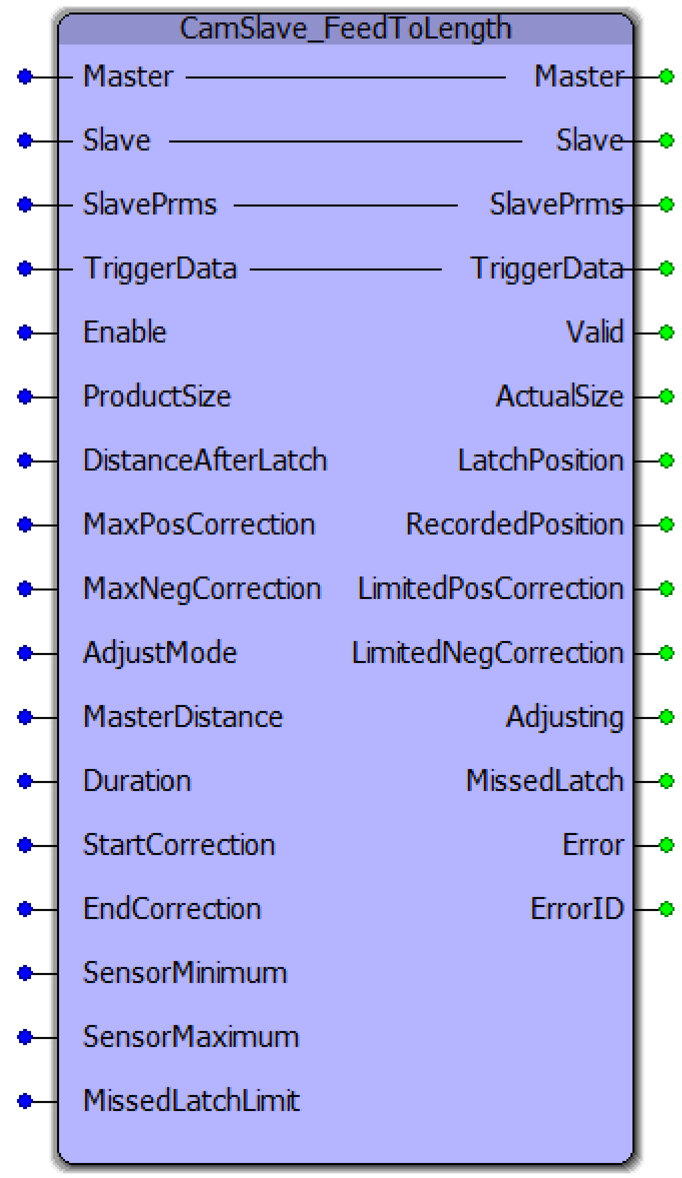

Y_CamIn_1
Y_CamIn
002
CamEngage
Master
Slave
CamID
EngData
TRUE
Master
Slave
Execute
CamTableID
EngagePosition
EngageWindow
EngageData
Periodic
Master
Slave
InSync
Busy
Active
CommandAborted
Error
ErrorID
EndOfProfile
Master
Slave
CamSync
CamBz
CamActive
CamAborted
CamErr
CamErrID
EOP


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
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
15
10
0
15
10
0
9
5
2
0
Shifting
Cam Enagaged
Master non cyclic position
Master cyclic position
Master shifted position
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.

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 while enable is held high.	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 product lengths that can occur without seeing a mark in the window. Valid sensor detections will reset the internal counter. The next valid sensor detection will reset the internal counter.	UINT#0 (interpreted as infinite)
VAR_OUTPUT				
B	Valid	BOOL	Indicates that 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 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.
- 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

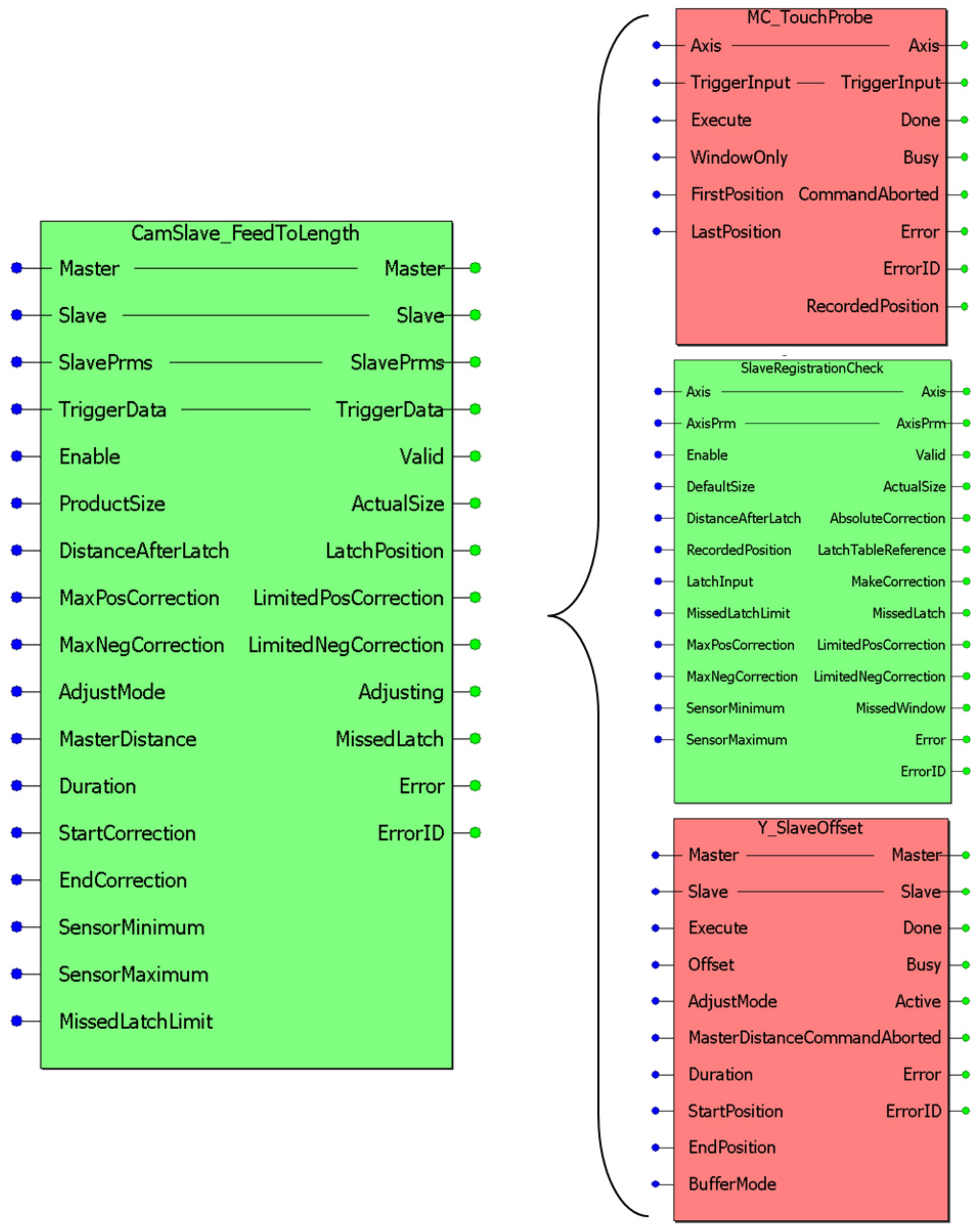
ErrorID	Meaning
0	No error
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4374	Torque move prohibited 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 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.
4626	The master slave relationship is defined. A slave cannot be a master to another axis.
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	Might be crossed or the same non-zero value
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

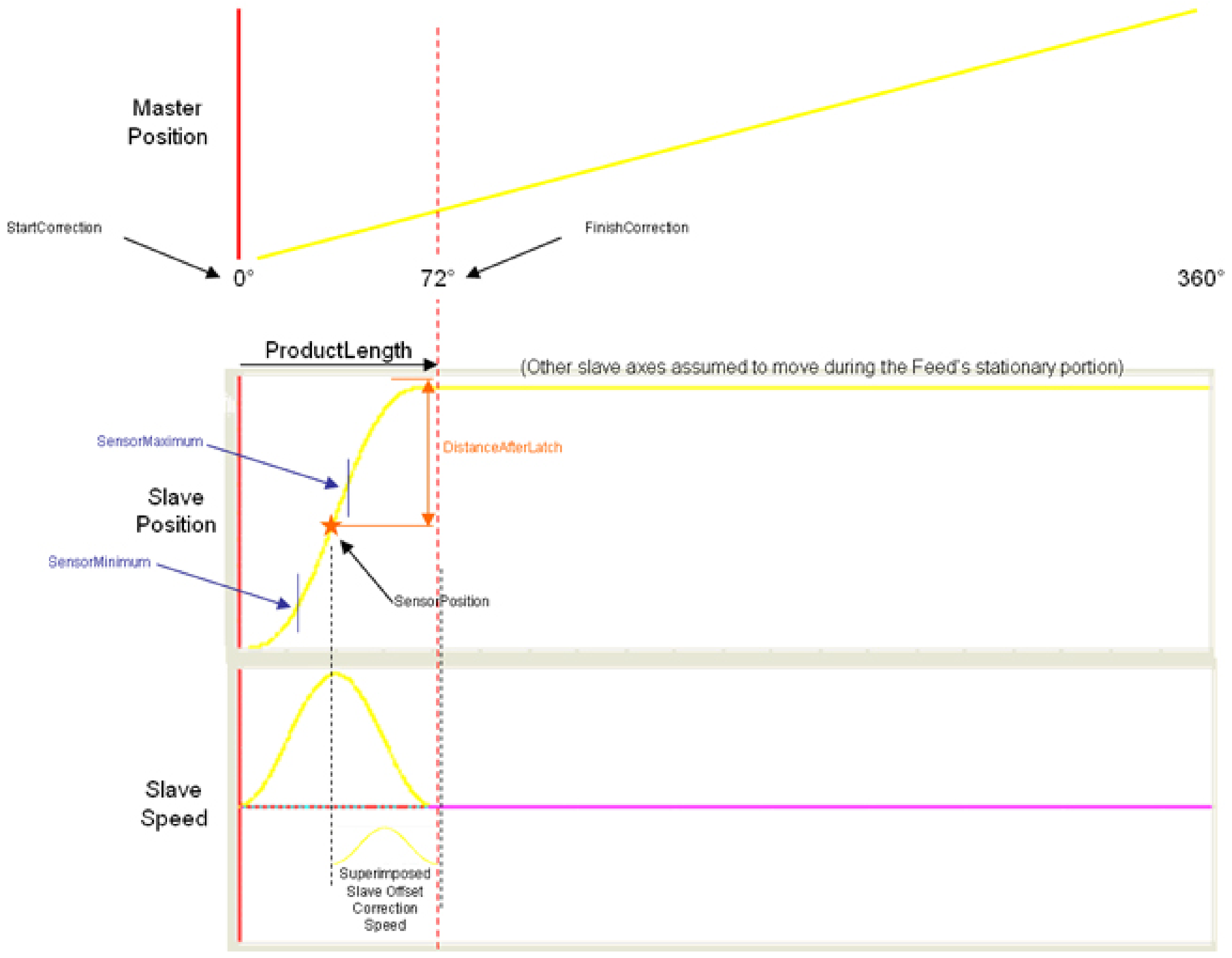
	match the required data size. Right click on the function block and select "Object Properties" to determine which parameters are ANY type.

Applications

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

Overview of Supporting Function Blocks

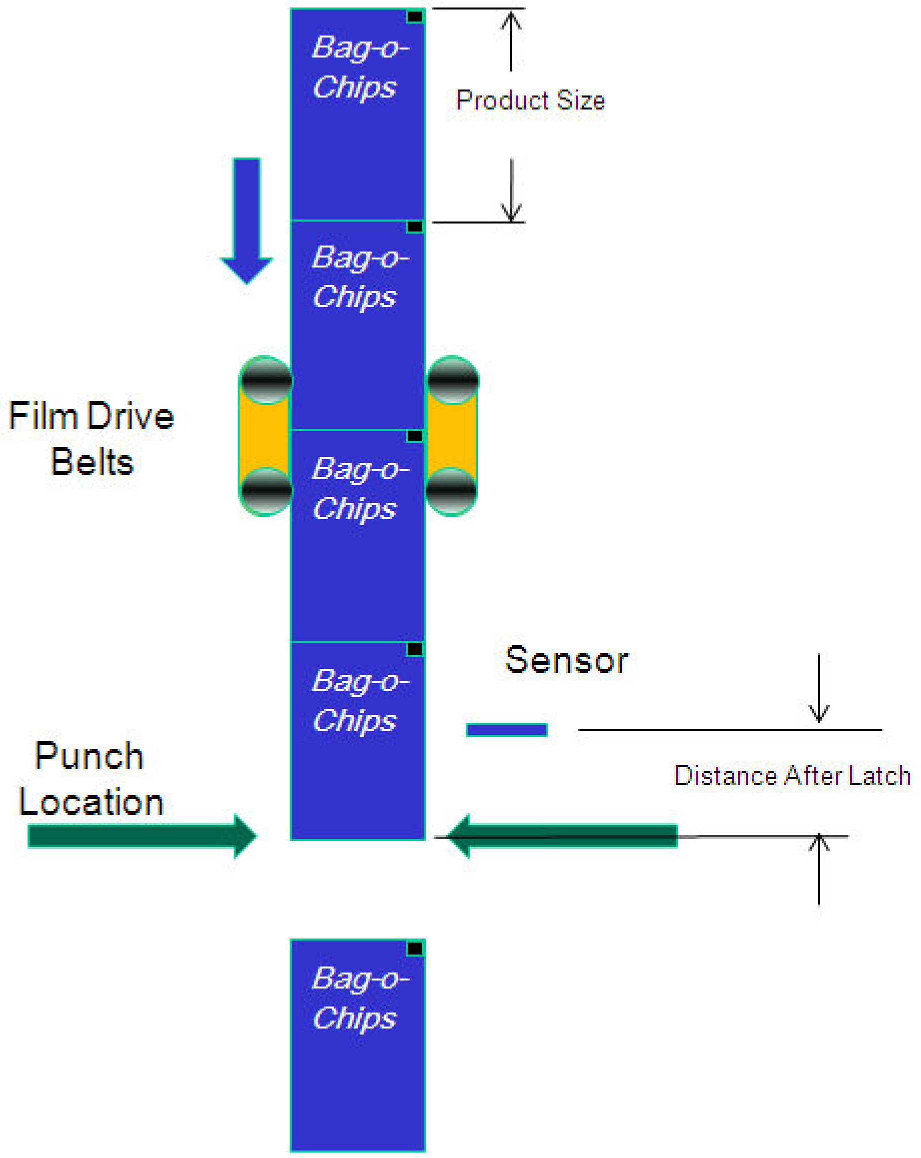

CamSlave_FeedToLength
Master
Master
Slave
Slave
SlavePrms
SlavePrms
TriggerData
TriggerData
Enable
Valid
ProductSize
ActualSize
DistanceAfterLatch
LatchPosition
MaxPosCorrection
LimitedPosCorrection
MaxNegCorrection
LimitedNegCorrection
AdjustMode
Adjusting
MasterDistance
MissedLatch
Duration
Error
StartCorrection
ErrorID
EndCorrection
SensorMinimum
SensorMaximum
MissedLatchLimit
MC_TouchProbe
Axis
Axis
TriggerInput
TriggerInput
Execute
Done
WindowOnly
Busy
FirstPosition
CommandAborted
LastPosition
Error
ErrorID
RecordedPosition
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
MasterDistance
CommandAborted
Duration
Error
StartPosition
ErrorID
EndPosition
BufferMode



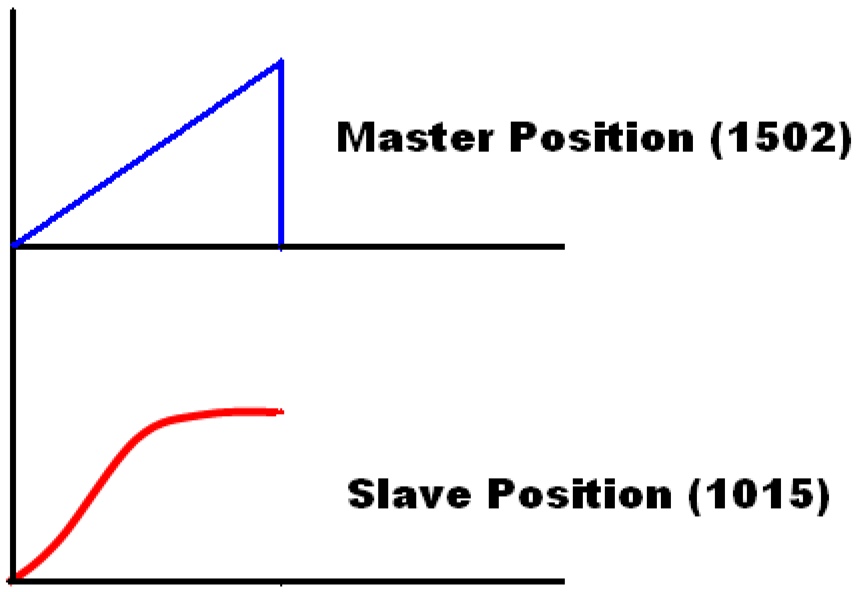
Application Example

Consider a form fill and seal application as shown below. Feed belts control payout of film for the form fill and seal machine.

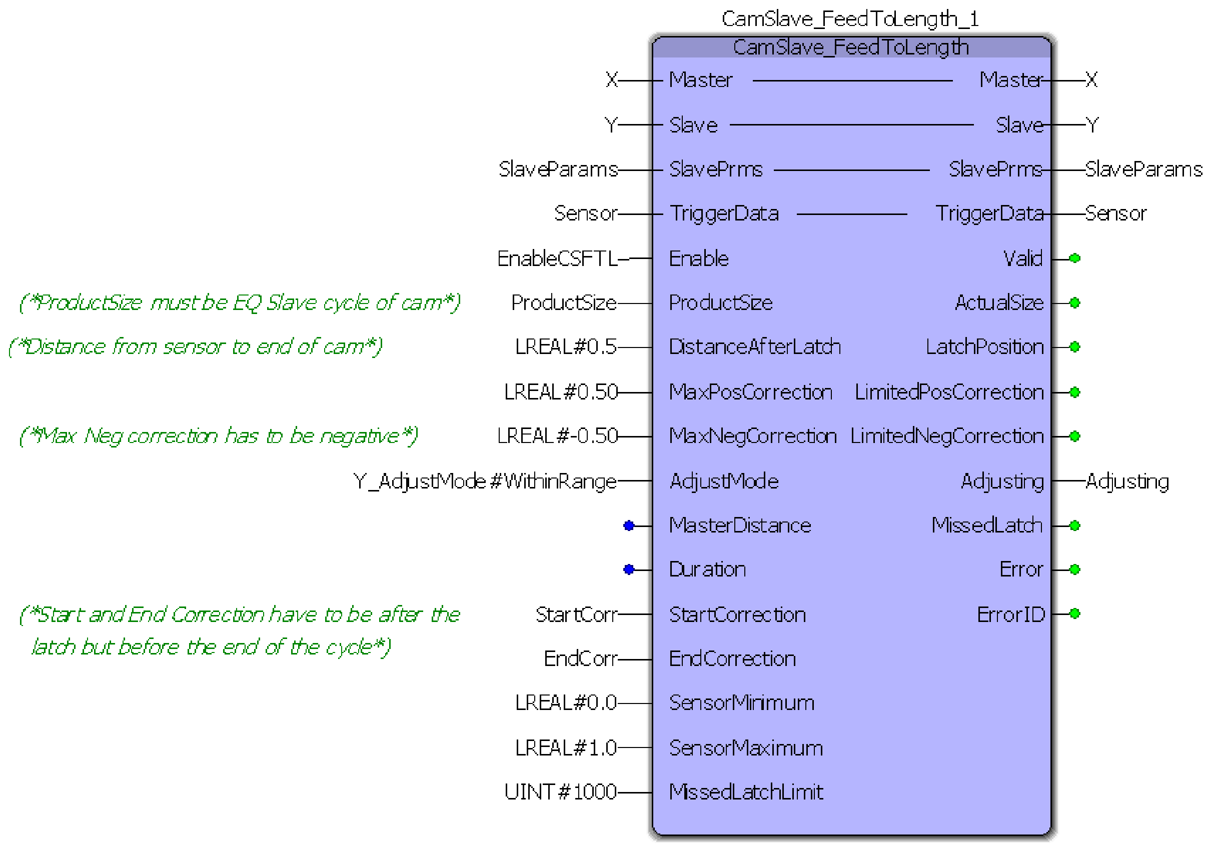
Distance After Latch is set to align the end of bag with the cutter/punch



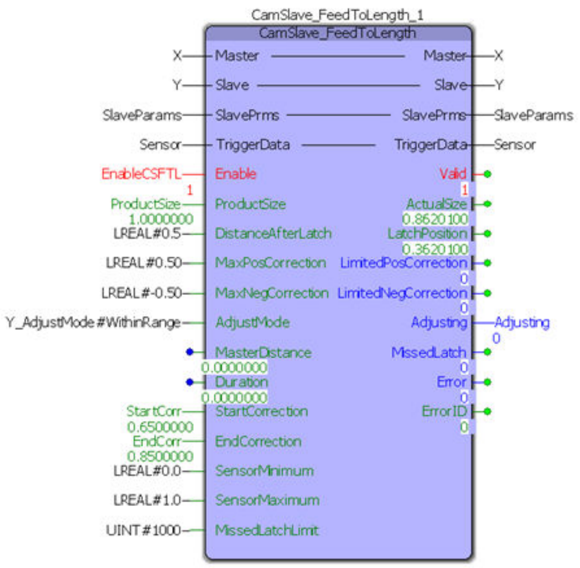
The film drive belt is the slave to a constantly running master. The nominal cam table is shown below. The master cycle is 0 - 1 units and the slave cycle is also between 0 and 1 units.

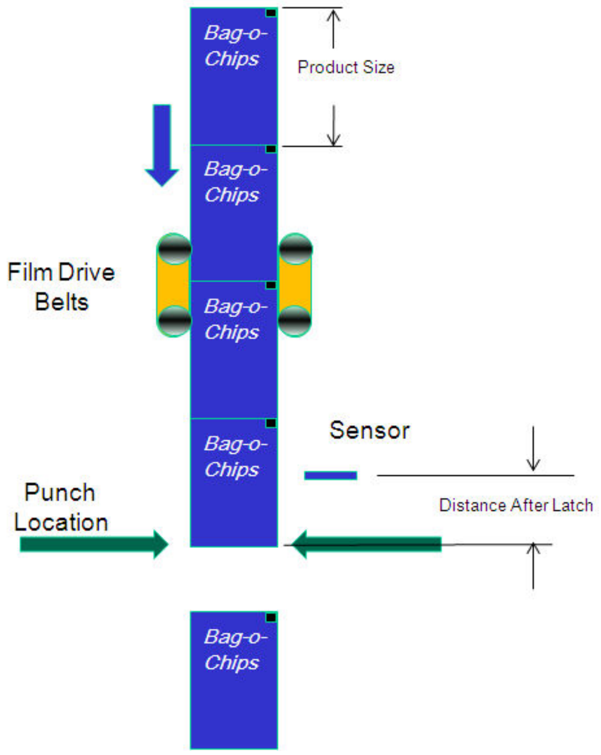


A sample screen shot of data that needs to be entered for the system described above is shown in the figure below. Care should be taken to ensure that the input parameters will generate motion that is physically achievable and desirable by the slave axis.


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



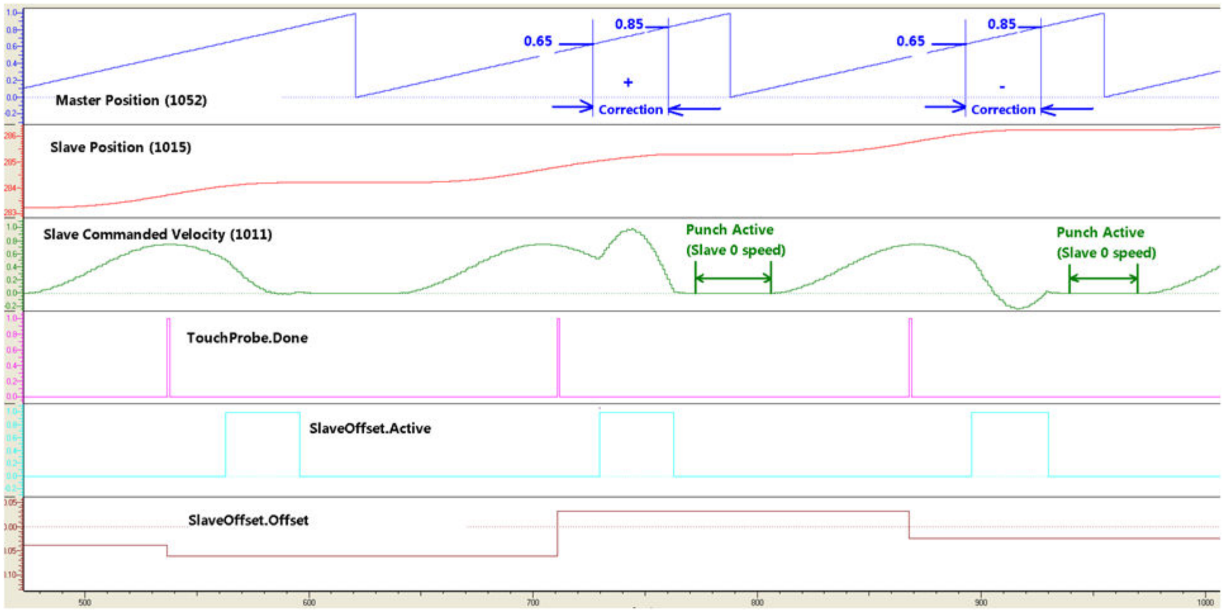
The calculation on how much adjustment needs to be made to make the slave axis (film feed) place the film exactly at the cutter/pinch location is explained below:

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

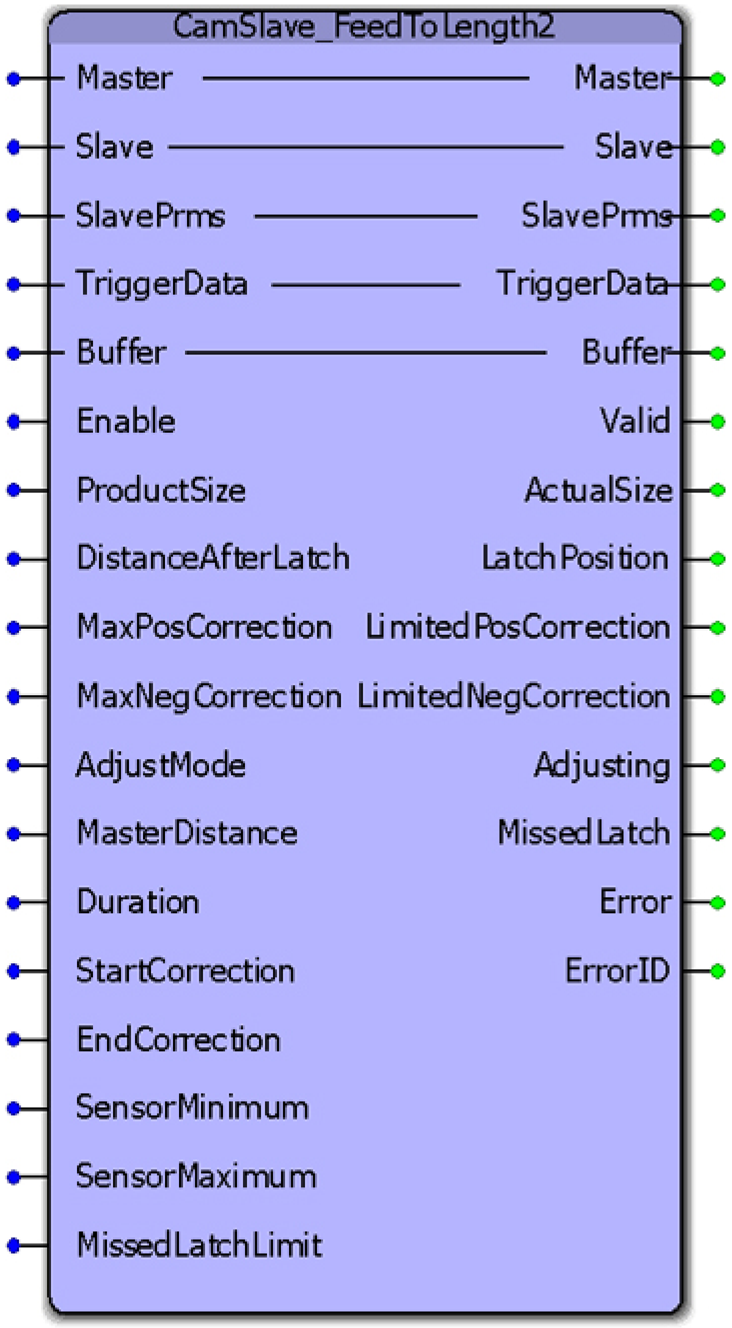
This will be the amount of offset added/subtracted (for this cycle) to any previous offsets in the slave position.

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.

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 while enable is held high.	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 product lengths that can occur without seeing a mark in the window. Valid sensor detections will reset the internal counter. The next valid sensor detection will reset the internal counter.	UINT#0 (interpreted as infinite)
VAR_OUTPUT				
B	Valid	BOOL	Indicates that 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	UDINT	Cumulative number of latches missed	
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.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

A Sigma-5 servo amplifier is required for use of this function block.

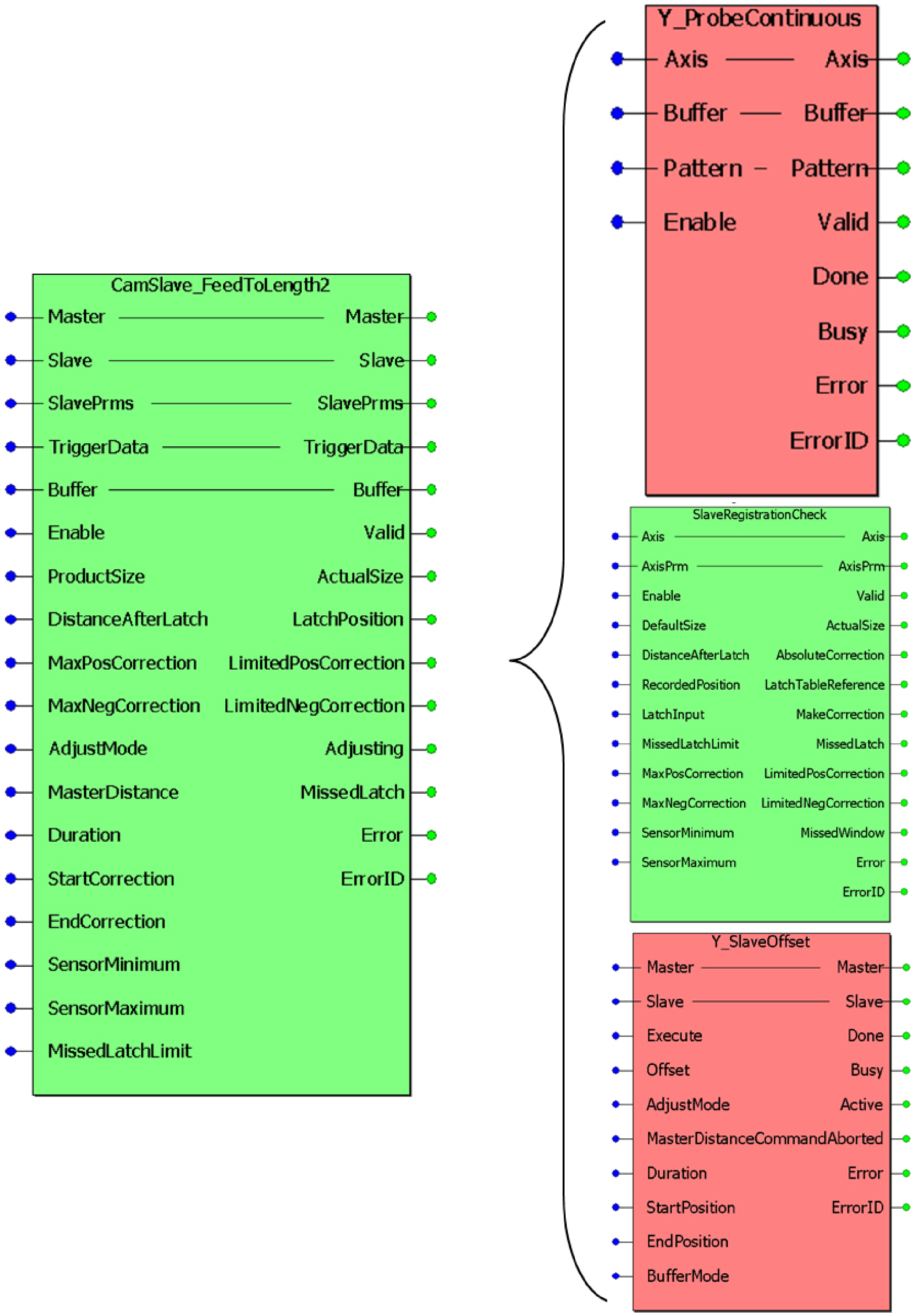
Error Description

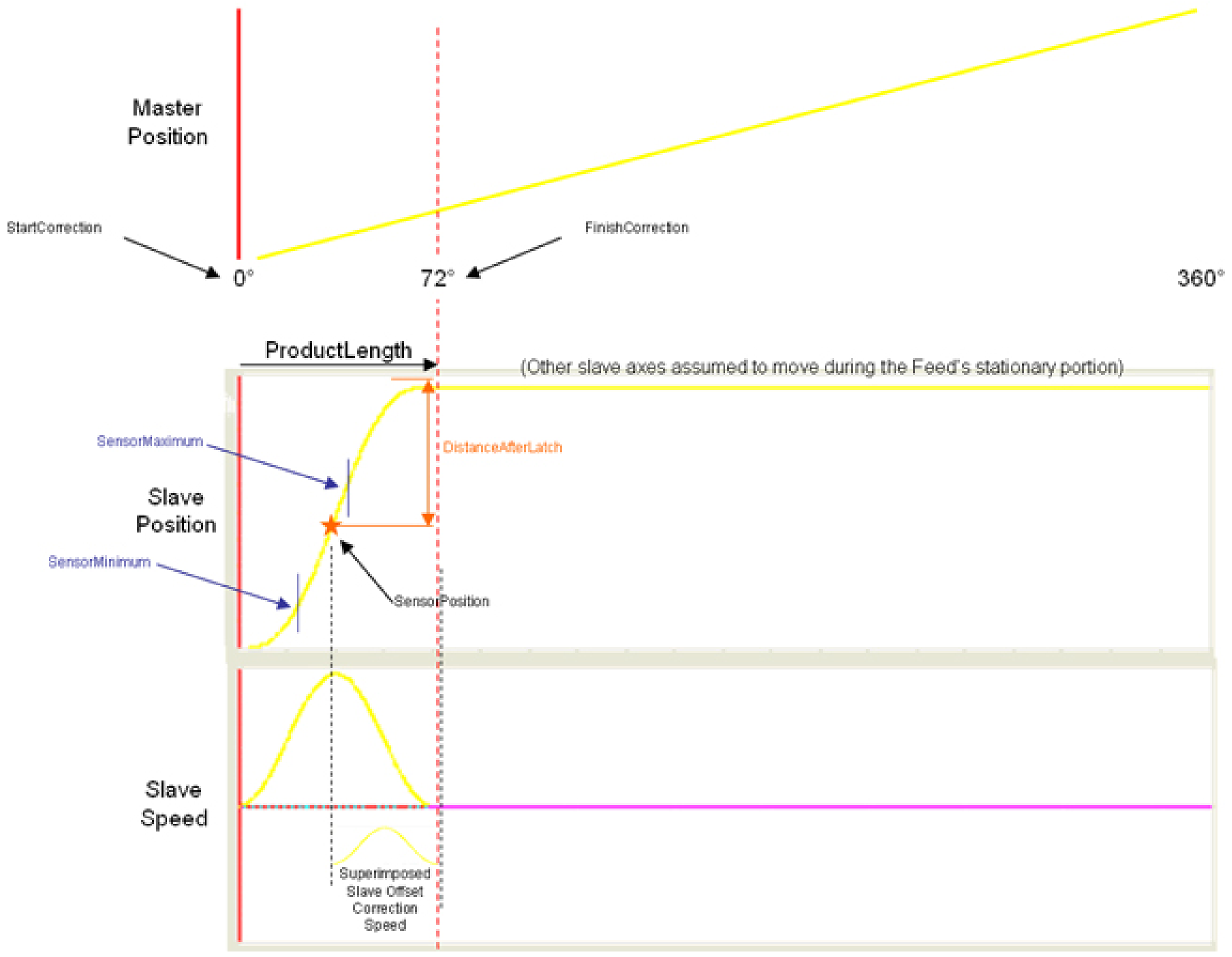
ErrorID	Meaning
0	No error
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4374	Torque move prohibited 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.
4406	Continuous Latch Mode 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)
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.
4626	The master slave relationship is defined. A slave cannot be a master to another axis.
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	Might be crossed or the same non-zero value
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.

Applications

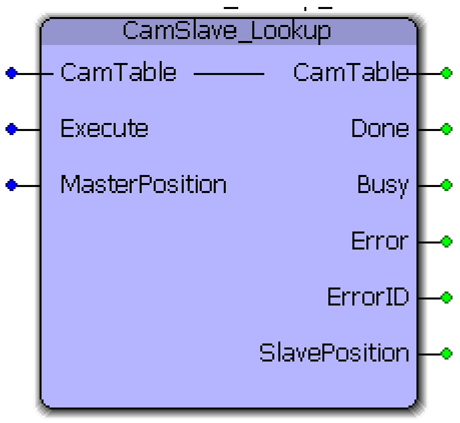
- Label Feeder
- Punch Press

Overview of Supporting Function Blocks


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


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.

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

			the function block. This output is cleared when 'Execute' or 'Enable' goes low.
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 provide 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.

This function determine 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.

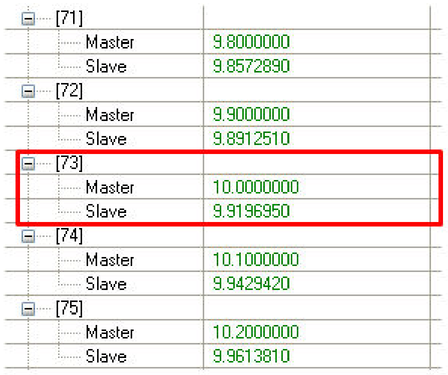
See the CamSlave_Lookup eLearning Module on Yaskawa's YouTube Channel.

Error Description

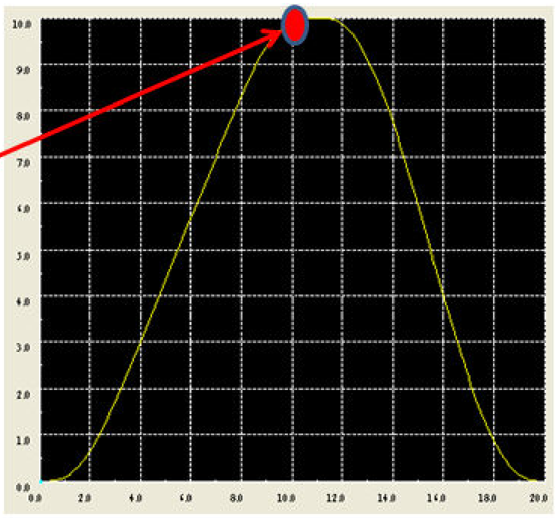
ErrorID	Meaning
0	No error
10114	Incorrect cam table size (check the CamTable.Header.Datasize)
10045	SlavePosition not found in Y_MS_CAM_STRUCT

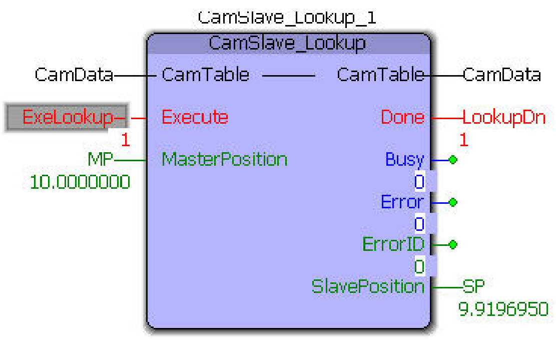
Example

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.

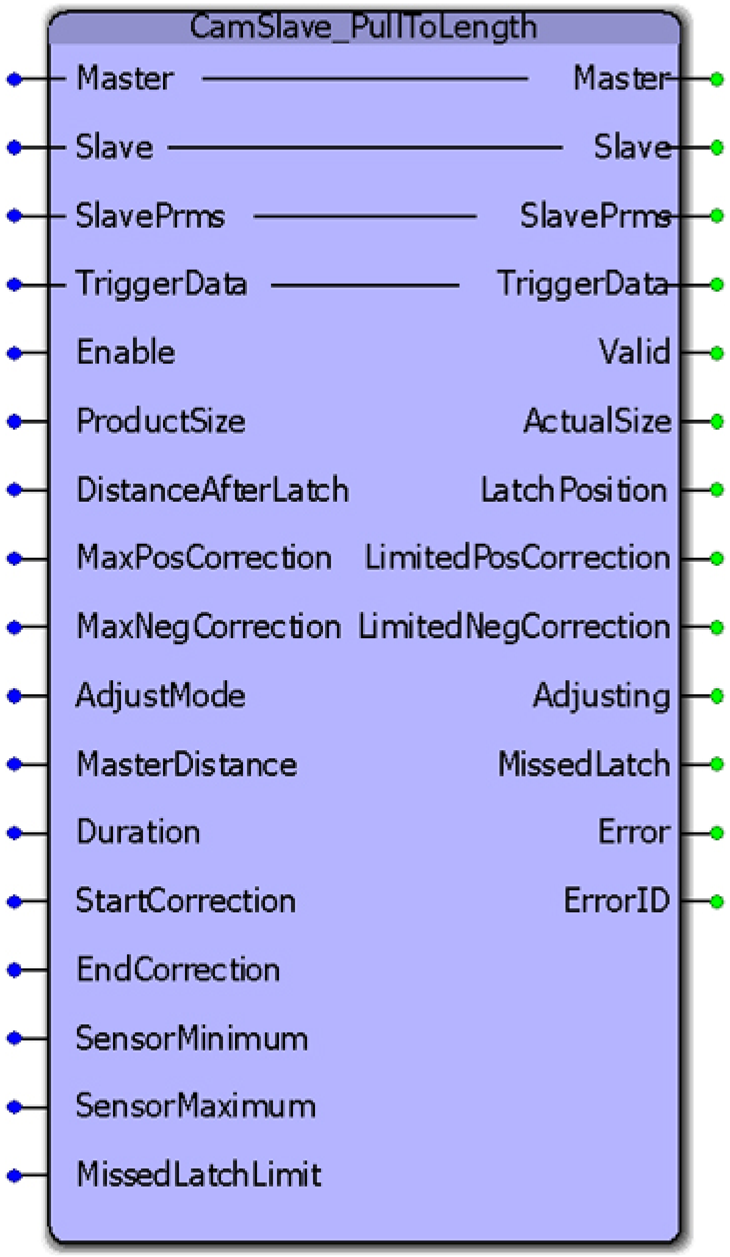


[71]	
Master	9.8000000
Slave	9.8572890
[72]	
Master	9.9000000
Slave	9.8912510
[73]	
Master	10.0000000
Slave	9.9196950
[74]	
Master	10.1000000
Slave	9.9429420
[75]	
Master	10.2000000
Slave	9.9613810





CamSlave_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.

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 while enable is held high.	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 product lengths that can occur without seeing a mark in the window. Valid sensor detections will reset the internal counter. The next valid sensor detection will reset the internal counter.	UINT#0 (interpreted as infinite)
VAR_OUTPUT				
B	Valid	BOOL	Indicates that 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	UDINT	Cumulative number of latches missed	
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.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

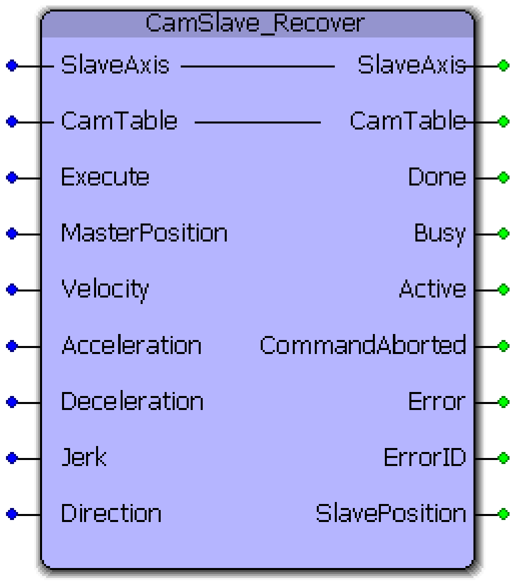
Notes

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	Might be crossed or the same non-zero value
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 a MC_MoveAbsolute function.

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	LREAL#0.0

			corresponding to this MasterPosition value.	
B	Velocity	LREAL	Velocity with which the slave axis recovers and moves to the position from the cam table corresponding to the master axis position	
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	
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	
B	*Jerk*	*LREAL*	*Not supported; reserved for future use. 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 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	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 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	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 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.
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	ERROR: Variable (ErrorID_10113_Description) is undefined.

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.

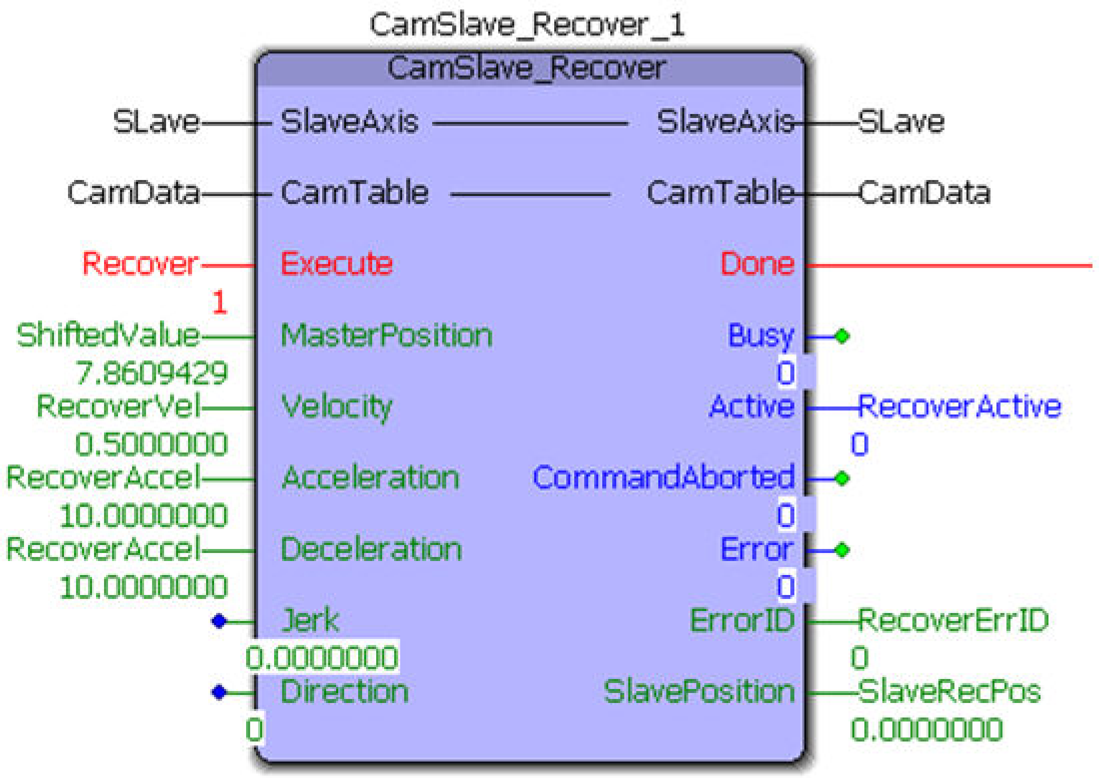
Example

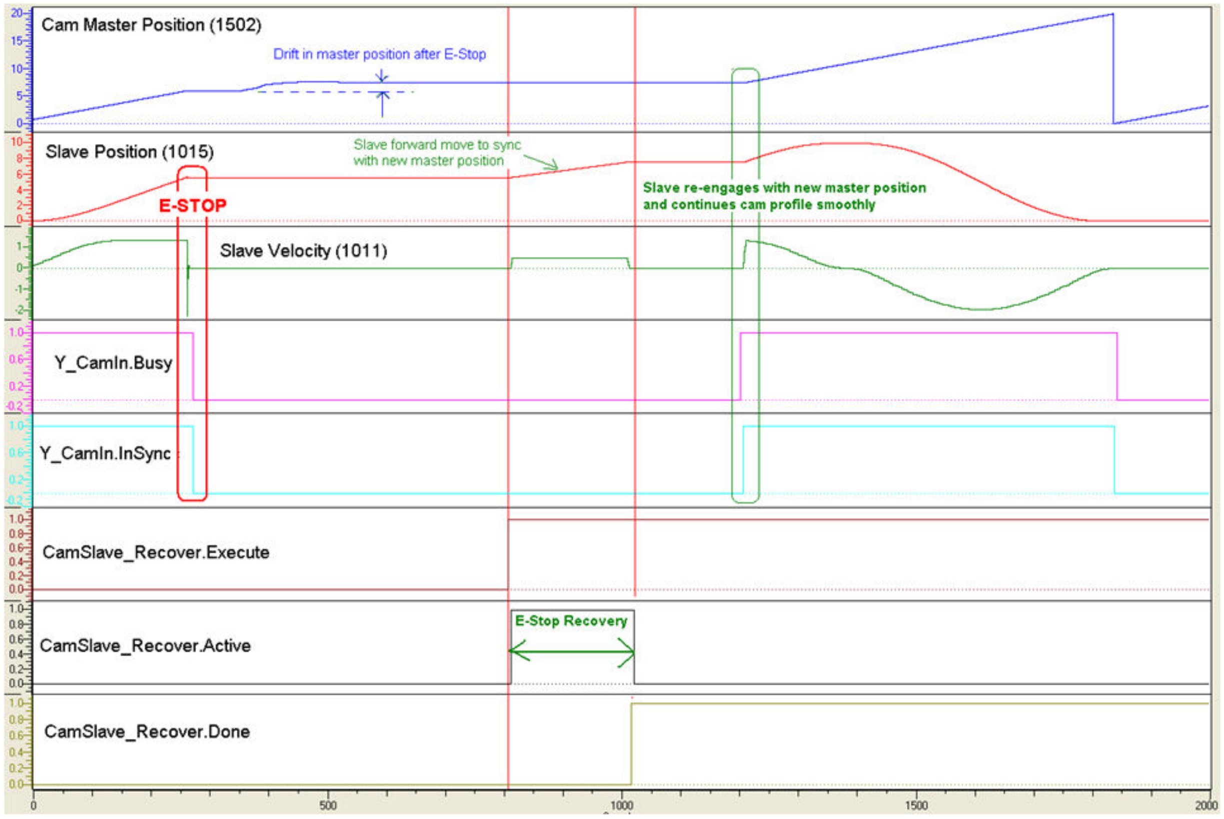
E-Stops can result in the instantaneous loss of control of the axes. Manually clearing debris or scrap from the machine and adjustments after E-Stops and alarms can cause a change in motor position, all resulting in a de synchronization of the master and slave.

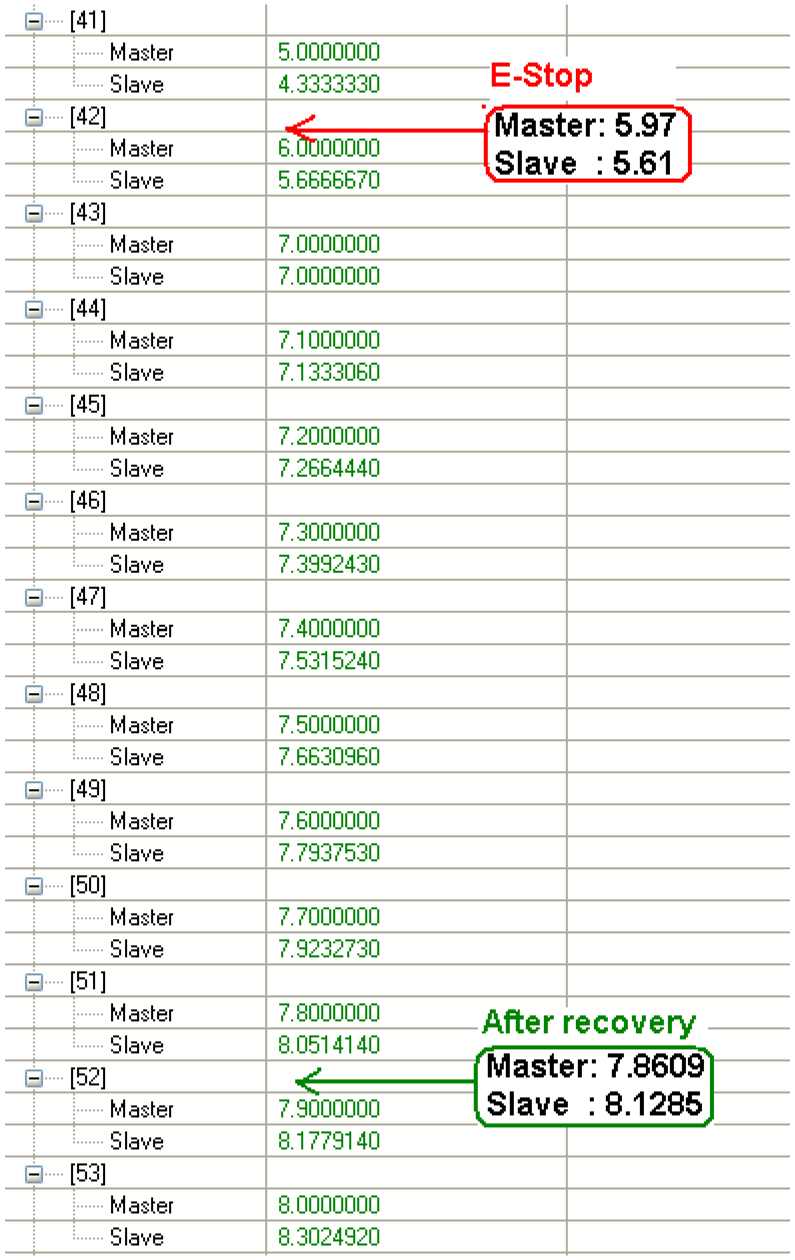
The example given below illustrates how the CamSlave_Recover block can solve E-Stop recovery issues. The logic analyzer plot shows the axes when the E-Stop occurred. At this point, the Y_CamIn outputs InSync and Busy change to FALSE. A slight drift in the master axis position can be seen after the E-Stop. 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.

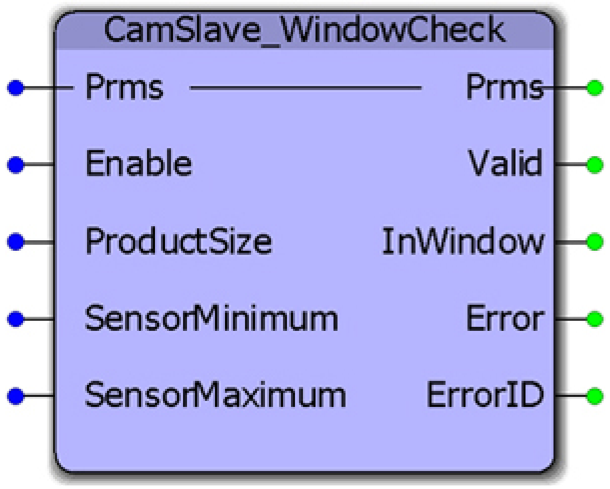



Cam Master Position (1502)
Drift in master position after E-Stop
Slave Position (1015)
Slave forward move to sync with new master position
E-STOP
Slave re-engages with new master position and continues cam profile smoothly
Slave Velocity (1011)
Y_CamIn.Busy
Y_CamIn.InSync
CamSlave_Recover.Execute
CamSlave_Recover.Active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

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.

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 while enable is held high.	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 outputs of the function are valid.	
V	InWindow	BOOL	Indicates the slave output	

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

Notes

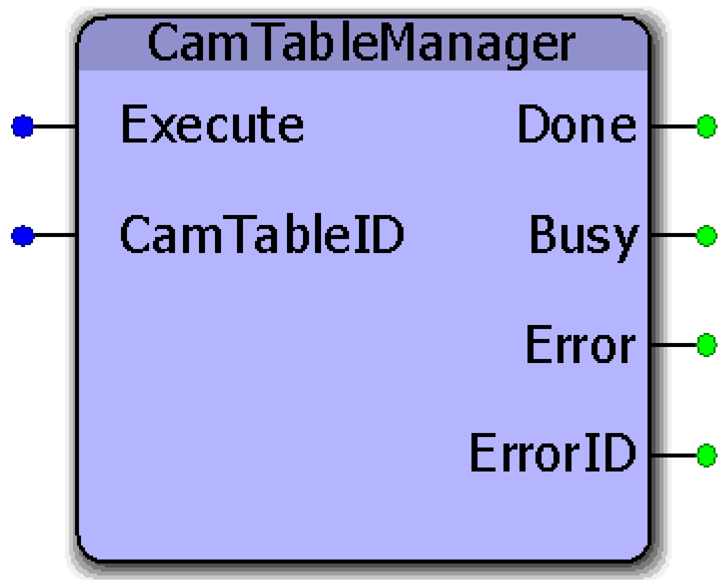
If SensorMinimum and SensorMaximum are both zero, this function does not check for a window and reports InWindow as TRUE.

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	Might be crossed or the same non-zero value

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.

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

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

Notes

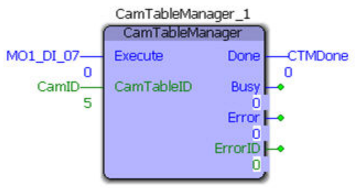
- This function block is unnecessary in applications which use a single, static cam table.
- See the CamTableManager eLearning Module on Yaskawa's YouTube Channel.

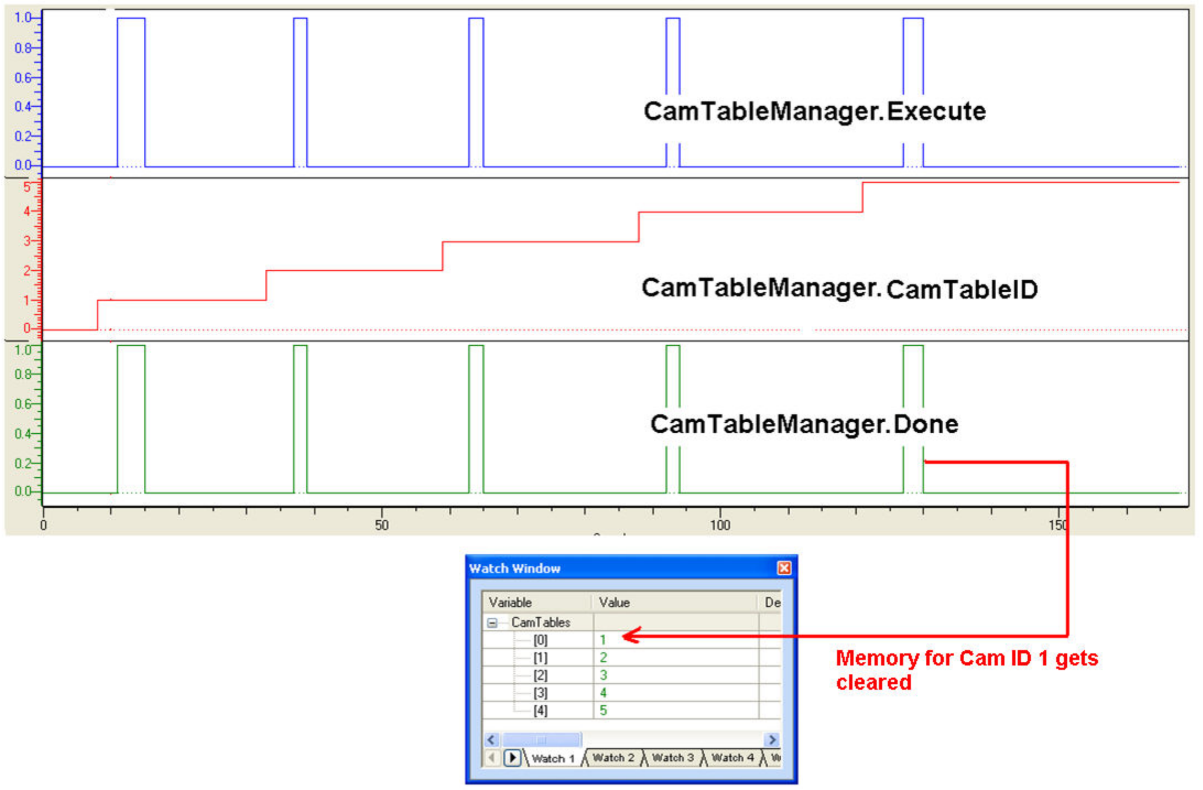
Error Description

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

Example 1

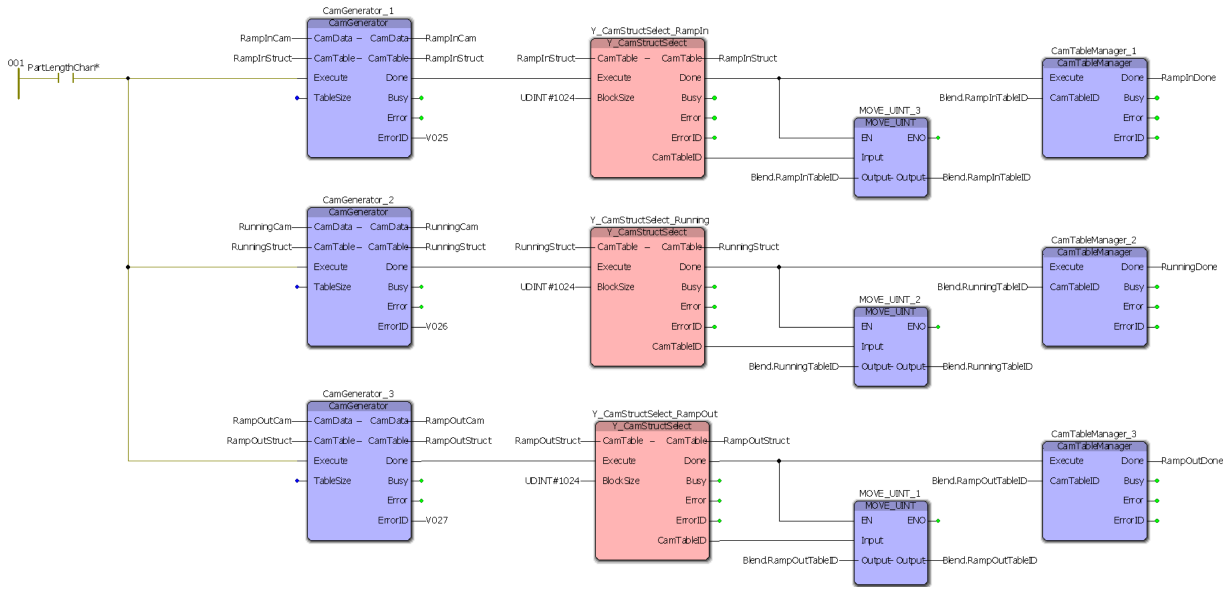
An example of using the CamTableManager is shown below. On the fifth execute of the CamTableManager block, the memory for the oldest CamTable ID gets released. In the example shown below, the memory for CamID 1 gets released. The next execution of the CamTableManager will release the memory for CamID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



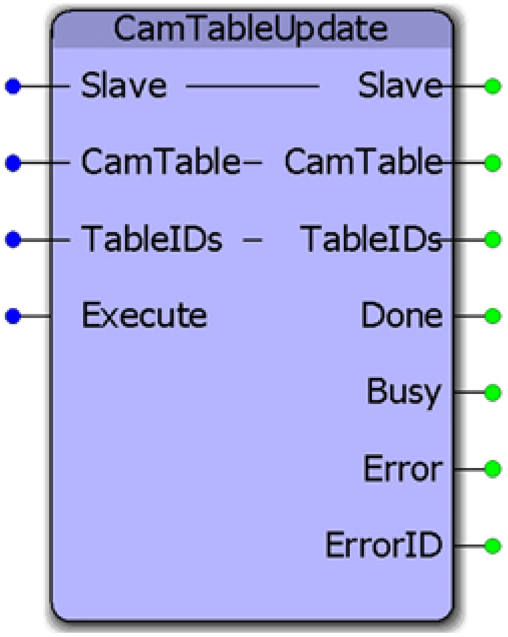


Variable	Value	De
CamTables		
[0]	1	
[1]	2	
[2]	3	
[3]	4	
[4]	5	

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.

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 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.

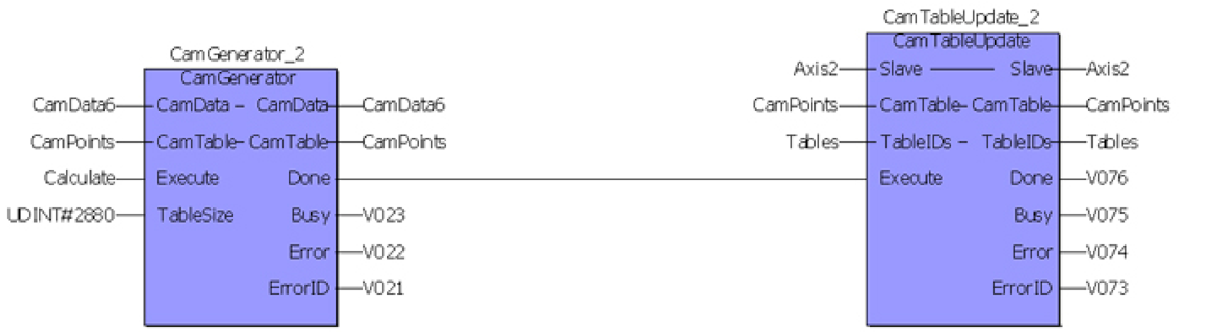
Notes

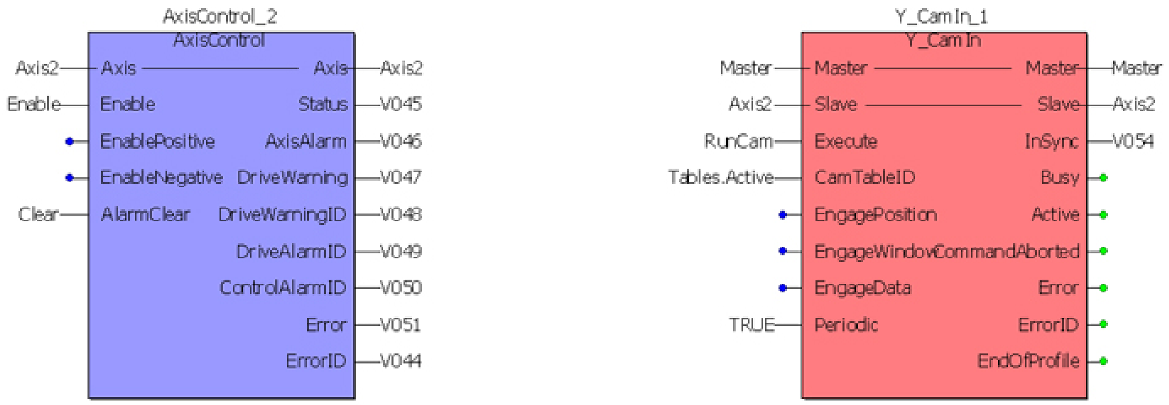
• If both TableIDs in the TableIDs input are zero, then this block automatically uses Y_CamStructSelect to send the first CamTable and obtain the CamTableID.

• 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:

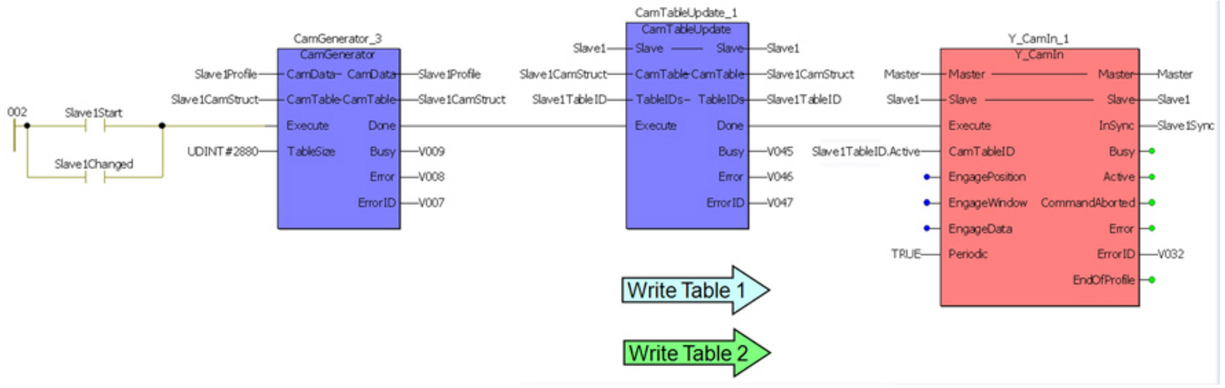
In this example, it is assumed that some event has occurred which triggers the need for a new cam table to be generated using CamGeneator. CamGenerator in turn fires CamTableUpdate to send the new CamTable to the motion engine. CamTableUpdate manages the active and inactive TableIDs, which can then be 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

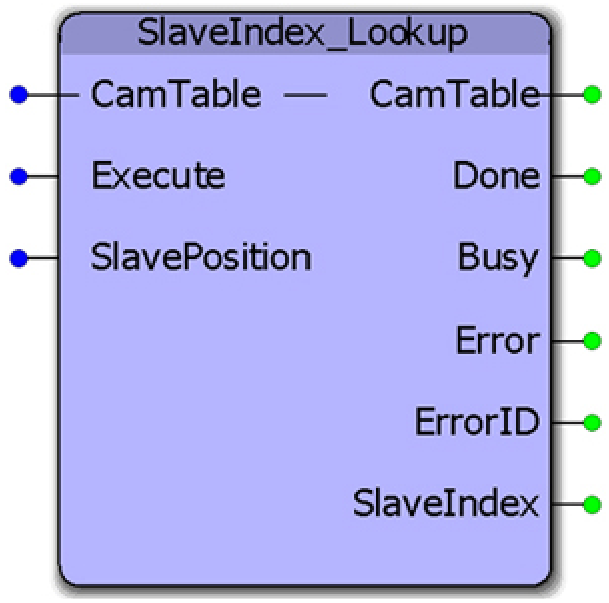


Error Description

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. Cam tables must have a header indicating the number of rows, number of columns and a feed forward velocity flag.
4887	CamTableID does not refer to a valid cam table.

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.

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 been completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or	

			'Enable' input, and reset if Done, CommandAborted, or Error is true.
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
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

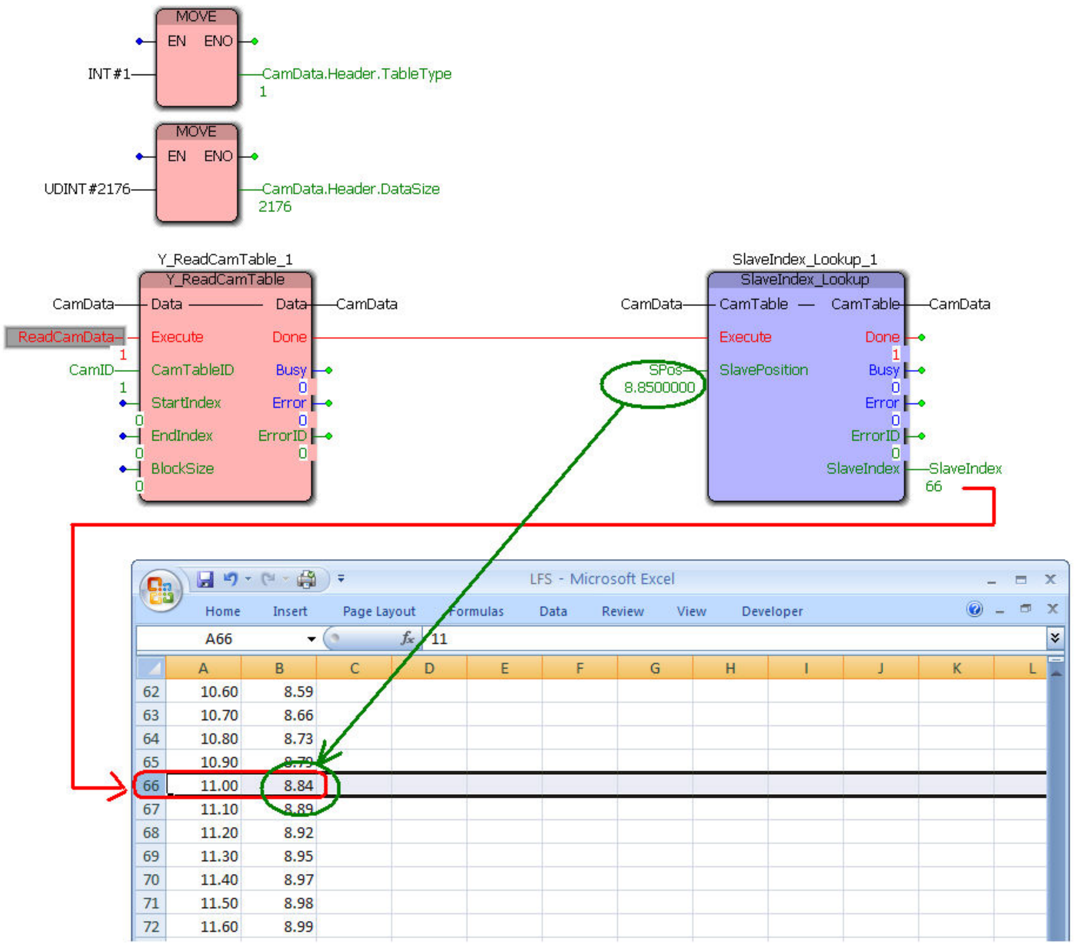
• The SlavePosition input should be a value between the maximum and minimum values of the slave’s position profile for the index value to be valid.

• If the SlavePosition input is a value between two slave positions in the cam table, the SlaveIndex will return the lower index.

Error Description

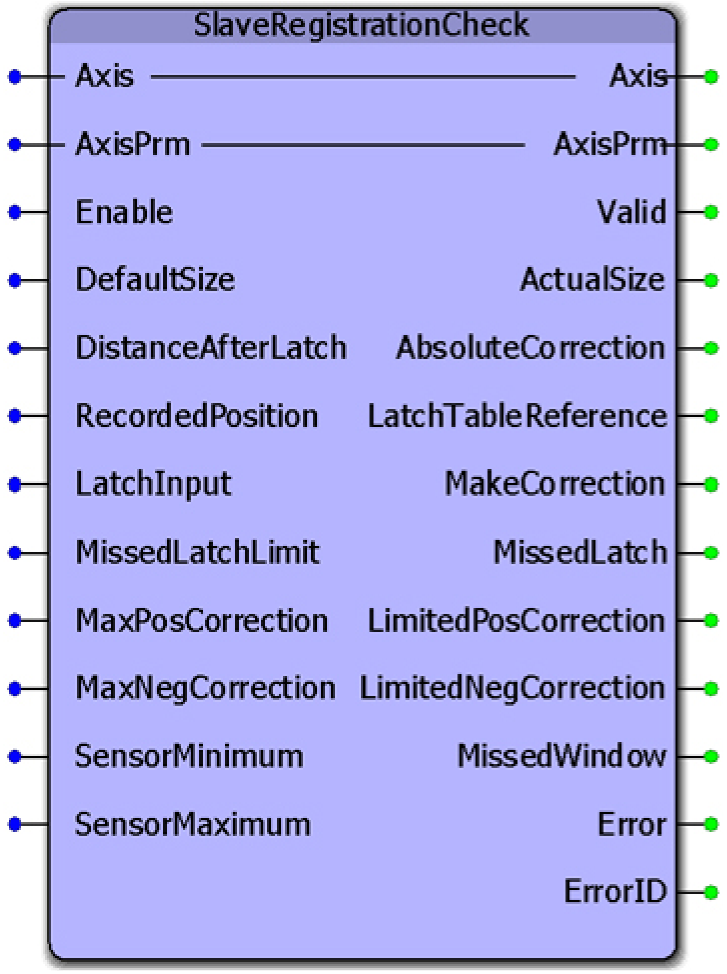
ErrorID	Meaning
0	No error
10045	SlavePosition not found in Y_MS_CAM_STRUCT

Example



	A	B	C	D	E	F	G	H	I	J	K	L
62	10.60	8.59										
63	10.70	8.66										
64	10.80	8.73										
65	10.90	8.79										
66	11.00	8.84										
67	11.10	8.89										
68	11.20	8.92										
69	11.30	8.95										
70	11.40	8.97										
71	11.50	8.98										
72	11.60	8.99										

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.

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	AXIS_REF	See AxisParameterStruct definition in MotionBlock types folder

VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute while enable is held high.	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 [u]). Used with MC_TouchProbe.RecordedPosition.	LREAL#0.0
B	LatchInput	BOOL	Typically connected to MC_TouchProbe.Done, signals the function to calculate any required correction amount	FALSE
B	MissedLatchLimit	UINT	The number of consecutive product lengths that can occur without seeing a mark in the window. Valid sensor detections will reset the internal counter. The next valid sensor detection will reset the internal counter.	
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	SensorMinimum			
V	SensorMaximum			UINT#0 (interpreted as infinite)
VAR_OUTPUT				
B	Valid	BOOL	Indicates that 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	Cumulative number of latches missed	
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 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 (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. Cam tables must have a header indicating the number of rows, number of columns and a feed forward velocity flag.
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.

Communications Toolbox

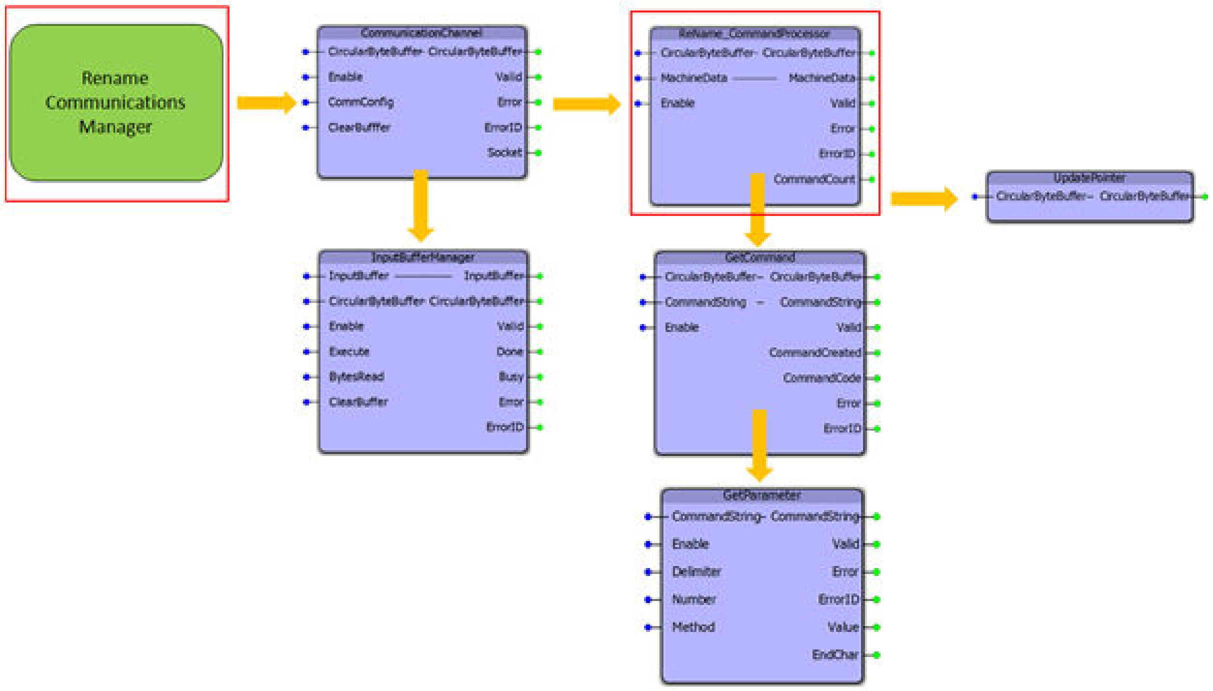
Communications Toolbox

The Communications Toolbox contains functions to provide advanced communication protocols such as DNS, SMTP, and FTP. Also included in the toolbox are a set of functions designed to parse a stream of commands and parameter data from either a serial port or TCP socket into useful machine commands.

Command Streaming Overview

The following graphic shows the organizational flow of functions that are part of the command streaming feature. Certain components of the Command Streaming code require customization for use in your project. The following must be copied from the toolbox, pasted, and renamed in your project.

1. ReName_Communications_Mgr POU
2. ReNameCommandProcessor
3. MyMachineStruct and any relevant sub structures. This may be significantly customized based on your application.



The command streaming tools provided in the Comm Toolbox are designed to interpret commands starting with a two character command code followed by either delimiter separated parameters or no parameters. Example commands are provided in this documentation.

FTP Datatype Customization:

If the file used with FTP needs to be increased in size, both the byte array declaration and the "MAXLENGTH" parameter of FILE_READ in FTP_SendFile need to be changed.

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

Getting Started: Communications

Requirements for v201

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

Firmware libraries:

- YDeviceComm
- PROCONOS

User libraries:

- Yaskawa_Toolbox (v204 or higher)

Communications Revision History

Current Version:

****** 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.

Previous Versions:

****** 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 Types

Data Type: SMTP_Data

Data Type Declaration

```
TYPE

SMTP_Data : STRUCT

DNSIP      : YC_STRING16;  (* DNS server IP (local), used to perform lookup of mail server domain *)

DNSPort    : UINT;         (* DNS port, default is 53, leave blank unless other port is used *)

SMTPDomain  : YC_STRING128;  (* SMTP server domain name (e.g. smtp.yourcompany.com), used for DNS
lookup *)

SMTPIP     : YC_STRING16;  (* The IP of the SMTP server, blank by default, provide IP to override DNS lookup
*)

SMTPPort   : UINT;         (* SMTP port, usually 25 - note: does not support SSL encrypted SMTP *)

LocalIP    : YC_STRING16;  (* Local IP of the controller *)

Domain     : YC_STRING128;  (* Domain for SMTP EHLO/HELO command, example: yaskawa.com *)

Sender     : YC_STRING128;  (* Sender e-mail address, example: john_smith@yaskawa.com *)

SenderName  : YC_STRING32;  (* Name of sender, example: John Smith *)

Subject    : YC_STRING128;  (* Subject of email, example: How awesome is the e-mail function block? *)

RcptArray   : rcpt_array;   (* Array of rcpt_struct (up to 10, or change datatype declaration), email and name
*)

NumRcpt    : INT;          (* Number of emails in rcpt_array *)

Timeout    : TIME;         (* Timeout for connecting to the SMTP server, defaults to 5s *)

END_STRUCT;

END_TYPE
```

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;
```

Data Type: FTP_Data

Data Type Declaration

```
TYPE

FTP_Data   : STRUCT

Username   : YC_STRING32;   (* Username to log in to the FTP server *)

Password   : YC_STRING32;   (* Password to log in to the FTP server *)

LocalIP    : YC_STRING16;   (* Local IP of the controller *)

FTPDomain  : YC_STRING128;  (* The domain name of the FTP server that will be resolved via DNS *)

FTPIP      : YC_STRING16;   (* The IP of the FTP server if a domain is not known or set *)

FTPPort    : UINT;          (* The port to connect to the FTP server through, default 21 *)

DNSIP      : YC_STRING16;   (* The DNS lookup server IP *)

DNSPort    : UINT;          (* The DNS port to connect through, default 53 *)

Timeout    : TIME;          (* Timeout for connecting to the FTP server or data connection, default 5s *)

END_STRUCT;

END_TYPE
```

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: CircularBufferStruct

Data Structure used to manage a circular buffer of data used by multiple function blocks.

Data Type Declaration

```
CircularBufferStruct:STRUCT

StorePointer:INT; (* FB Output - Pointer updated when new elements added to buffer *)

UsePointer:INT; (* FB Output -  Pointer updated when elements of buffer have been read *)

Size:INT; (*  User Input - Size of circular buffer  *)

CmdDelimiters:DelimiterArray; (* User Input - Delimiters separating Command Strings. Default is carriage
return or carriage return line feed *)

PrmDelimiter:YTB_STRING1; (*  User Input - Delimiters separating parameters within a command. Default is a
comma *)

LastDelimiter:INT; (* Element used by GetCommand *)

Data:YTB_ByteArray8192;

END_STRUCT;
```

Data Type: CommStruct

For use with CommunicationChannel function block. Contains information about the communication interface used.

Data Type Declaration

```
CommStruct: STRUCT

CommType:INT; (*  Set 1 for Serial, 2 for Ethernet  *)

InactivityTimeout:TIME; (*  Use this to allow the MPiec to close the socket if no communication has been received on the channel in the time required.  *)

BufferSize:UDINT; (* number of bytes to read per scan from Ethernet buffer, if left at 0 entire buffer will be transferred *)

Serial:SerialConfig;

Ethernet:EthernetConfig;

END_STRUCT;
```

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

```
EthernetConfig: STRUCT

LocalIPAddress:STRING; (*  User Input – Ethernet address of controller *)

LocalPort:UINT; (*  User Input – Ethernet port number to open*)

RemoteIPAddress:STRING;

RemotePort:UINT;

END_STRUCT;
```

Data Type: RecipientArray

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

Data Type Declaration

```
TYPE

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

END_TYPE
```

Data Type: RecipientStruct

```
TYPE

RecipientStruct : STRUCT

Email : YC_STRING128;

Name  : YC_STRING32;

END_STRUCT;

END_TYPE
```

Data Type: SerialConfig

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

Data Type Declaration

```
SerialConfig: STRUCT

PortNum:UINT; (*   For use with the Y_OpenSerialPort function block    *)

BaudRate:DINT; (*   For use with the Y_SetDeviceOption function block   *)

DataBits:DINT; (*   For use with the Y_SetDeviceOption function block   *)

StopBits:DINT; (*   For use with the Y_SetDeviceOption function block   *)

Parity:DINT; (*   For use with the Y_SetDeviceOption function block   *)

HandShake:DINT; (*   For use with the Y_SetDeviceOption function block   *)

END_STRUCT;
```

Enumerated Types

Enumerated Type: COM_Type

Enum Type for CommType

Data Type Declaration

```
COM_Type:(na,Serial,Ethernet); (*   Enumerated type to be used with CommStruct.CommType   *)
```

Enumerated Type: Method

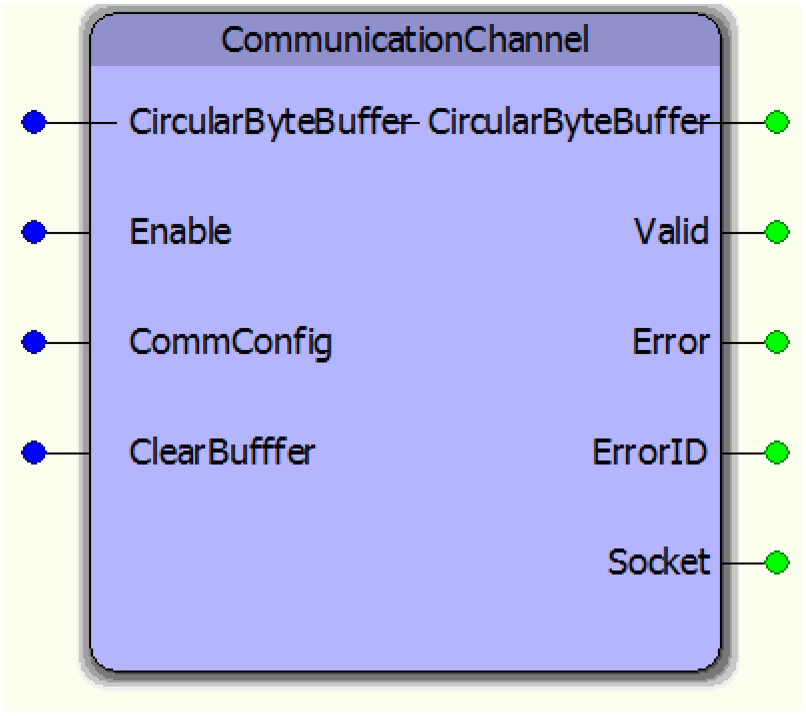
Enum Type for GetParameter Method

Data Type Declaration

```
Method:(Parameter,Character); (*   For use with the GetParameter function.  Specifies how the value is obtained.   *)
```

Function Blocks

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.

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 while enable is held high.	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 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 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

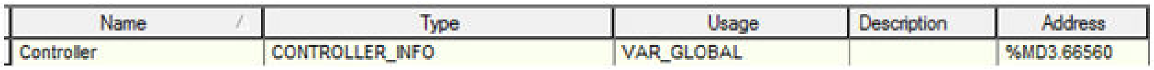
ErrorID	Meaning
0	No error
8705	The maximum number of concurrently open user IO devices (sockets/files) has been reached.
8706	The socket handle was invalid.
8707	The IP address string was not in a valid format.
8708	The socket could not be created.
8709	The specified address or port is already in use on the local network.
8710	The specified address or port is not available for use.
8711	Unable to accept new socket connection.
8712	Unable to bind to the specified address.
8713	The socket type argument was invalid.
8714	The local address or port was not valid.
8715	The socket could not be connected.
8716	There is no network routing path to the specified address.
8717	The socket is already connected to another endpoint.
8718	The socket connection attempt was actively refused by the remote peer.
8719	The socket 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	The Buffer argument to WriteDevice and ReadDevice is required.
8724	The device option ID was invalid.
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 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 controllers 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

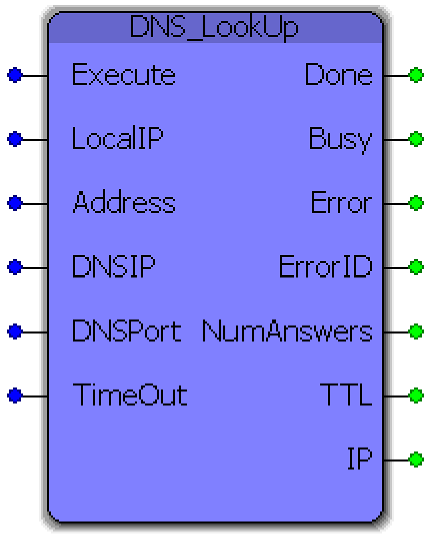
2. Add the following code to the initialize routine to obtain controller's IP address. The variable IPAddress is a STRING. The BUF_TO_STRING function block is located in the PROCONOS firmware library. As shown below, we are using it to extract 15 bytes of the IPAddress. These bytes equate to xxx.xxx.xxx.xxx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

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

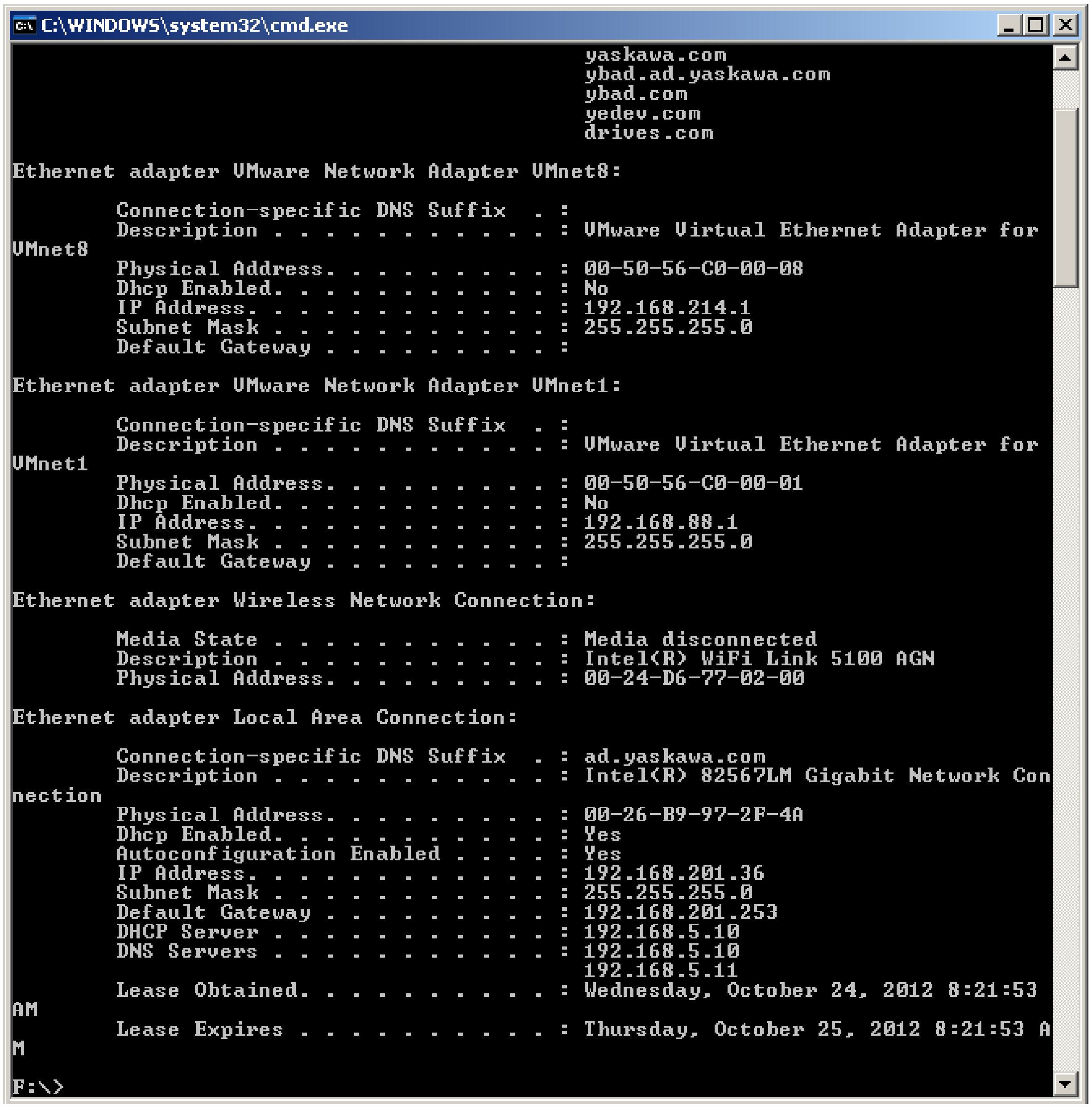
- 'Address' must be a domain name (i.e. yaskawa.com), not an IPV4 address. Passing an IPV4 address is what is referred to as a "reverse DNS lookup" and is not supported by this block (reason: the Y_DeviceComm library needs an IPV4 address, not a domain name).

- 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

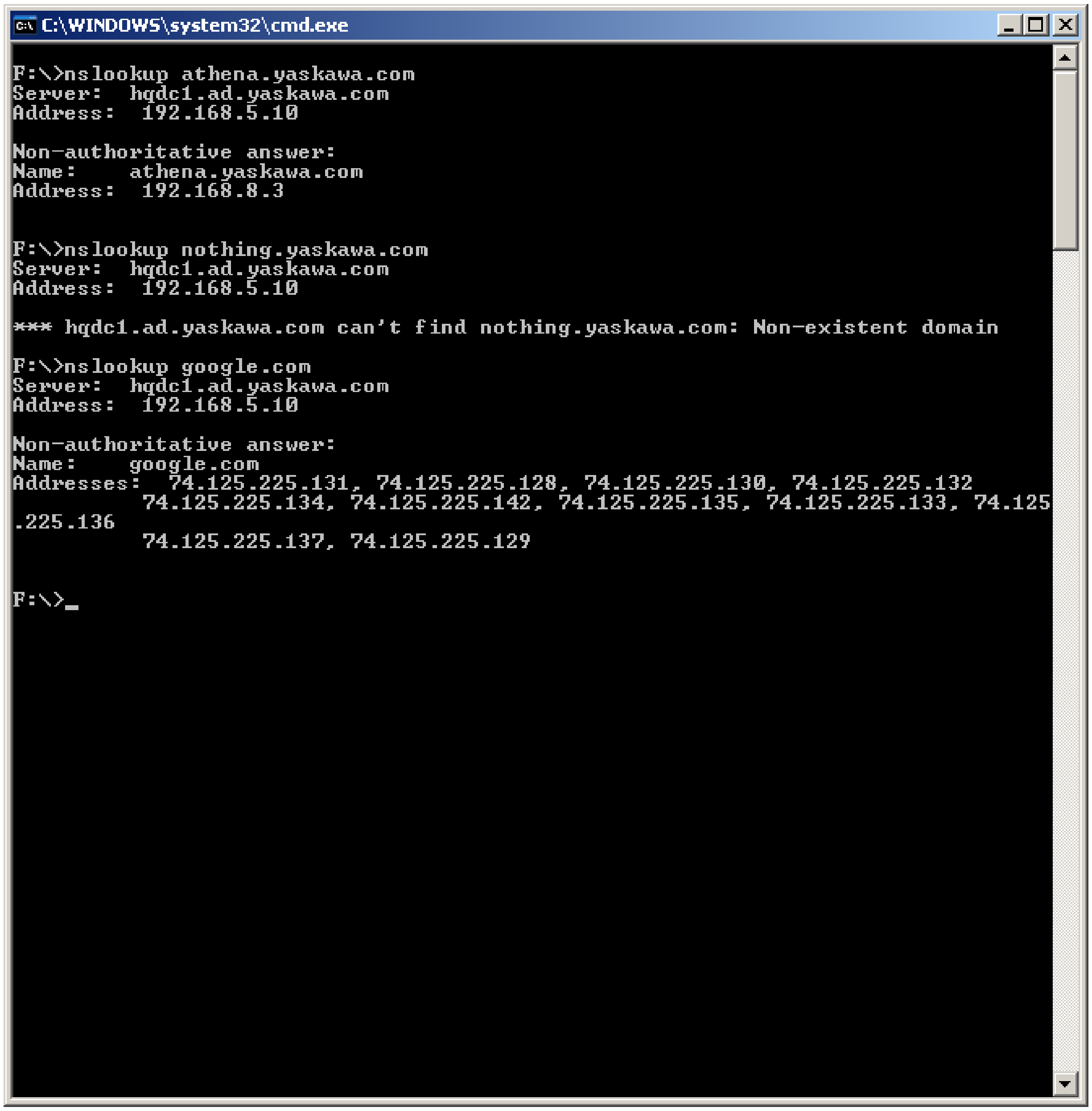
In order to perform a DNS lookup a connection to a DNS server must first be established. What DNS server you configure this block to use depends on your particular network set up. The easiest way to determine what DNS server to use (or at least to get started) is to open up the Windows command prompt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



C:\WINDOWS\system32\cmd.exe

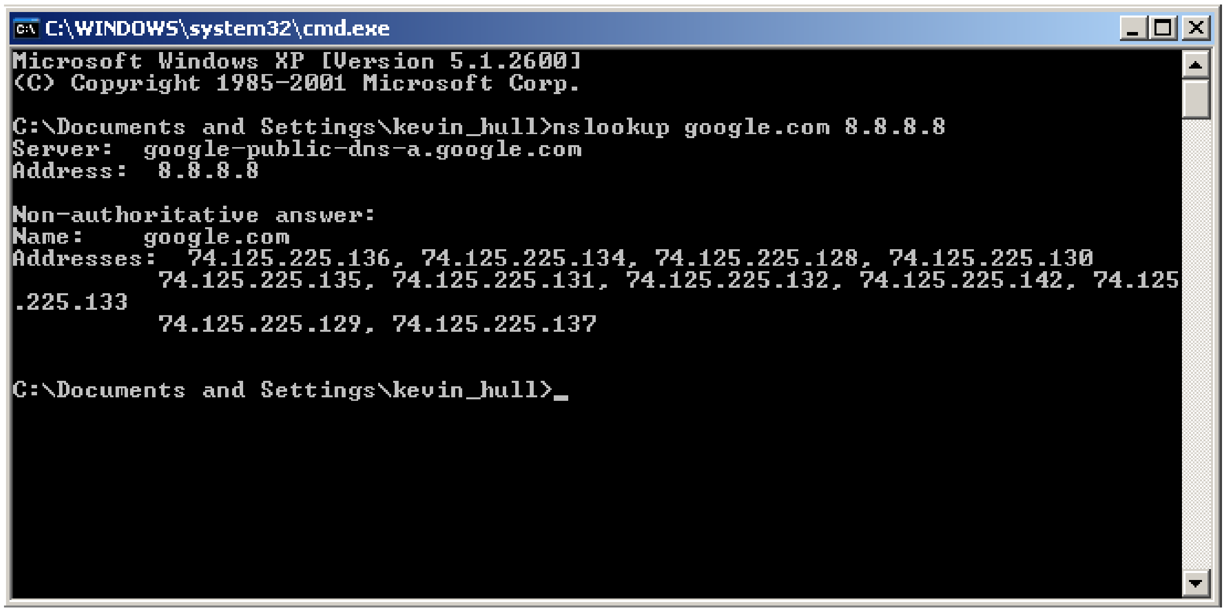
```
                                                yaskawa.com
                                                ybad.ad.yaskawa.com
                                                ybad.com
                                                yedev.com
                                                drives.com

Ethernet adapter VMware Network Adapter VMnet8:

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

Ethernet adapter VMware Network Adapter VMnet1:

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

Ethernet adapter Wireless Network Connection:

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

Ethernet adapter Local Area Connection:

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

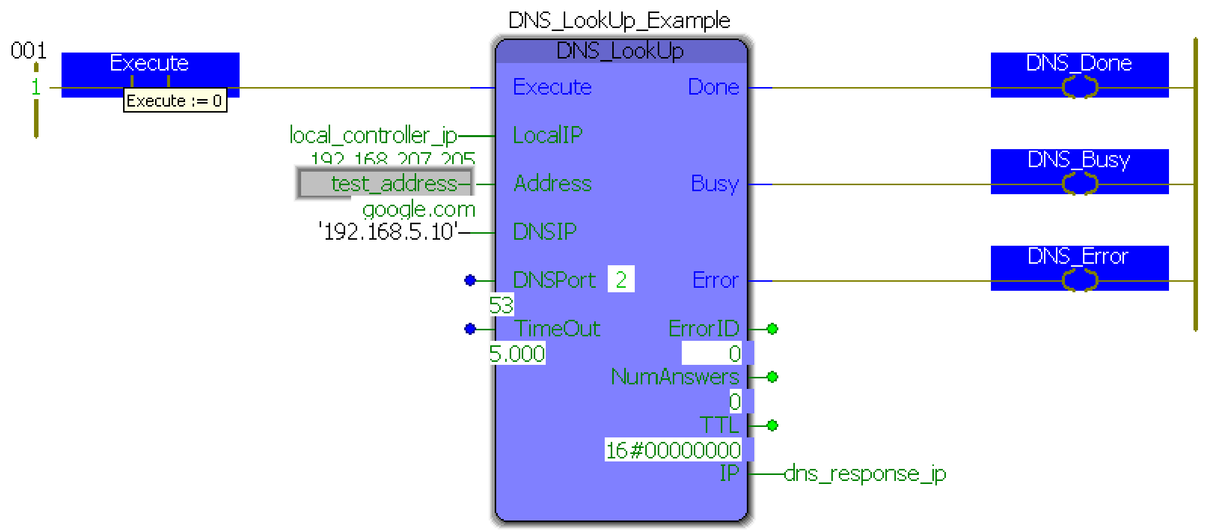
Error Description

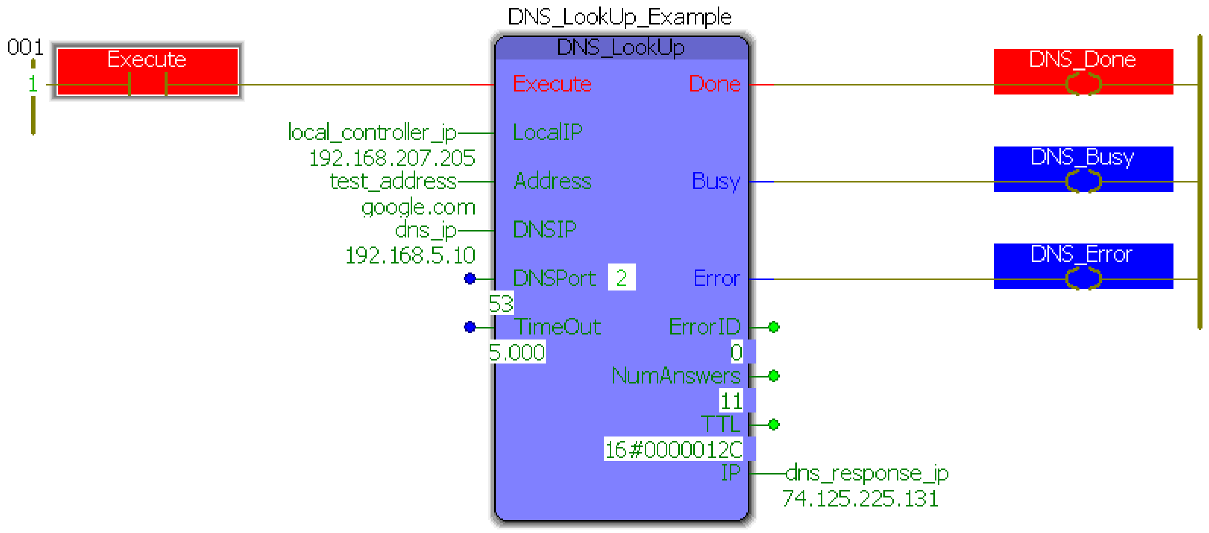
ErrorID	Meaning
0	No error
8705	The maximum number of concurrently open user IO devices (sockets/files) has been reached.
8706	The socket handle was invalid.
8707	The IP address string was not in a valid format.
8708	The socket could not be created.
8709	The specified address or port is already in use on the local network.
8710	The specified address or port is not available for use.
8711	Unable to accept new socket connection.
8712	Unable to bind to the specified address.
8713	The socket type argument was invalid.
8714	The local address or port was not valid.
8715	The socket could not be connected.
8716	There is no network routing path to the specified address.
8717	The socket is already connected to another endpoint.
8718	The socket connection attempt was actively refused by the remote peer.
8719	The socket 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	The Buffer argument to WriteDevice and ReadDevice is required.

8724	The device option ID was invalid.
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 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

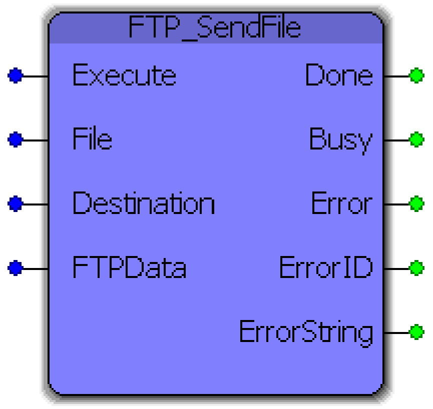
The following example demonstrates the blocks ability to perform a lookup for an external address ("google.com") using an internal DNS server. The LocalIP, Address and DNSIP have all be configured and DNSPort and TimeOut have been left to defaults.





If you compare the output of the block ("74.125.225.131") to the nslookup performed above you will that this IP address is in the list. You can also see that NumAnswers is set to 11 which also 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.

FTP_SendFile



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

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.	
V	File	YC_STRING128	The full file name and location on the controller, e.g. '/flash/user/data/example.csv'.	
V	Destination	YC_STRING64	The full file name and destination on the FTP server, e.g. 'metrics/example.csv'.	
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).

- It is suggested that your FTP server either have a internal/external domain name OR use a static IP address as a change in address will prevent the block from transferring files. See "Setup" for more details.

- Your FTP user account for this block must have "Write" privileges to be able to write files to the server. Optionally, your account may also have "Append" privileges. Note that if your destination files already exists and your user only has "Write" then the file will be overwritten. If the file exists and your user has "Append" 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 IO devices (sockets/files) has been reached.
8706	The socket handle was invalid.
8707	The IP address string was not in a valid format.
8708	The socket 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.
8711	Unable to accept new socket connection.
8712	Unable to bind to the specified address.
8713	The socket type argument was invalid.
8714	The local address or port was not valid.
8715	The socket could not be connected.
8716	There is no network routing path to the specified address.
8717	The socket is already connected to another endpoint.
8718	The socket connection attempt was actively refused by the remote peer.
8719	The socket 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	The Buffer argument to WriteDevice and ReadDevice is required.

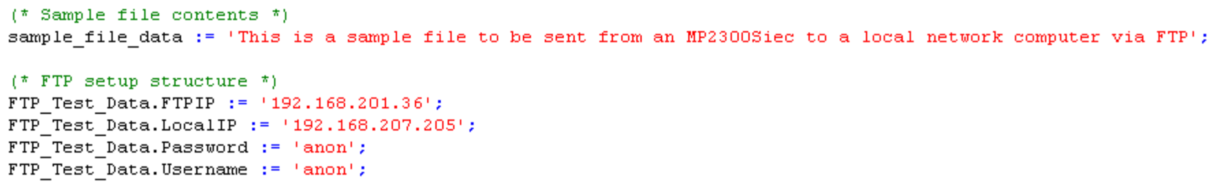
8724	The device option ID was invalid.
8725	The device option value was not the right size or the data was out of range.
8726	The serial port ID was not a valid serial port.
8727	The serial port could not be opened.
12200	Connect to FTP server timeout, no connection was established within the supplied 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 unrecognised
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	Transcation failed

Basic Functionality Example - Transferring a File

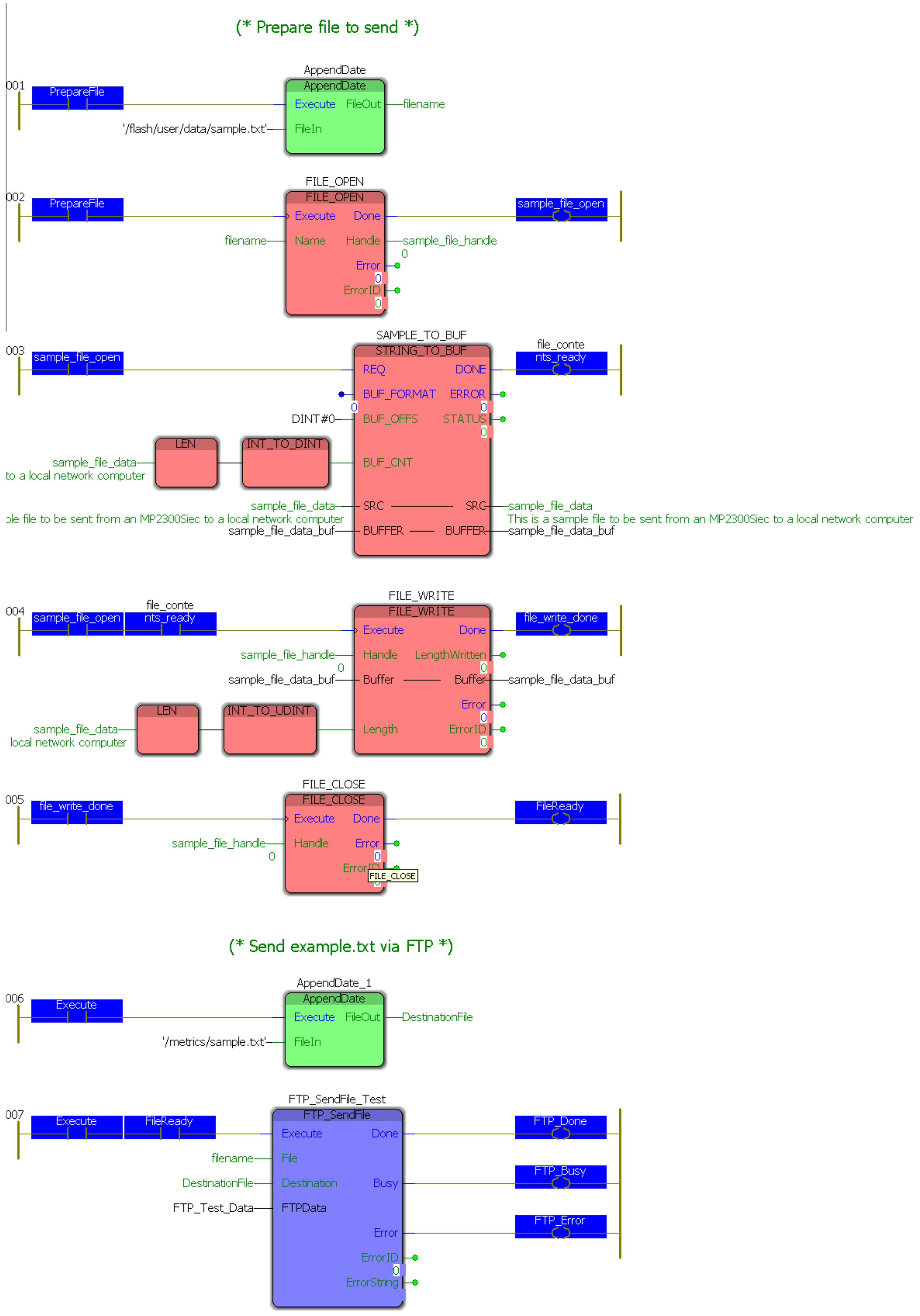
This examples demonstrates how to configure the block using the data structure, create a file to send and execute the FTP_SendFile block.

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.



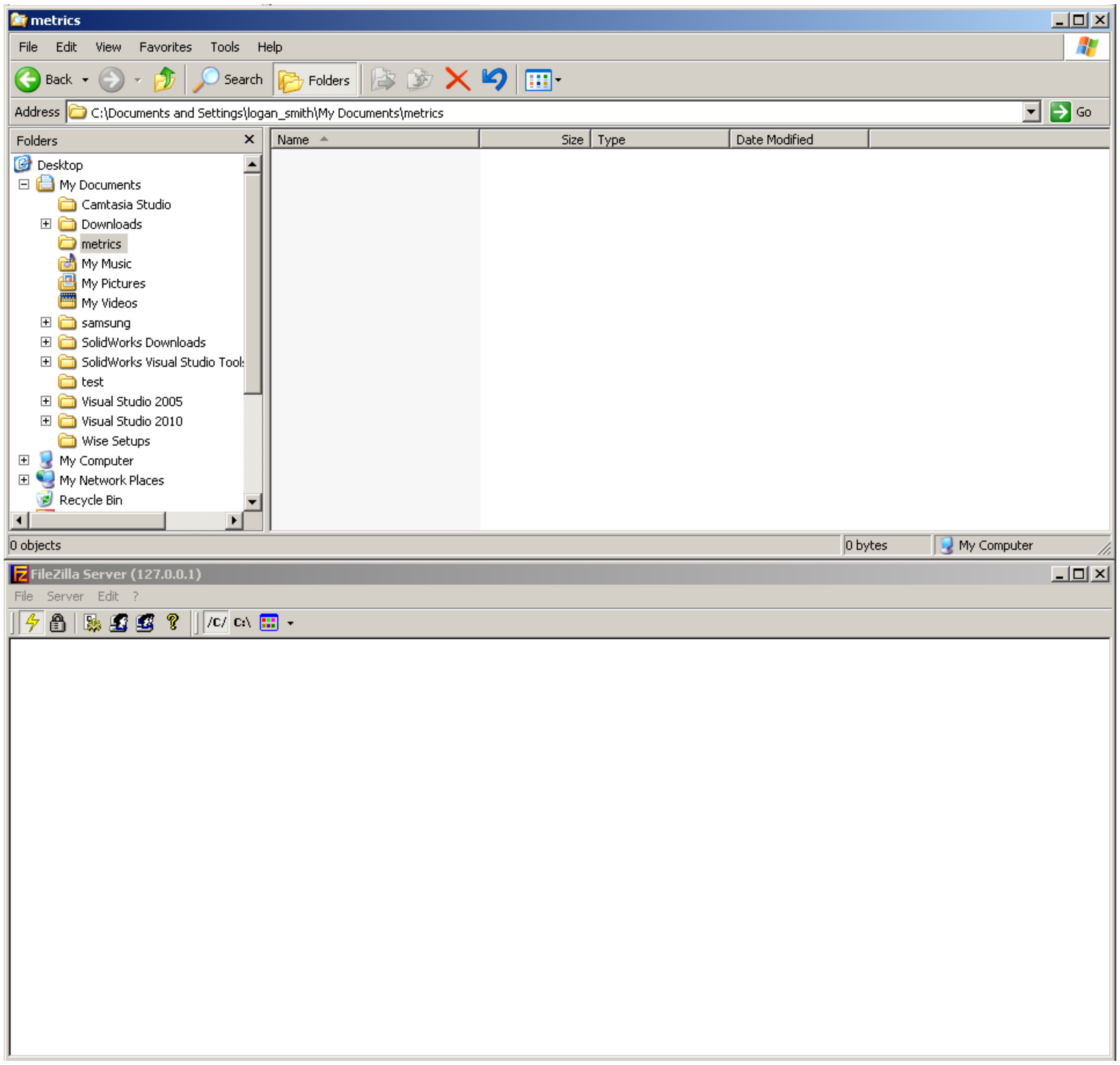
```
(* Sample file contents *)
sample_file_data := 'This is a sample file to be sent from an MP2300Siec to a local network computer via FTP';

(* 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.

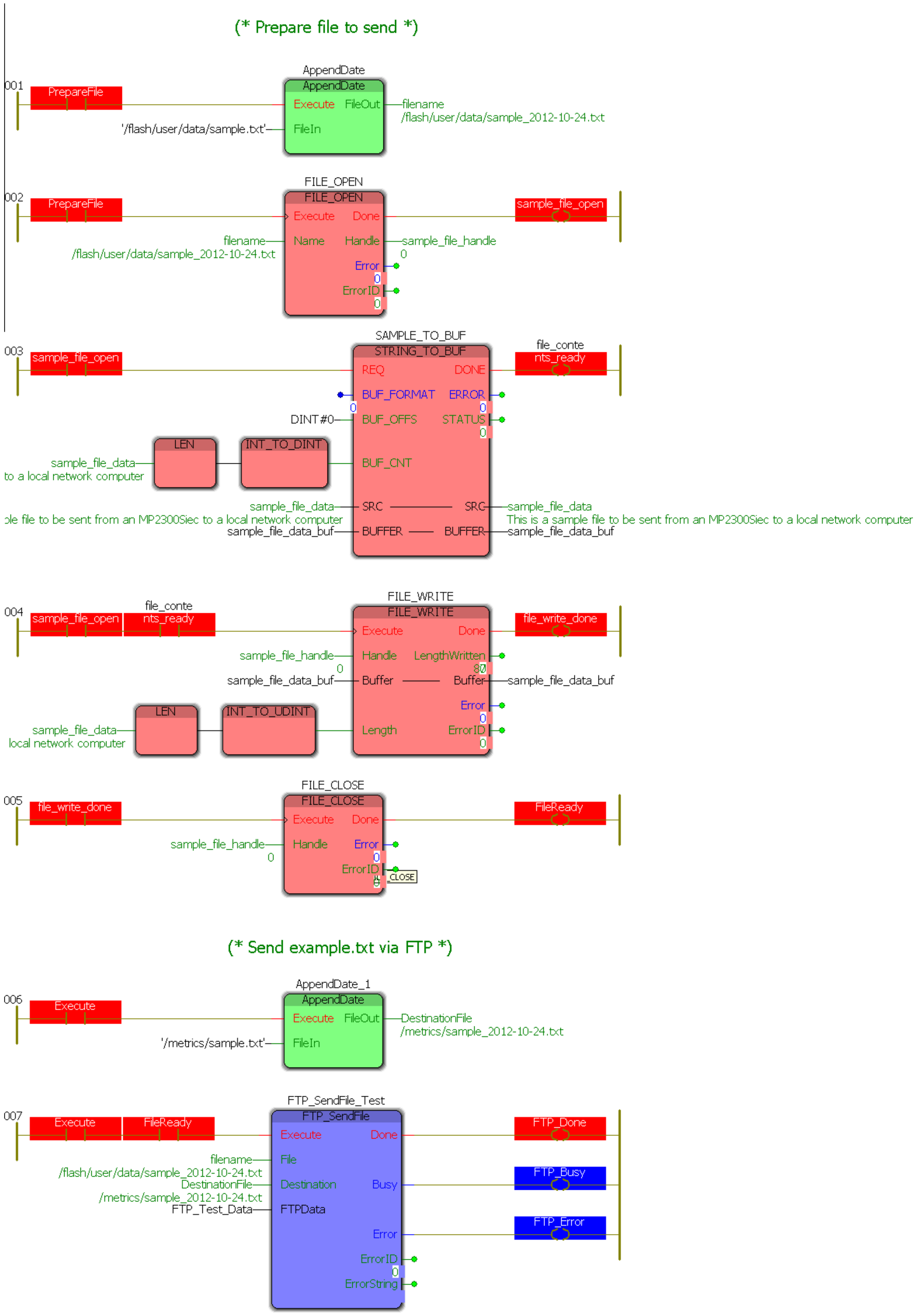


(* Prepare file to send *)
001
PrepareFile
AppendDate
AppendDate
Execute
FileOut
filename
'/flash/user/data/sample.txt'
FileIn
002
PrepareFile
FILE_OPEN
FILE_OPEN
Execute
Done
sample_file_open
filename
Name
Handle
sample_file_handle
0
Error
0
ErrorID
0
003
sample_file_open
SAMPLE_TO_BUF
STRING_TO_BUF
REQ
DONE
file_conte nts_ready
BUF_FORMAT
ERROR
0
DINT#0
BUF_OFFS
STATUS
0
LEN
INT_TO_DINT
sample_file_data
to a local network computer
BUF_CNT
sample_file_data
ple file to be sent from an MP2300Siec to a local network computer
SRC
SRC
sample_file_data
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
sample_file_data_buf
Buffer
Buffer
sample_file_data_buf
LEN
INT_TO_UDINT
Error
0
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

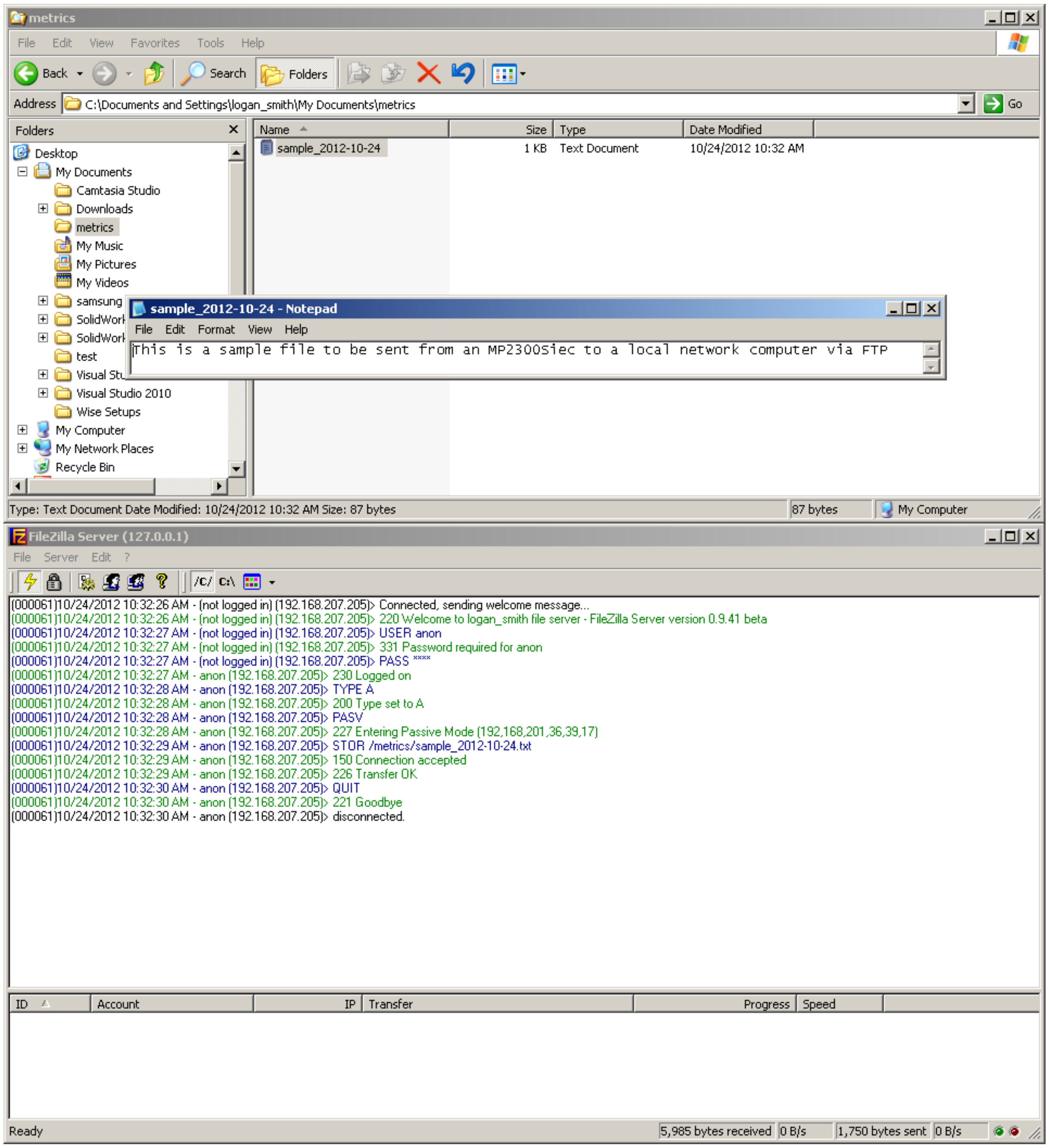
The destination folder is empty to begin with and the FTP server log has been cleared prior to connection so that the results will be obvious.



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 *)
001
PrepareFile
AppendDate
Execute FileOut
FileIn
'/flash/user/data/sample.txt'
filename
/flash/user/data/sample_2012-10-24.txt
002
PrepareFile
FILE_OPEN
Execute Done
Name Handle
Error
ErrorID
filename
/flash/user/data/sample_2012-10-24.txt
sample_file_open
sample_file_handle
003
sample_file_open
SAMPLE_TO_BUF
STRING_TO_BUF
REQ DONE
BUF_FORMAT ERROR
BUF_OFFS STATUS
BUF_CNT
SRC SRC
BUFFER BUFFER
DINT#0
LEN
INT_TO_DINT
sample_file_data
to a local network computer
ple file to be sent from an MP2300Siec to a local network computer
sample_file_data_buf
file_conte
nts_ready
This is a sample file to be sent from an MP2300Siec to a local network computer
004
sample_file_open
file_conte
nts_ready
FILE_WRITE
Execute Done
Handle LengthWritten
Buffer Buffer
Error
Length ErrorID
sample_file_handle
sample_file_data_buf
LEN
INT_TO_UDINT
sample_file_data
local network computer
file_write_done
005
file_write_done
FILE_CLOSE
Execute Done
Handle Error
ErrorID
CLOSE
sample_file_handle
FileReady
(* Send example.txt via FTP *)
006
Execute
AppendDate_1
AppendDate
Execute FileOut
FileIn
'/metrics/sample.txt'
DestinationFile
/metrics/sample_2012-10-24.txt
007
Execute
FileReady
FTP_SendFile_Test
FTP_SendFile
Execute Done
File
Destination Busy
FTPData
Error
ErrorID
ErrorString
filename
/flash/user/data/sample_2012-10-24.txt
DestinationFile
/metrics/sample_2012-10-24.txt
FTP_Test_Data
FTP_Done
FTP_Busy
FTP_Error

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



metrics
File Edit View Favorites Tools Help
Back Search Folders
Address C:\Documents and Settings\logan_smith\My Documents\metrics
Folders
Desktop
My Documents
Camtasia Studio
Downloads
metrics
My Music
My Pictures
My Videos
samsung
SolidWork
SolidWork
test
Visual Stu
Visual Studio 2010
Wise Setups
My Computer
My Network Places
Recycle Bin
Name Size Type Date Modified
sample_2012-10-24 1 KB Text Document 10/24/2012 10:32 AM
Type: Text Document Date Modified: 10/24/2012 10:32 AM Size: 87 bytes
87 bytes
My Computer
sample_2012-10-24 - Notepad
File Edit Format View Help
FileZilla Server (127.0.0.1)
File Server Edit ?
ID Account IP Transfer Progress Speed
Ready
5,985 bytes received 0 B/s 1,750 bytes sent 0 B/s

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

```
(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.
```

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.

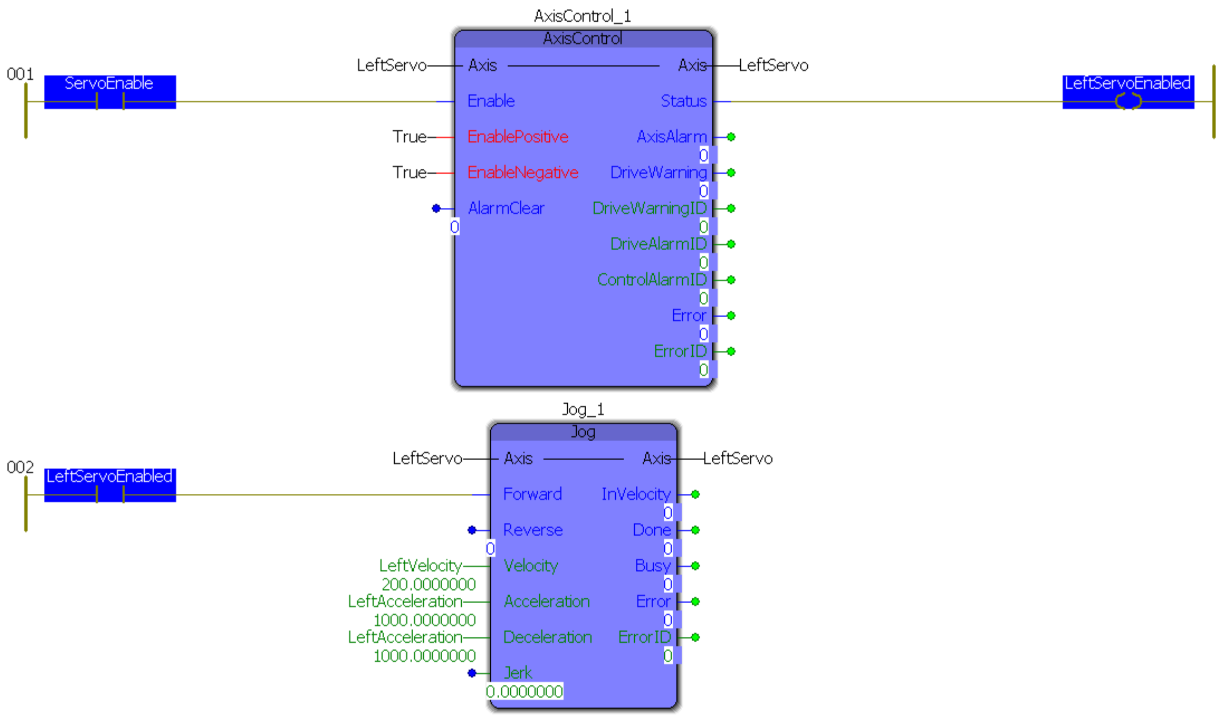
Advanced Functionality Example - Transferring a Metrics File at a Specified Rate

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.

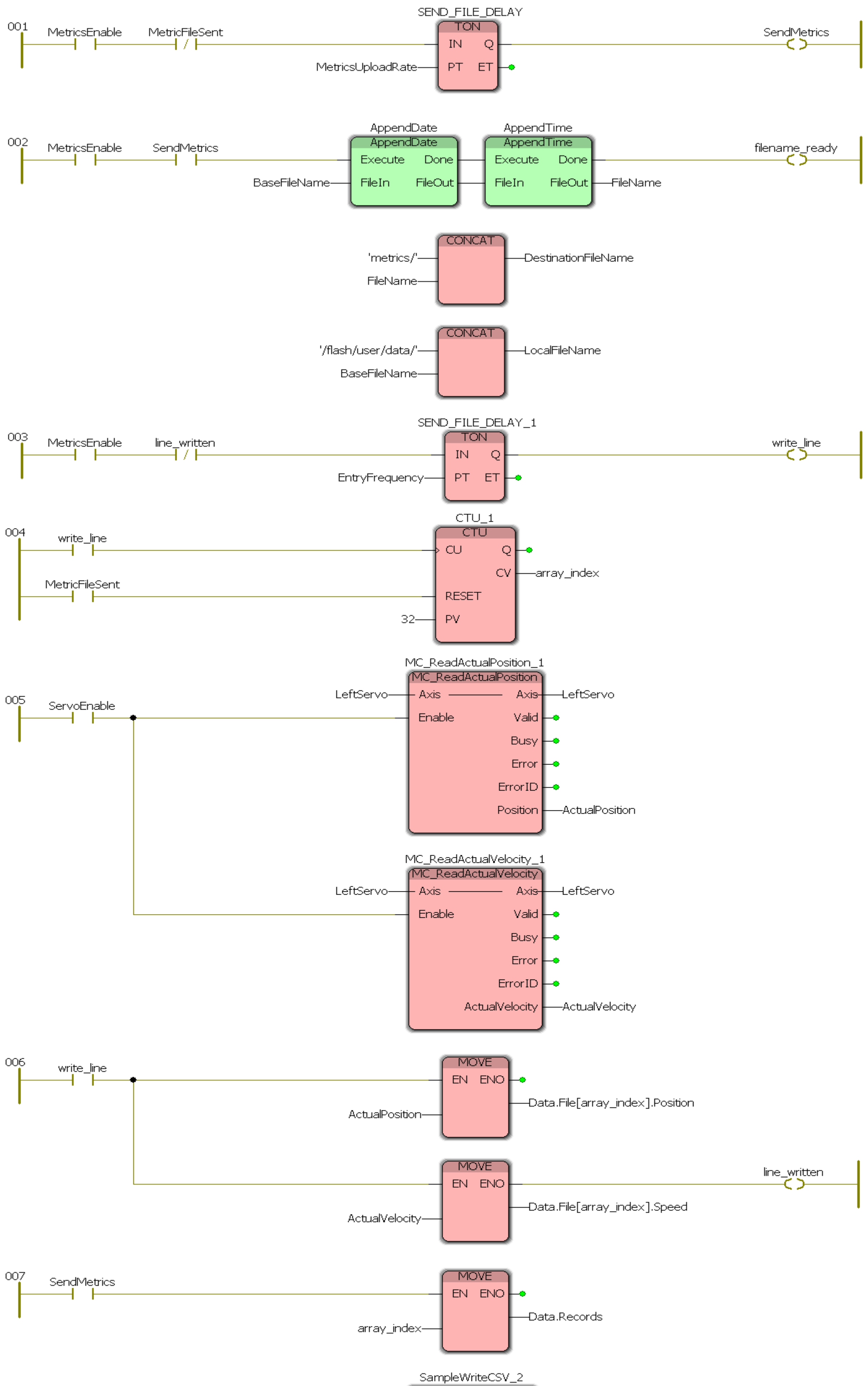
The same data configuration structure was used but there is no preset message for the file as it will be created dynamically.

```
(* 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.

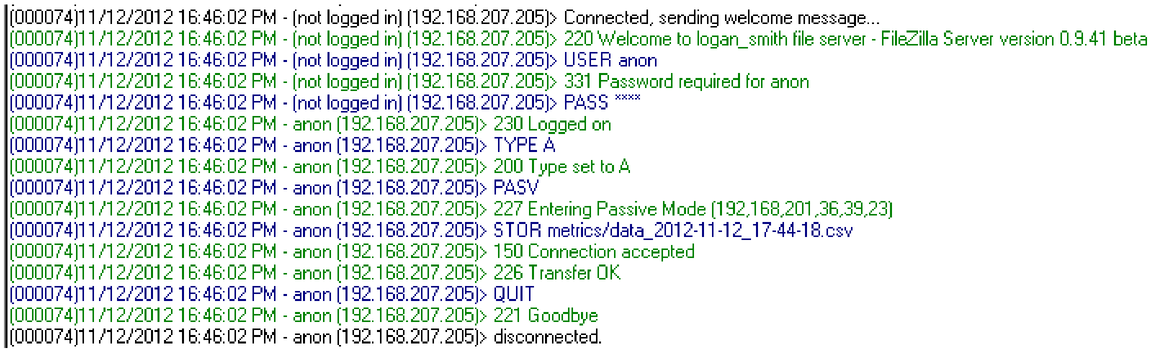


Controlling this example is very simple. The servo is turned on by "ServoEnable" which then in turn starts the jog at a constant velocity. The rest of the example is controlled in the main program:

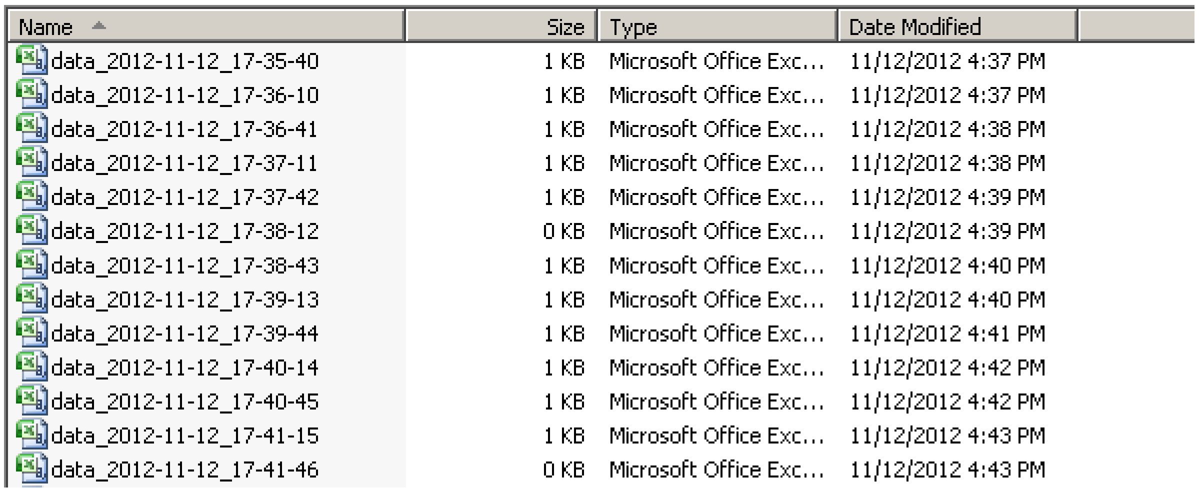

SEND_FILE_DELAY
001
MetricsEnable
MetricFileSent
TON
IN Q
PT ET
MetricsUploadRate
SendMetrics
002
MetricsEnable
SendMetrics
AppendDate
Execute Done
FileIn FileOut
BaseFileName
AppendTime
Execute Done
FileIn FileOut
FileName
filename_ready
CONCAT
'metrics/'
FileName
DestinationFileName
CONCAT
'/flash/user/data/'
BaseFileName
LocalFileName
SEND_FILE_DELAY_1
003
MetricsEnable
line_written
TON
IN Q
PT ET
EntryFrequency
write_line
CTU_1
004
write_line
MetricFileSent
CTU
CU Q
CV
RESET
PV
32
array_index
MC_ReadActualPosition_1
005
ServoEnable
MC_ReadActualPosition
LeftServo
Axis Axis
Enable Valid
Busy
Error
ErrorID
Position
LeftServo
ActualPosition
MC_ReadActualVelocity_1
MC_ReadActualVelocity
LeftServo
Axis Axis
Enable Valid
Busy
Error
ErrorID
ActualVelocity
LeftServo
ActualVelocity
006
write_line
MOVE
EN ENO
ActualPosition
Data.File[array_index].Position
MOVE
EN ENO
ActualVelocity
Data.File[array_index].Speed
line_written
007
SendMetrics
MOVE
EN ENO
array_index
Data.Records
SampleWriteCSV_2

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:

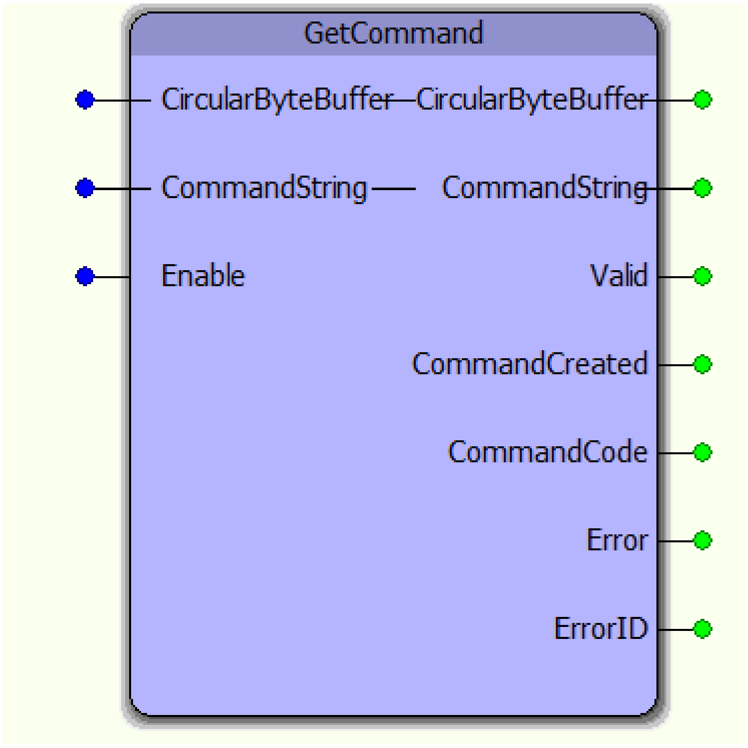


```
(000074)11/12/2012 16:46:02 PM - (not logged in) (192.168.207.205)> Connected, sending welcome message...
(000074)11/12/2012 16:46:02 PM - (not logged in) (192.168.207.205)> 220 Welcome to logan_smith file server - FileZilla Server version 0.9.41 beta
(000074)11/12/2012 16:46:02 PM - (not logged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

Parameters

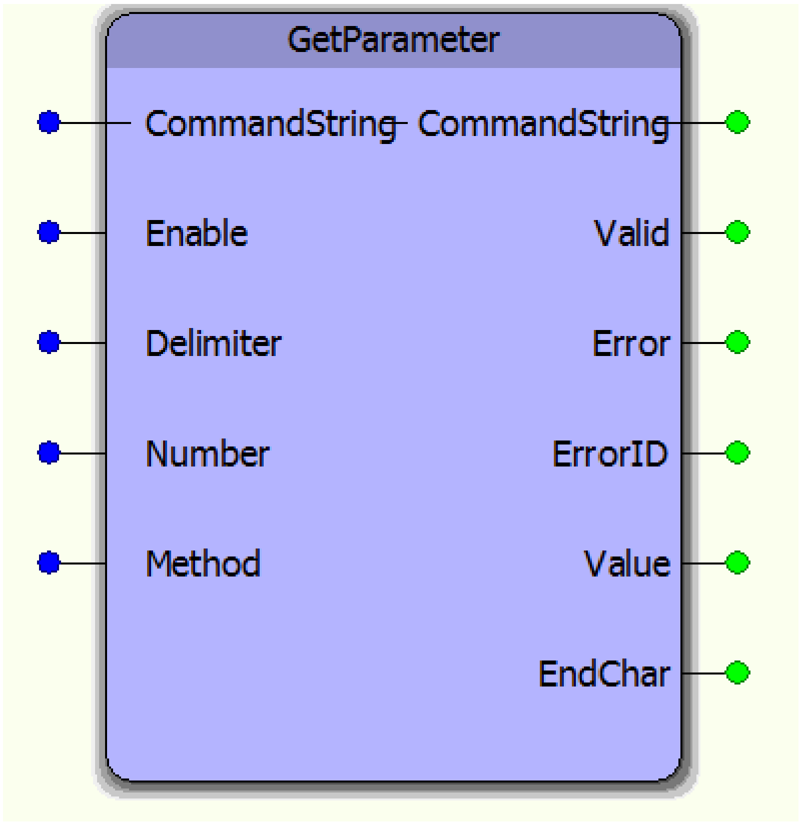
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	CircularByteBuffer	CircularBufferStruct	Structure containing a data buffer and other operational information required to manage the CircularByteBuffer.	
V	CommandString	YTB_STRING512	Input string containing at least two bytes of command characters and any optional parameters separated by a PrmDelimiter.	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	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	CommandCreated	BOOL	Indicates that the CommandString VAR_IN_OUT contains a new CommandString.
B	CommandCode	INT	Integer value corresponding to the first two ASCII characters of the CommandString.
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.
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
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.

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	CommandString	YTB_STRING512	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 while enable is held high.	FALSE
V	Delimiter	YTB_STRING1	String value of the character separating parameters within the CommandString	BYTE#44 (comma - ',')
B	Number	INT	Depending on Method input,	INT#0

			either the number of the parameter value to be found or	
B	Method	Method	ERROR: Variable (ParameterDescription_Method) is undefined.	Method#Parameter
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
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.	
V	Value	STRING	Value of the parameter being searched for	
V	EndChar	INT	Last character position in the CommandString to be searched	

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.

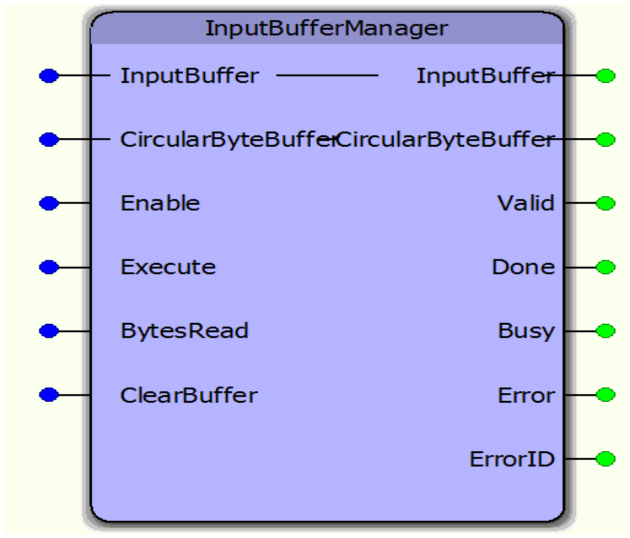
    - Example: CommandString = 'MV,2,4,6' Delimiter = ',' Number = 5
      When Valid = TRUE, Value = 4, EndChar = 7

- 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.

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

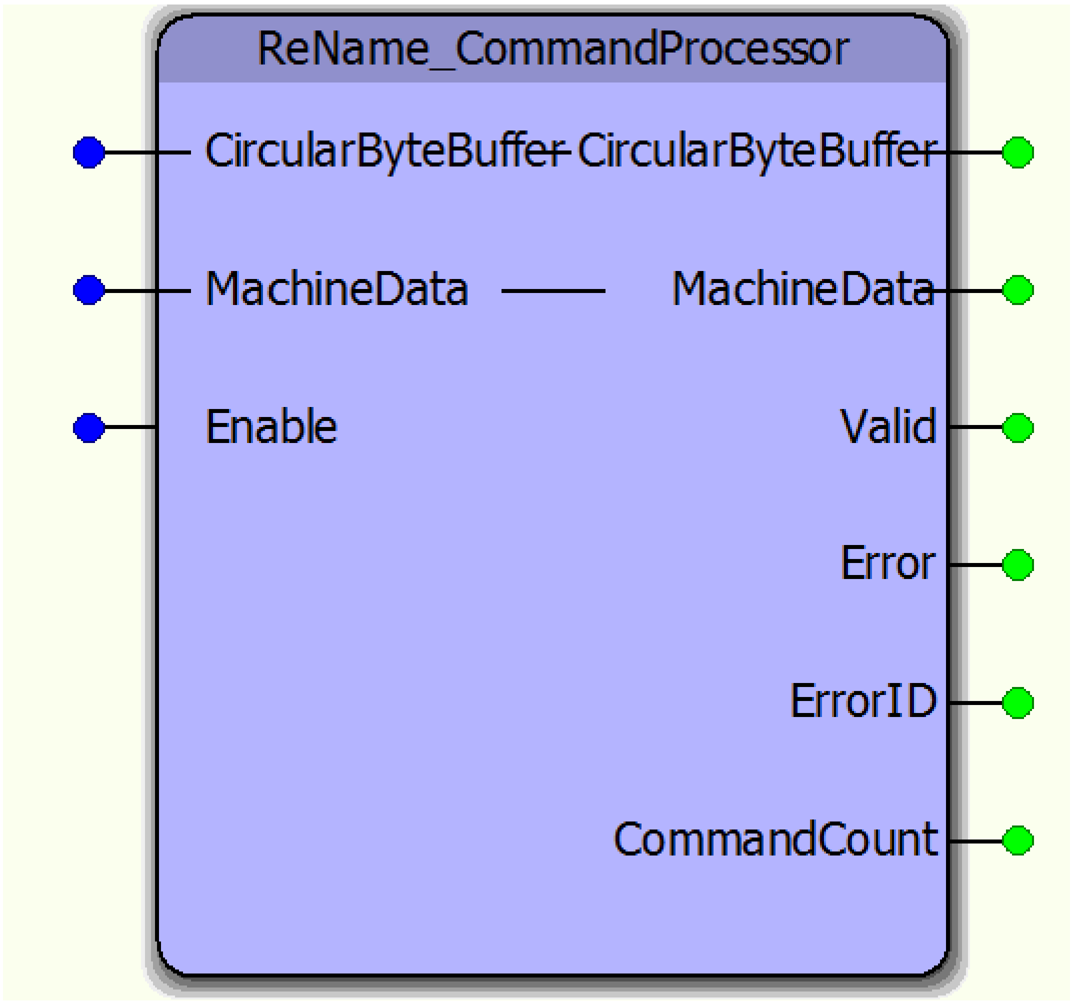
Notes

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.

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	FALSE

			execute while enable is held high.	
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.	
V	CommandCount	UDINT	Number of commands that have been processed since this function block was enabled.	
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.	
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.

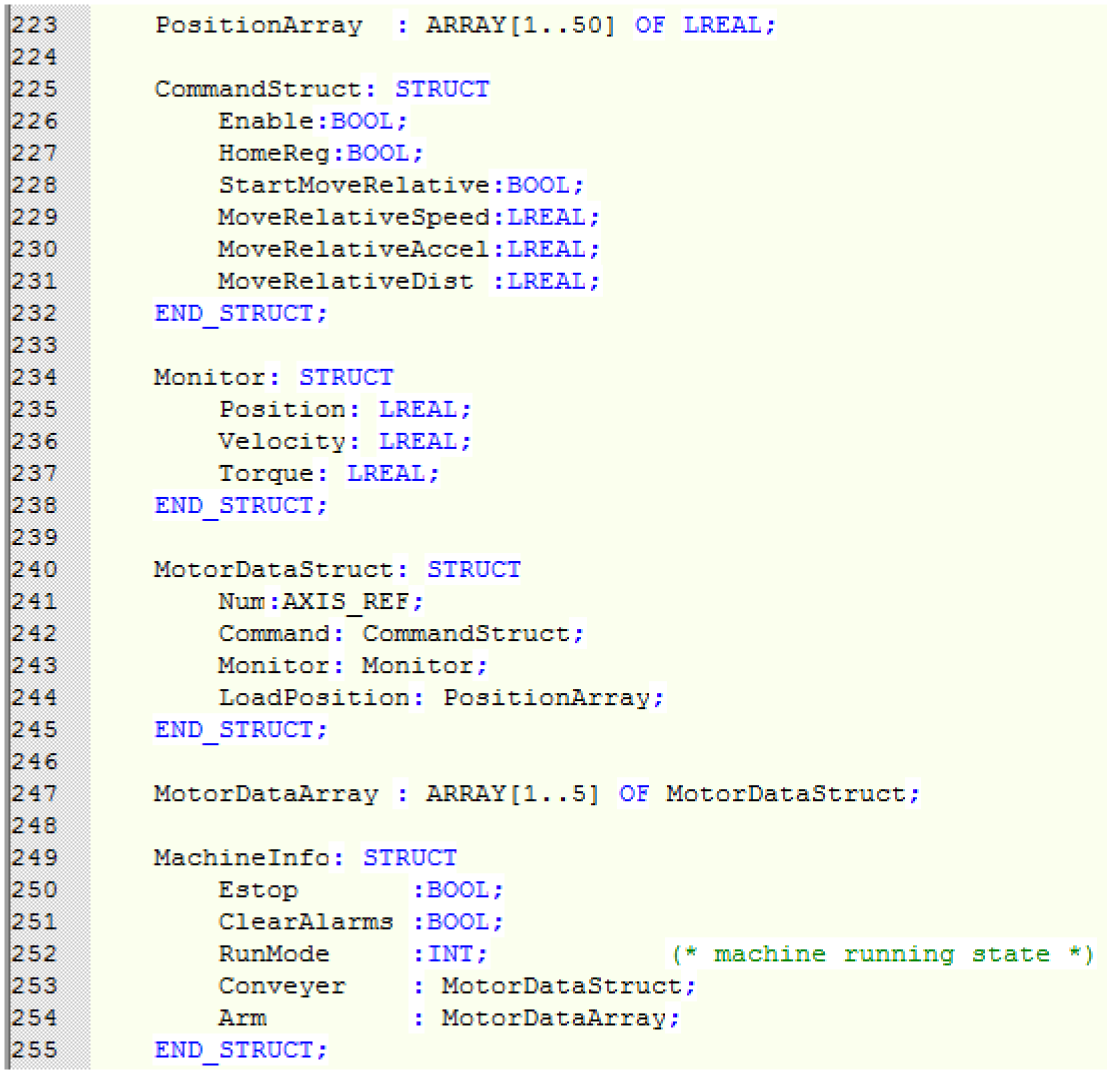
Error Description

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

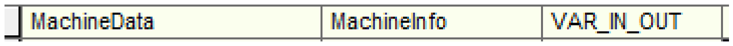
1. Copy this Function block from the Comm Toolbox, paste it into your project, and rename with a different (but similar) name.

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.



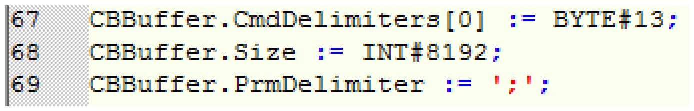
```
    PositionArray  : ARRAY[1..50] OF LREAL;

    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 := ';';
```

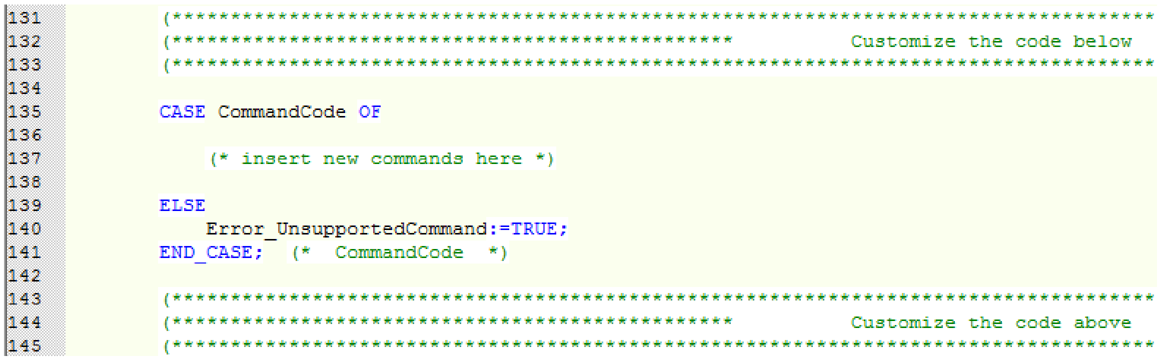
a. a. CmdDelimiters are used to mark the end of a complete command. Up to four characters can be specified. Typically, <cr>, which is BYTE#13 or <cr><lf>, which is BYTE#13 BYTE#10 are used. If CmdDelimiters not specified, will default functionality will automatically accept Carriage Return or Carriage Return & Line Feed.

b. 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).

c. 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:

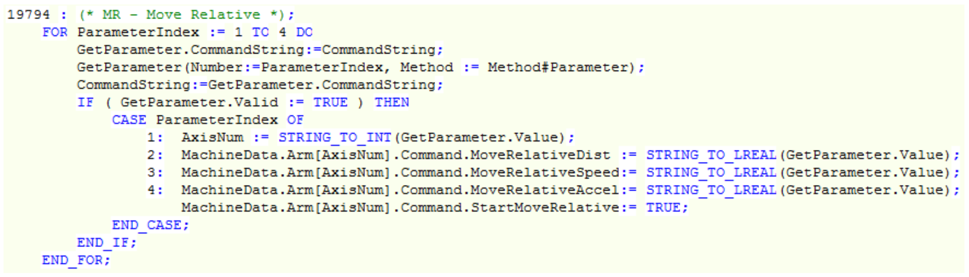
   a. Move Relative command

      a. MR,<axisnumber>,<distance>,<speed>,<accel/decel>

      b. Calculate the CommandCode which corresponds to the ASCII characters 'MR'. The equation is: CHAR_TO_INT('M') * 256 + CHAR_TO_INT('R') = 19794.

      c. Add the CommandCode to the case statement.

      d. 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;
```

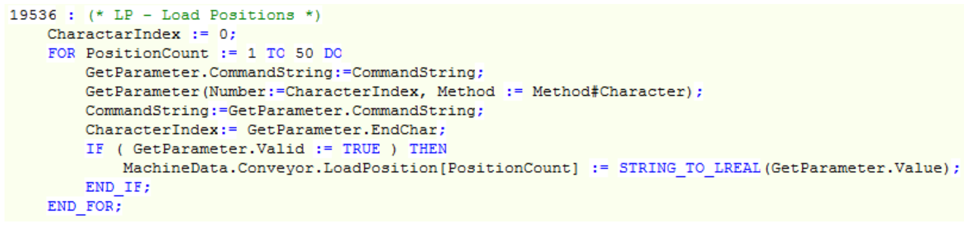
   b. Load Positions command

      a. LP,<Position1>,<Position2>,…,<Position50>

      b. Calculate the CommandCode which corresponds to the ASCII characters 'LP'. The equation is: CHAR_TO_INT('L') * 256 + CHAR_TO_INT('P') = 19536

      c. Add the CommandCode to the case statement.

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

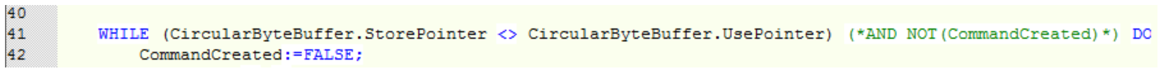


```
19536 : (* LP - Load Positions *)
    CharactarIndex := 0;
    FOR PositionCount := 1 TO 50 DO
        GetParameter.CommandString:=CommandString;
        GetParameter(Number:=CharacterIndex, Method := Method#Character);
        CommandString:=GetParameter.CommandString;
        CharacterIndex:= GetParameter.EndChar;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

none

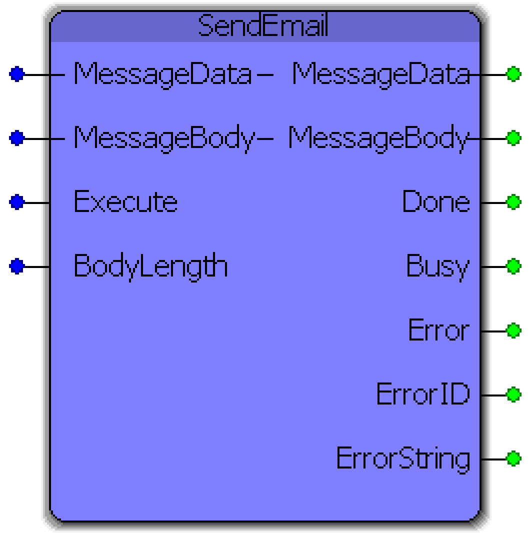
ReName_CommunicationsMgr

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.

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.	
E	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.	
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.
B	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

- This block does 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).

- While "BodyLength" input is optional, it is highly suggested that you pass this variable to the block as it reduces 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 even in more complicated configurations.

Error Description

ErrorID	Meaning
0	No error
8705	The maximum number of concurrently open user IO devices (sockets/files) has been reached.
8706	The socket handle was invalid.
8707	The IP address string was not in a valid format.
8708	The socket 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.
8711	Unable to accept new socket connection.
8712	Unable to bind to the specified address.
8713	The socket type argument was invalid.
8714	The local address or port was not valid.
8715	The socket could not be connected.
8716	There is no network routing path to the specified address.
8717	The socket is already connected to another endpoint.
8718	The socket connection attempt was actively refused by the remote peer.
8719	The socket 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	The Buffer argument to WriteDevice and ReadDevice is required.
8724	The device option ID was invalid.
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 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 unrecognised
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	Transcation failed

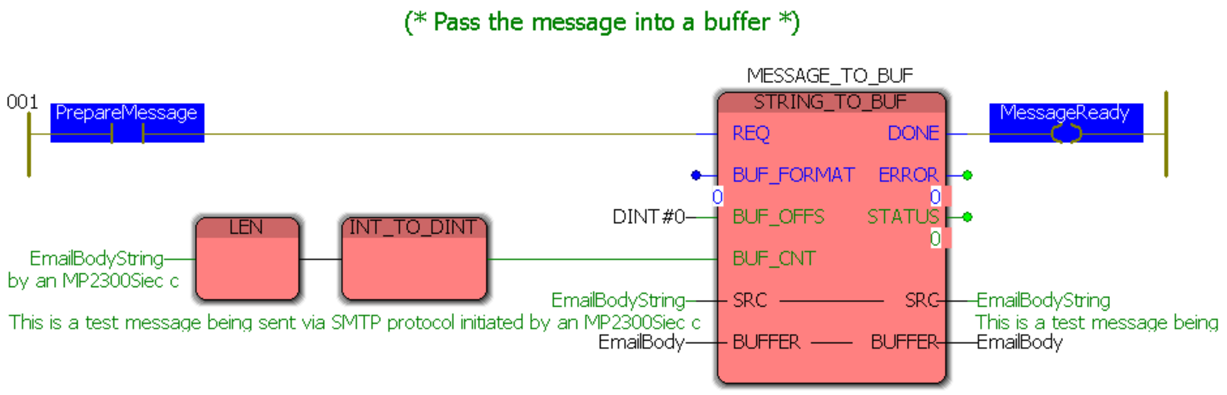
Example

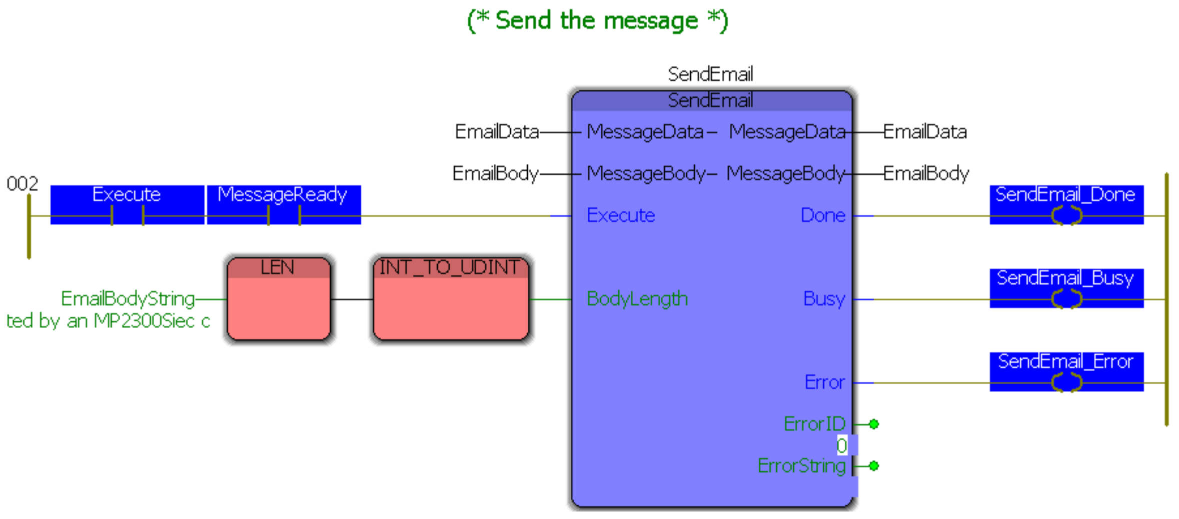
As this is a complicated function, additional examples are provided in separate help files listed under "Additional Examples" and prefixed with "SMTP_". The example shown here sets up the block, creates a message body and sends an e-mail to external Gmail account.

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

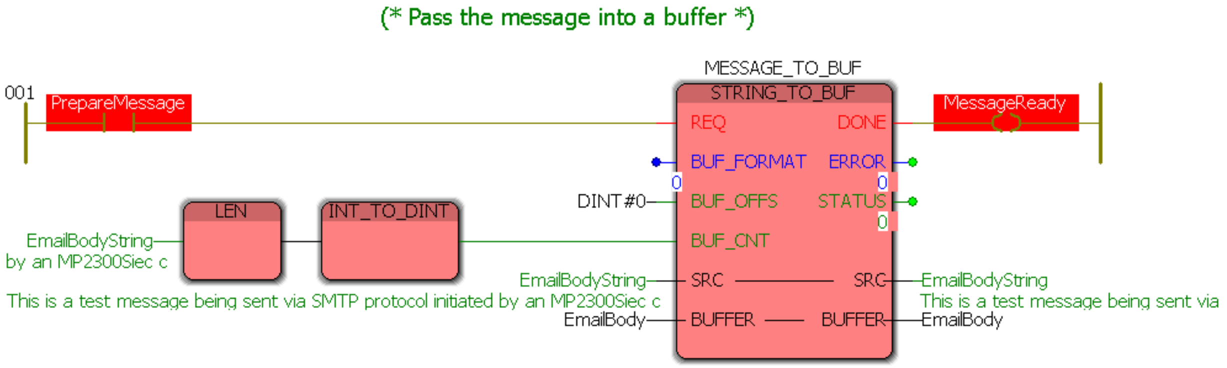
```
(* E-mail Setup *)
EmailBodyString := 'This is a test message being sent via SMTP protocol initiated by an MP2300Siec controller.';

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';
```

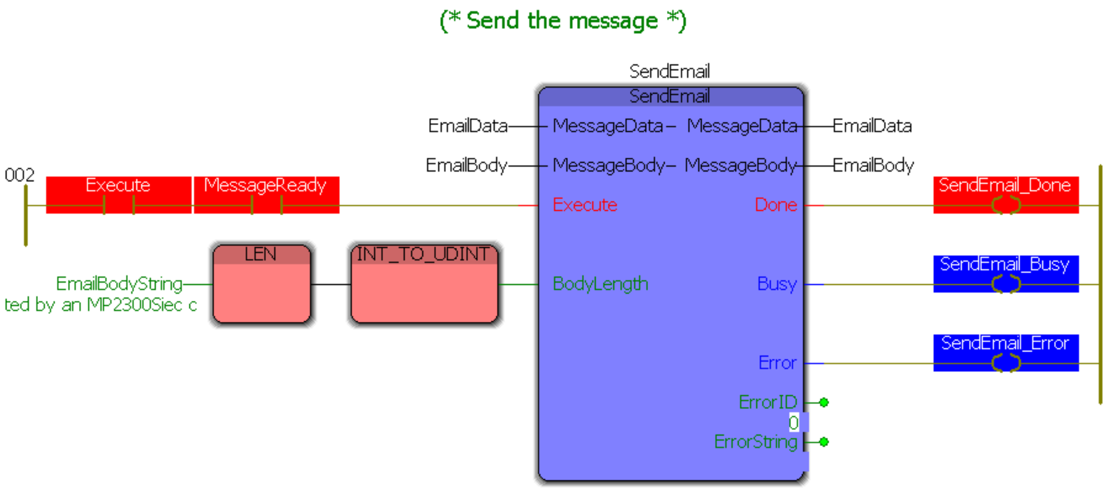
The most basic form of sending an e-mail is simply converting a string to a byte array via the STRING_TO_BUF function block provided in the PROCONOS firmware library. With the data structure shown above and this STRING_TO_BUF block, the email is configured and ready for use.



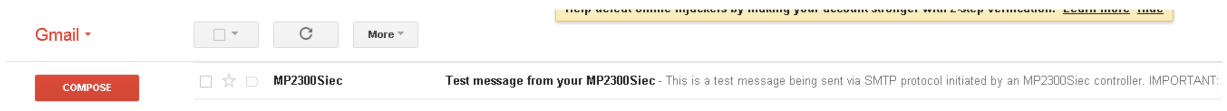


After toggling PrepareMessage, here is the result.



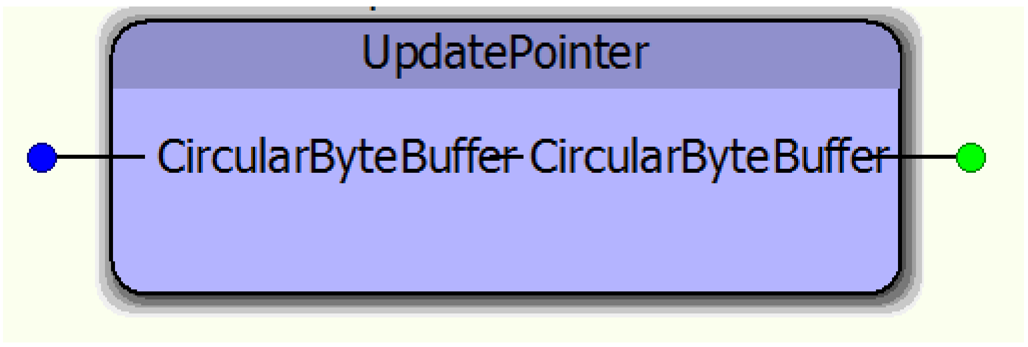


And to demonstrate the end result, here is the e-mail in the inbox of the Gmail account used. The sender and subject are both listed correctly and a portion of the send message can be seen.



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.

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

File_RW Toolbox

The File Read / Write Template is different than the other toolboxes because some of the main 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.

Getting Started: File_RW

Requirements for v202

To use the File_RW Template, your project must also contain the following:

Firmware libraries:

- PROCONOS

User libraries:

- Yaskawa_Toolbox (v204 or higher)

Using the File_RW Template

See Yaskawa's Youtube Webinar - CSV File Transfer with the File_RW Template for more info.

File_RW Revision History

Current Version:

(******************* 2013-09-02 v202 released. Created using 2.4.0 firmware
*******************************)

1) ReadValue - Added "OR (x = DataBuffer.Length)" to cause EOF flag even if the <CR> is not the last byte in a line.

2) Read_CSV_File & Write_CSV_File - Added PreStringError 10017 to detect if the controller already has a String Conversion alarm posted before the function blocks execute.

Previous Versions:

Data Types

Data Type: ByteBufferStruct

Data Type Declaration

```
TYPE

ByteBufferStruct: STRUCT

     Char: ByteArray4096;

     FilePosition:DINT;

     Length:UINT;

END_STRUCT;

END_TYPE;
```

Data Type: MyDataStruct

This datatype MyDataStruct and its two supporting user defined datatypes (MyData and MyDataArray) must copied and pasted into your main project and customized to meet your specific data format.

Rename it in your main project to avoid naming conflicts, which will cause compile errors.

Data Type Declaration

```
TYPE

(************************       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;

END_TYPE;
```

Data Type: SeparatorList

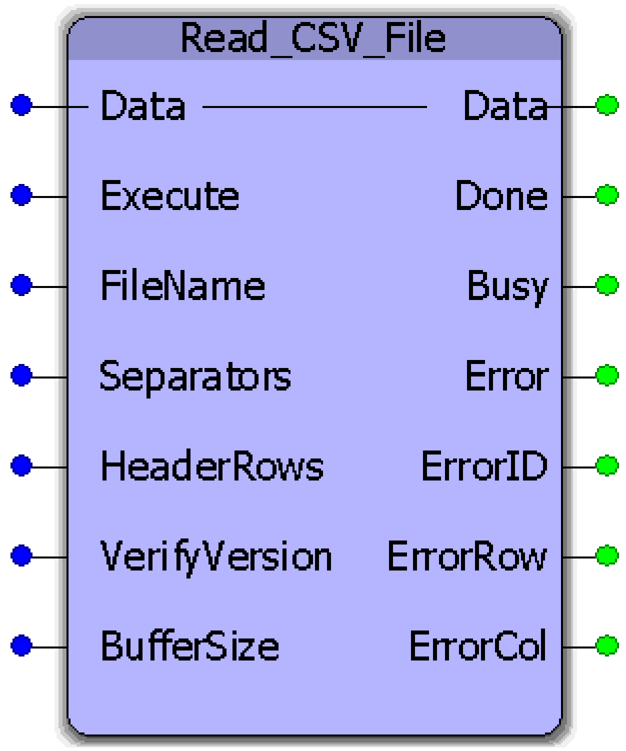
Optional SeparatorList can be populated byte values corresponding to ASCII characters that represent the delimiters between data in columns. For example the TAB character is BYTE#09. If no separators are specified, the function block will default to searching for comma (BYTE#44.)

Data Type Declaration

```
TYPE

SeparatorList:ARRAY[0..4] OF BYTE;  (*  User can select up to four characters that will be used as value
Separators, like comma, semicolon, etc.  *)

END_TYPE;
```

Function Blocks

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.

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#''

			STRING#'flash/user/data/mydata.csv'	
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 been completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
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.)
- 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

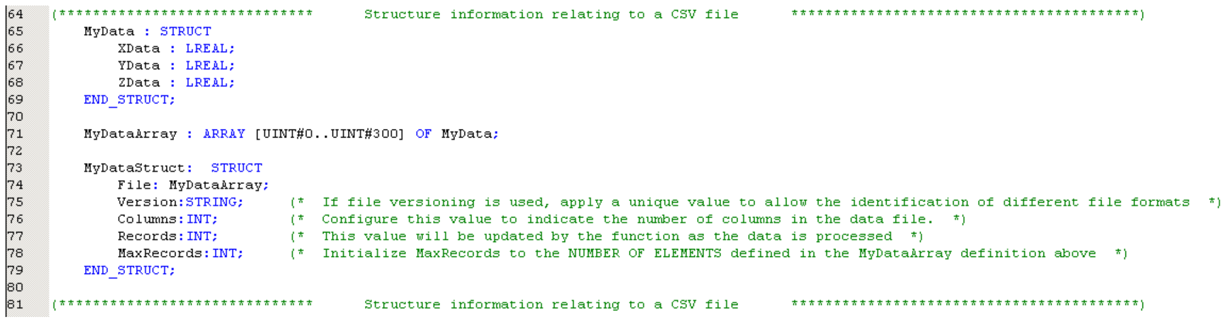
ErrorID	Meaning
0	No error
4	File is already open.
5	File is opened, write protected or access denied.
6	File name not defined.
10	End of data reached.
12	The number of characters to be read is greater than the data buffer.
13	Invalid positioning mode or position specified is before the beginning of the file.
20	File could not be closed.
22	No data could be read.
24	Position could not be set.
10117	String Conversion Error already exists on the controller. Clear the alarm 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.
10121	CSV file contains an unsupported version.
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 outputs for details

10126	NoDataError - The End Of File was reached, but the record count is zero
10127	TooManyRecords - DataType is not large enough
10128	MaxNotDefined - User must set the maximum number of records that can be added to 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

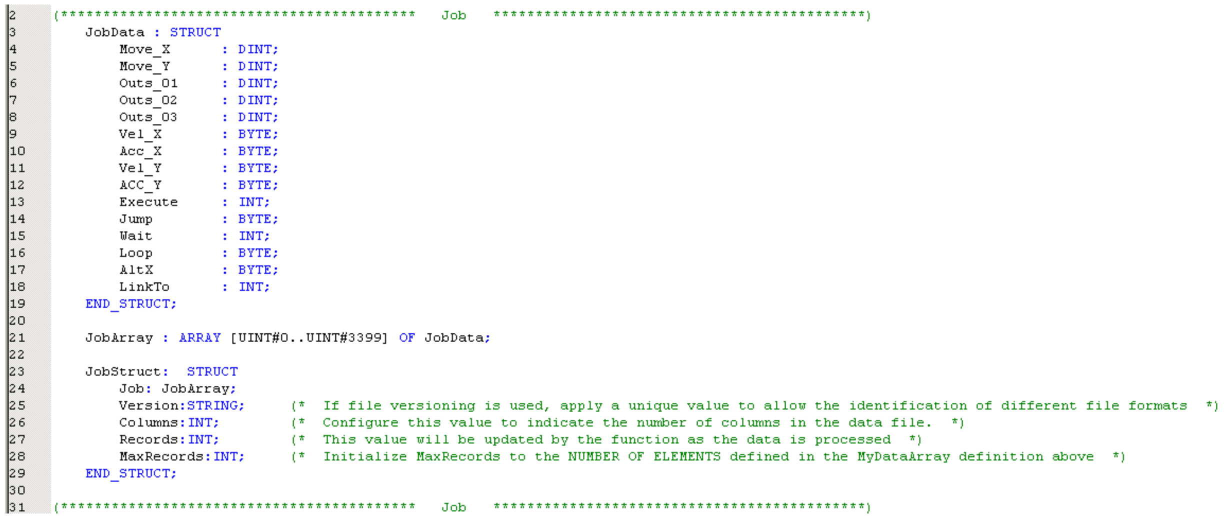
Read_CSV_File must be customized to accommodate your data. Some supporting functions used by Read_CSV_File (ReadBuffer and ReadValue) do not require customization and can remain in the File_RW_Toolbox. To effectively use this function, follow these steps:

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



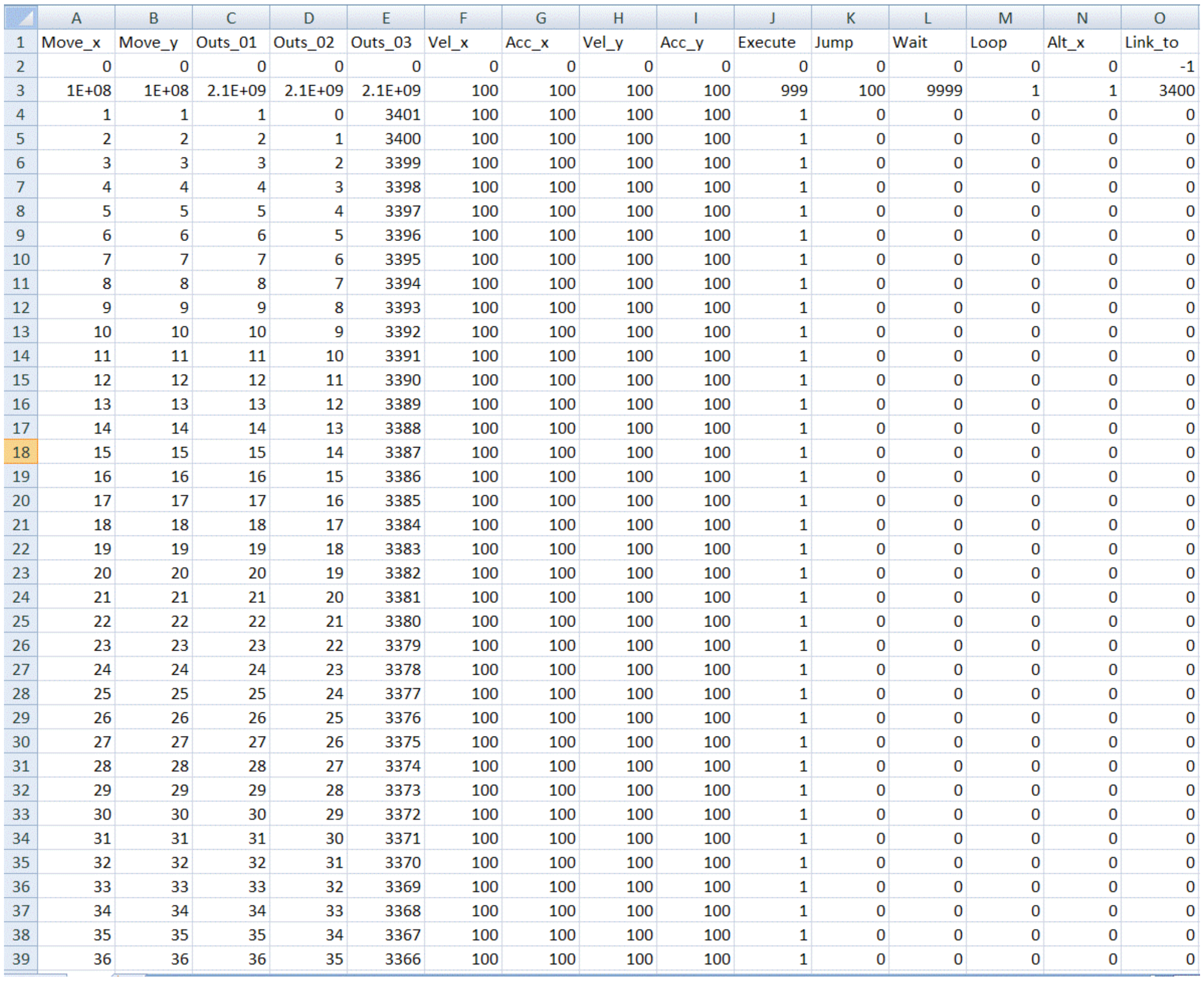
```
(******************************      Structure information relating to a CSV file      *****************************************)
    MyData : STRUCT
        XData : LREAL;
        YData : LREAL;
        ZData : LREAL;
    END_STRUCT;

    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:



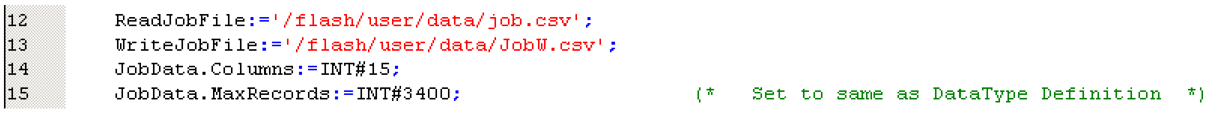
```
(******************************************   Job   ********************************************)
    JobData : STRUCT
        Move_X       : DINT;
        Move_Y       : DINT;
        Outs_01      : DINT;
        Outs_02      : DINT;
        Outs_03      : DINT;
        Vel_X        : BYTE;
        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.



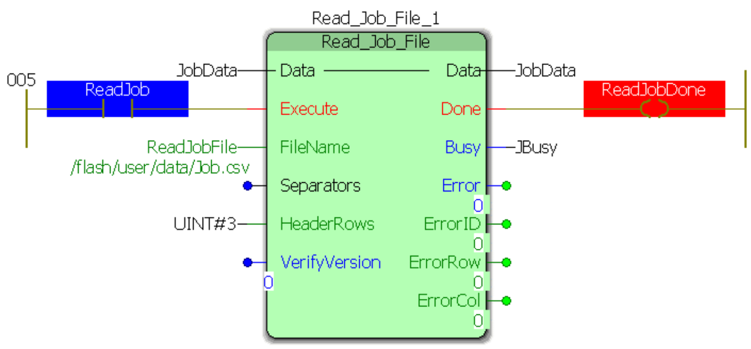
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Move_x	Move_y	Outs_01	Outs_02	Outs_03	Vel_x	Acc_x	Vel_y	Acc_y	Execute	Jump	Wait	Loop	Alt_x	Link_to
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
3	1E+08	1E+08	2.1E+09	2.1E+09	2.1E+09	100	100	100	100	999	100	9999	1	1	3400
4	1	1	1	0	3401	100	100	100	100	1	0	0	0	0	0
5	2	2	2	1	3400	100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

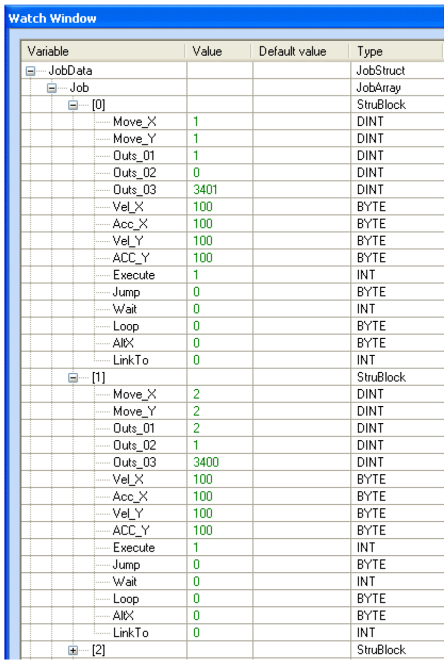
3) Initialize the data required for "MyDataStruct" as shown below. Most importantly, set Columns and MaxRecords.



```
        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.



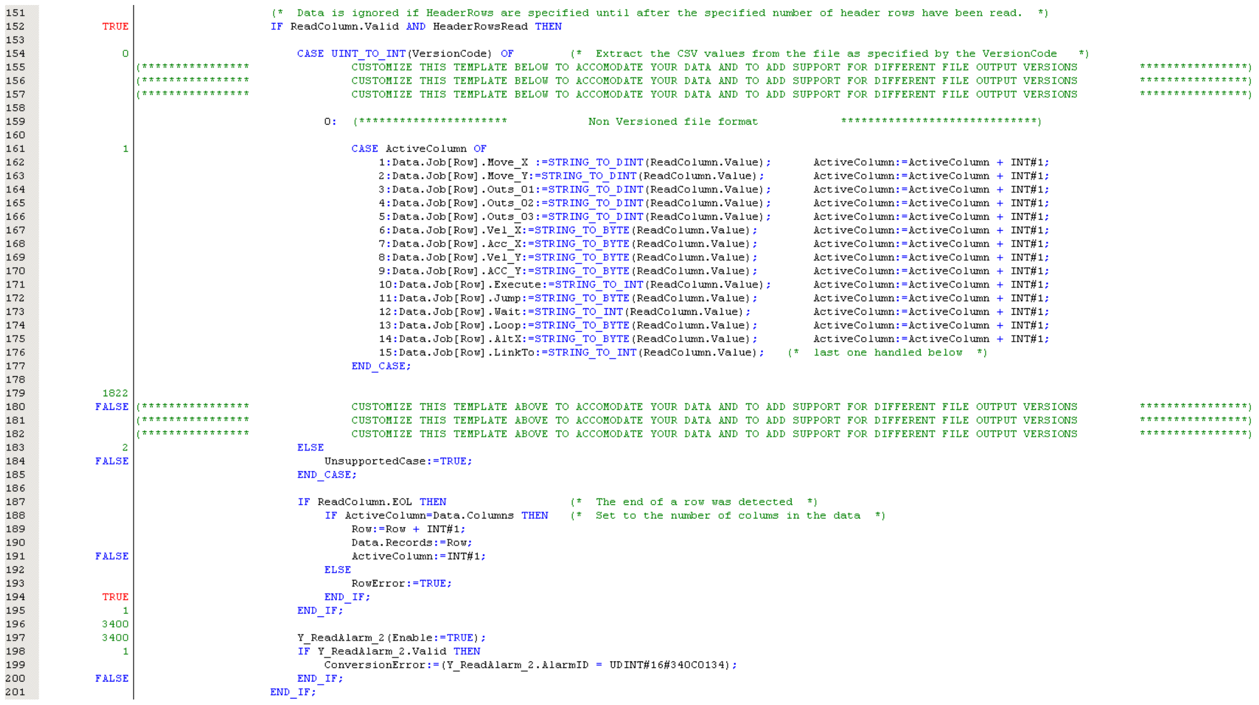

Watch Window

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).

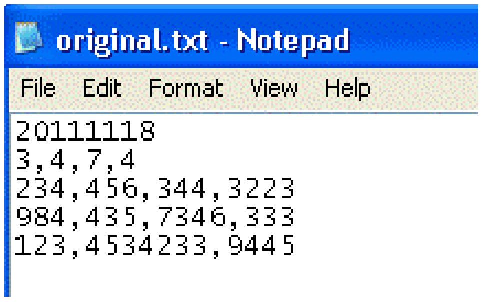
6) Locate the comments near the middle of the Read_CSV_File function indicating the area to be customized. Modify the lines that convert the STRING data from the file into the MyDataStruct structure.


```
                         (*  Data is ignored if HeaderRows are specified until after the specified number of header rows have been read.  *)
                         IF ReadColumn.Valid AND HeaderRowsRead THEN

                             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            ****************************)

                                     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;

(****************                    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;

                             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;
                                     ActiveColumn:=INT#1;
                                 ELSE
                                     RowError:=TRUE;
                                 END_IF;
                             END_IF;

                             Y_ReadAlarm_2(Enable:=TRUE);
                             IF Y_ReadAlarm_2.Valid THEN
                                 ConversionError:=(Y_ReadAlarm_2.AlarmID = UDINT#16#340C0134);
                             END_IF;
                         END_IF;
```

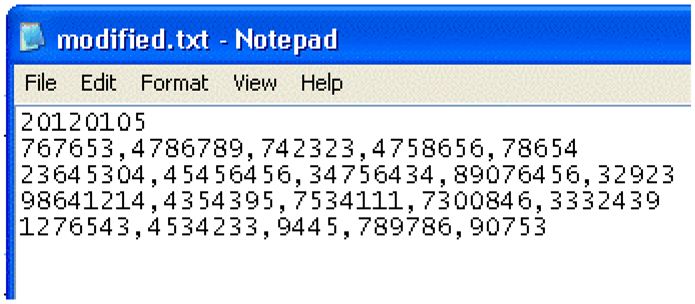
Customizing for file versioning

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
23645304,4545 6456,34756434,89076456,32923
98641214,4354395,7534111,7300846,3332439
1276543,4534233,9445,789786,90753
```

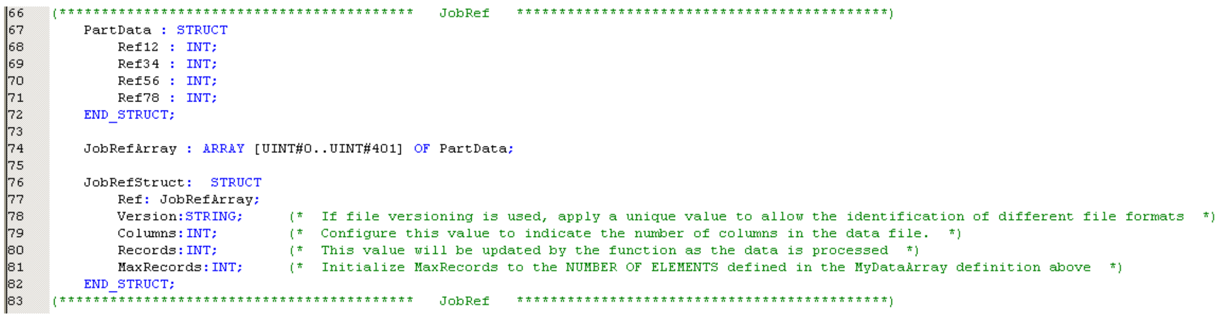
To use file versioning, follow the steps below:

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

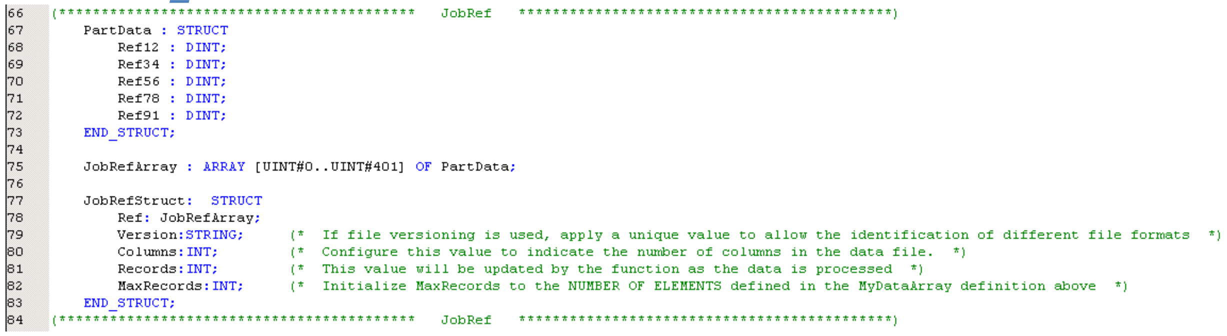
3. Customize the DataType to reflect the most current data specification.

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   ********************************************)
```

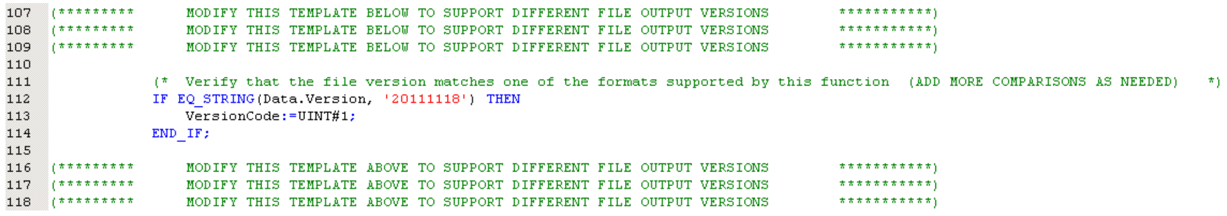
Modified DataType:



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

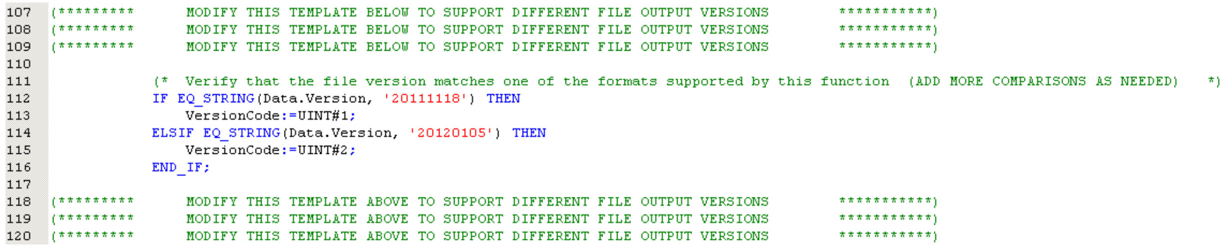
3) Customize the Read-CSV_File function block to determine if the version code detected is supported.

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        ***********)
```

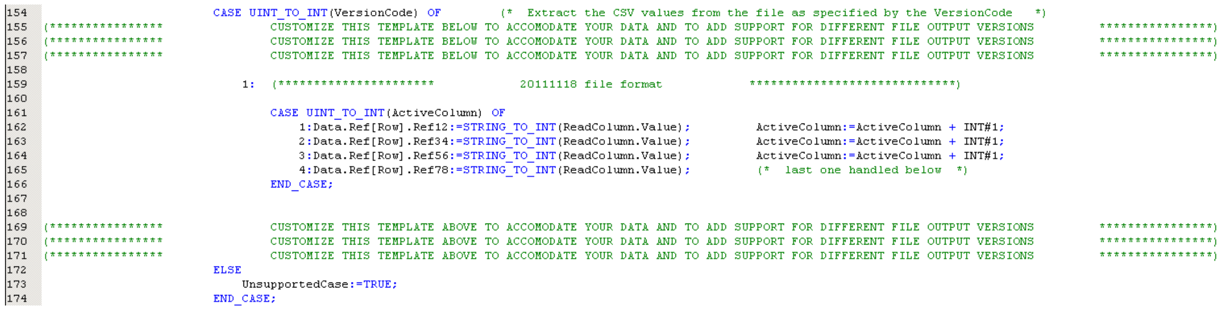
Modified code:



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

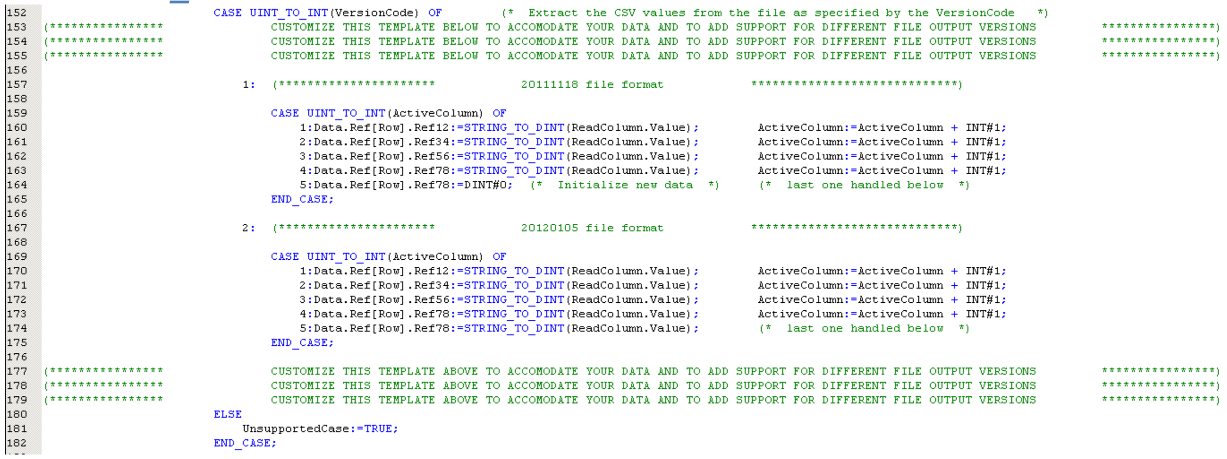
4) Customize the Read_CSV_File function block to read multiple versions.

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;
```

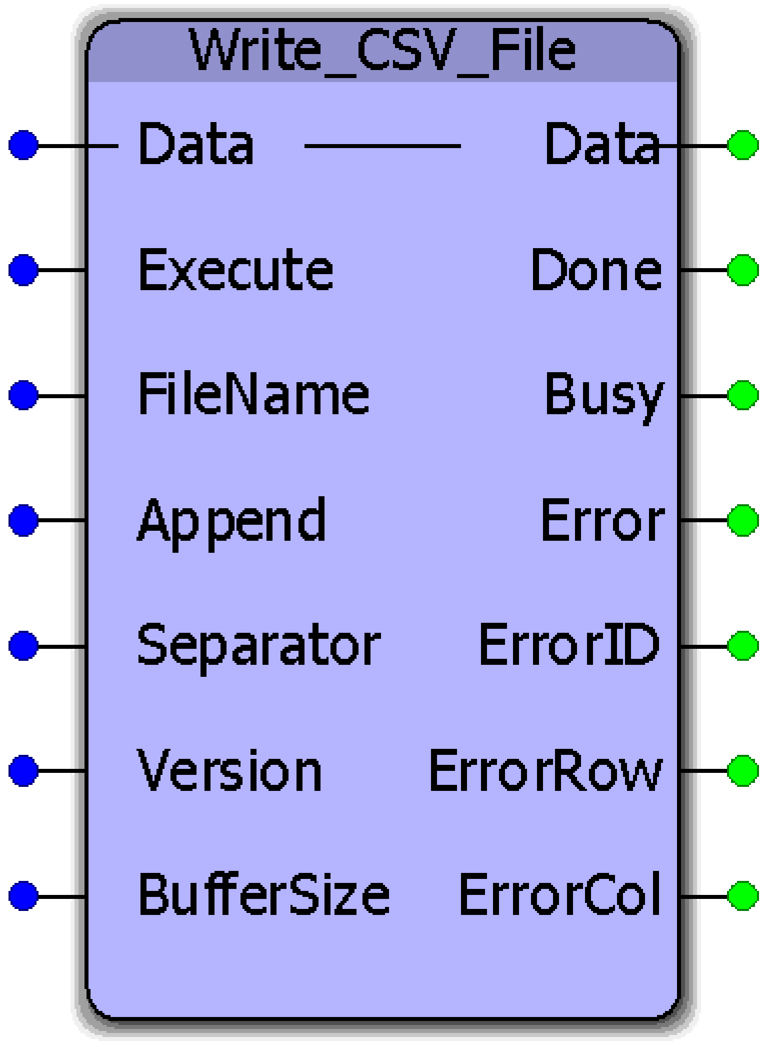
Modified code:



```
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.

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 Append=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 be customized to select between multiple output formats.	UDINT#0
V	BufferSize	UDINT	Specifies the number of bytes in the file to process at one time. If unconnected, the default is 2048 bytes. BufferSize can be adjusted up or down if necessary to accommodate various file sizes and will depend upon the CYCLIC task in which the Read_CSV_File function block is executed.	UDINT#0
VAR_OUTPUT				
B	Done	BOOL	Set high when the slave first synchronizes with the master (Running cam profile is synchronized). This output is reset when execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
E	ErrorID	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

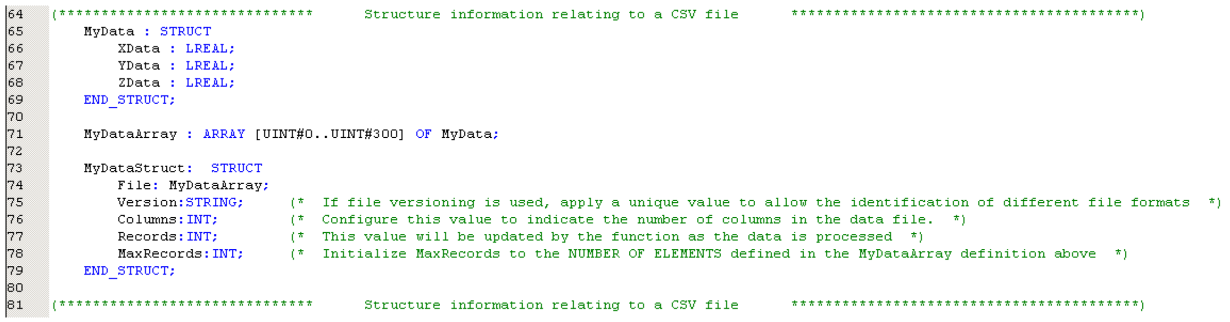
ErrorID	Meaning
0	No error
2	The length of the source buffer does not fit. The size of bytes to be copied assigned in BUF_CNT is larger than the available size of the SRC.
3	The length of the destination buffer does not fit. The sum of the bytes to be copied assigned in BUF_CNT and the offset in the connected byte stream assigned in BUF_OFFS is larger than the size of the connected byte stream.
4	This data type is not supported / File is already open
5	The alignment does not fit to this data type. The size to be copied assigned in BUF_CNT must be divisible by the size of the data type without a rest / File is opened, write protected or access denied
6	The conversion INTEL/MOTOROLA has failed / File name not defined
7	The string length does not fit. Additional checks are necessary for the data type string. This is described in the chapter 'String specialties'.
8	The destination buffer has a wrong data type. In some cases the data type is checked. This is described in the special chapter for each data type.
9	The offset value is not correct. In some cases the offset is checked. This is described in the special chapter for each data type.
10	The BUF_CNT does not fit. In some cases the size to be copied is checked. This is described in the special chapter for each data type.
11	The addresses of the source and the destination are the same / No memory available for writing the data
12	The number of characters to be written is greater than the data buffer
20	File could not be closed
21	File could not be deleted
23	No data could be written
10116	Problem converting string data to the output buffer
10117	String Conversion Error already exists on the controller. Clear the alarm 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.
10121	CSV file contains an unsupported version.
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 outputs for details
10126	NoDataError - The End Of File was reached, but the record count is zero
10127	TooManyRecords - DataType is not large enough
10128	MaxNotDefined - User must set the maximum number of records that can be added to 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.

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      ******************************************)
```

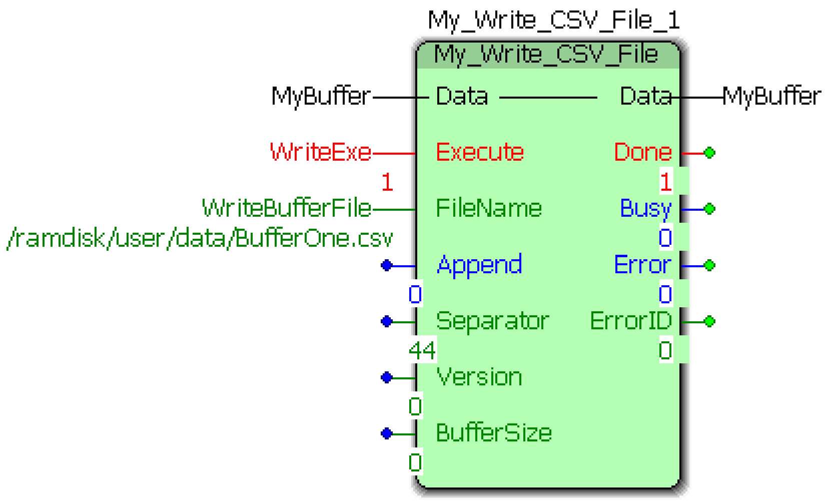
2) Modify the "MyData" dataType definition shown above such that it represents the data to be written. An example follows which shows a customized datatype:

```
(*******************************************   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.



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

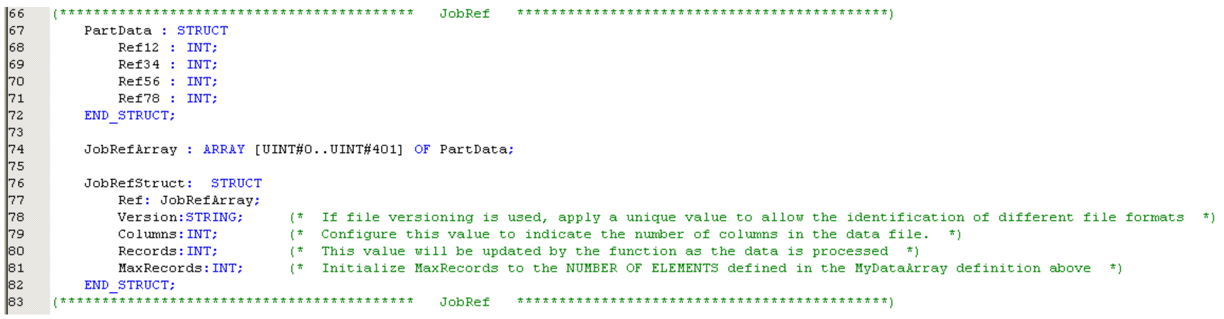
The function has the capability to write multiple versions of the same structure. For example, a portion of the data from the structure can be written to one file, and a different set of data can be written to another file.

To use file versioning, follow the steps below:

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

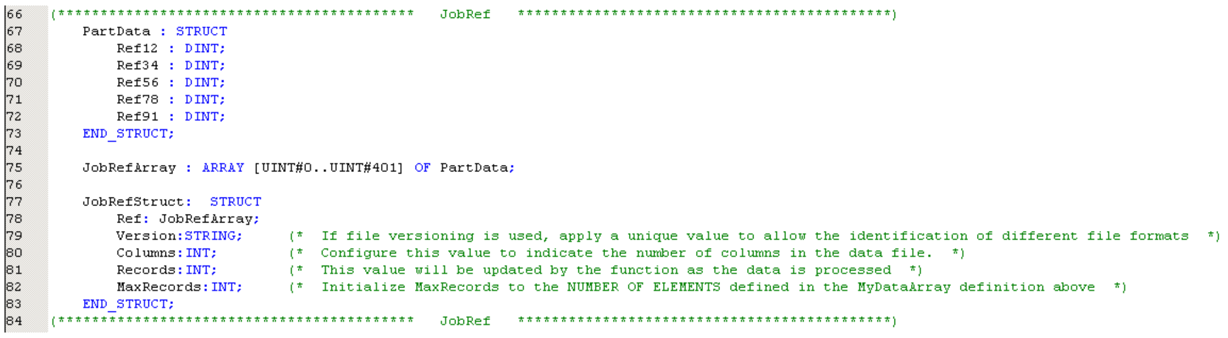
2) Customize the DataType to reflect the most current data specification.

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:

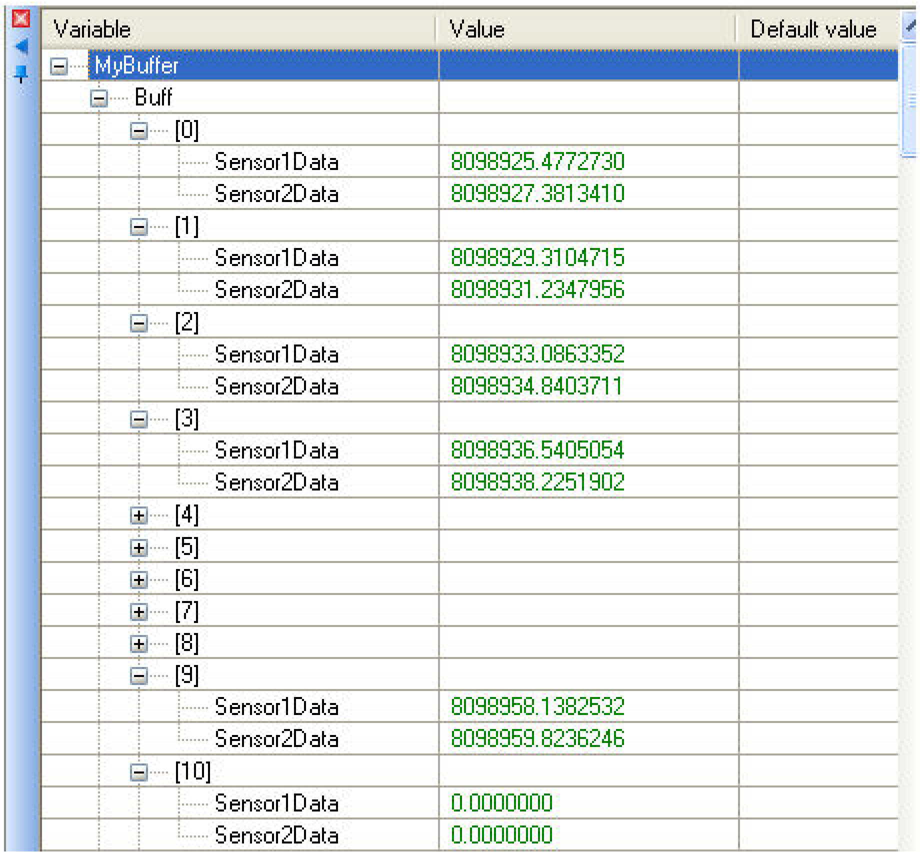
Modified code:

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

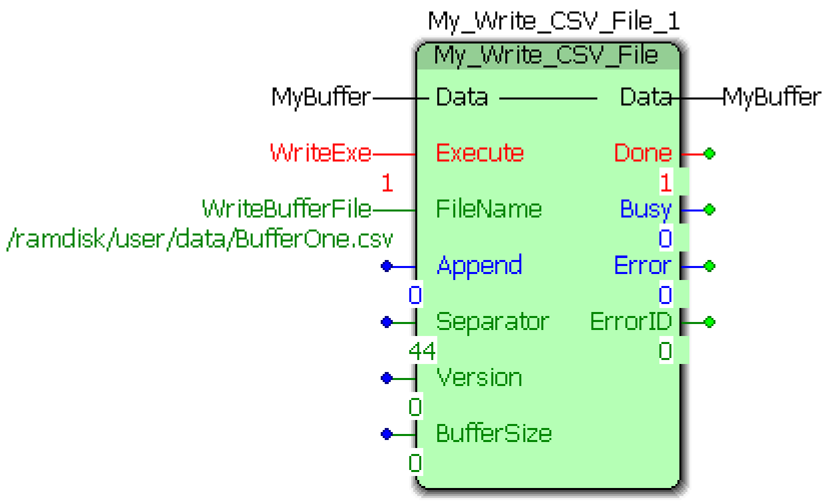
Original code:

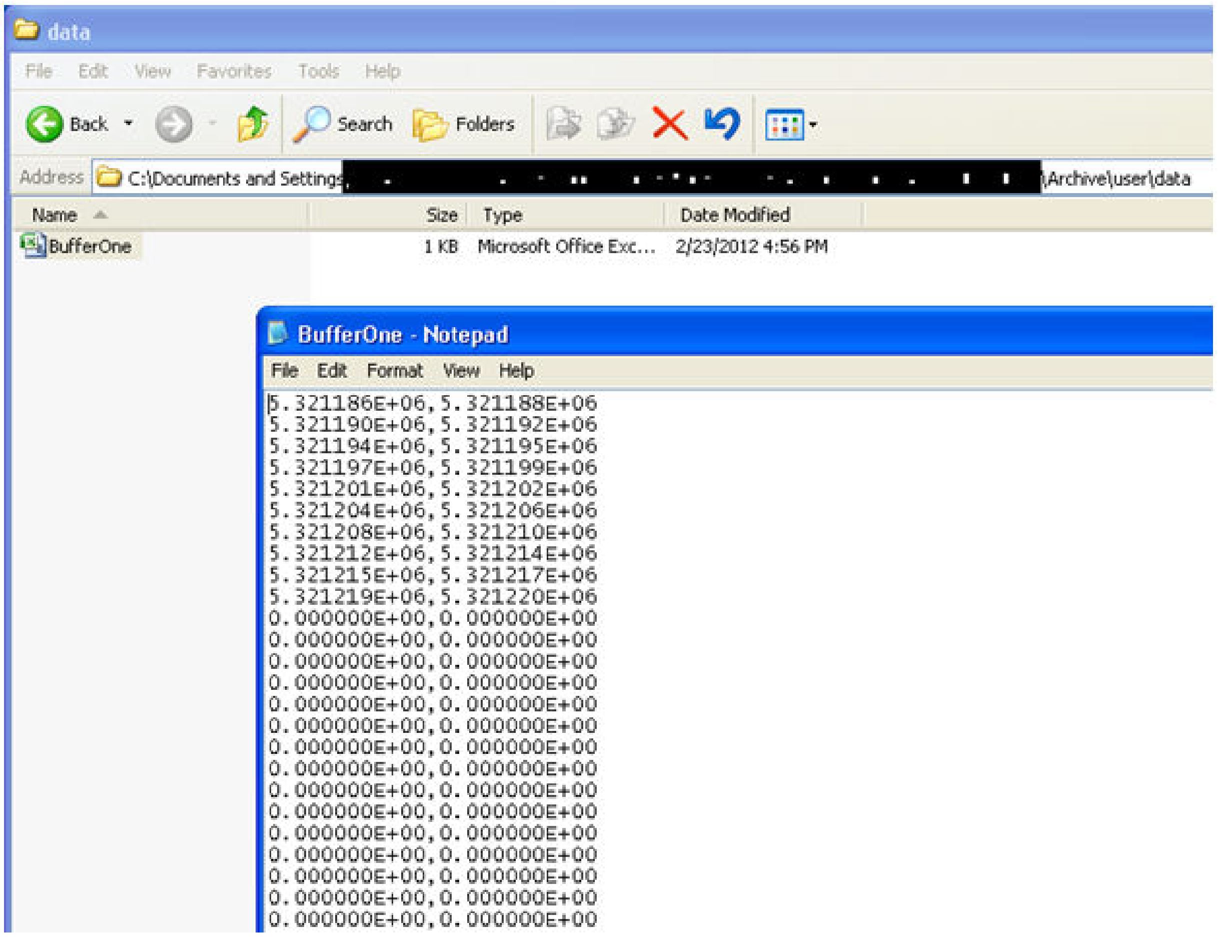
Modified code:

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

Gantry Toolbox

The Gantry Toolbox consists of the following:

Data Types:

Data Type	Usage
DataTypes not used directly with any of the Gantry Toolbox function blocks.	
GantryPositions	Can be used to store absolute positions within the coordinate system.
DataTypes for external use with Gantry Toolbox function blocks	
AXIS_REF	Identifies an axis
GantryStruct	Contains all information pertaining to a gantry system.
PathDetails	For Use with PathGenerator FB
DataTypes that support other DataTypes (no need for direct use by the application programmer)	
PathIdStruct	For use with PathGenerator and MovePath FBs
PathPairs	For use by PathGenerator FB
PathPointArray	For use by PathDetails STRUCT in PathGenerator FB
PathStruct	For Use with PathGenerator FB
SegmentArray	For use with MovePath FB
SegmentDetails	For use with MovePath FB
SegmentStruct	For use with MovePath FB
WPos	Supporting structure for GantryPositions
XPos	Supporting structure for GantryPositions
YPos	Supporting structure for GantryPositions
ZPos	Supporting structure for GantryPositions

Enumerated Types:

Function Block	Description
TB_PatternType	For use with PathDetails structure

Function Blocks:

Function Block	Description
Calculate_Angles	Calculates start and traversed angles for arcs (used as an input to PathGenerator function block)
Gantry_Home	Moves 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.
Gantry_Power	Enables or disable all axes configured as part of a gantry system.
Gantry_Return_Home	Moves all gantry axes back to the home position as defined by the home positions in the GantryStruct.
Gantry_Stop	Executes the MC_Stop block for all axes configured as part of a gantry system.
GotoXY	Performs an absolute move the X and Y axes to a specific location within the gantry coordinate system.
GotoXYZ	Performs an absolute move the X,Y, and Z axes to a specific location within the gantry coordinate system.
GripperControl	Operates a simple gripper device if the actuator can be controlled via a digital output.
Interpolator	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.
Interpolator3D	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.
Move_Path	This function block moves X and Y axes according to a path profile generated by the PathGenerator and specified in the PathStruct structure
PathGenerator	This function block converts straight line vector and arc segment data into cam files, which will provide coordinated motion by using the Move_Path function block
Pick_Part	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.
Place_Part	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.
SegmentLookup	Used to show active segment and output flags status
XY_MoveAbsolute	Used to create absolute motion for an XY gantry system
XY_MoveRelative	Used to create relative motion for an XY gantry system

Getting Started: Gantry

Requirements for v203

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

Firmware libraries:

- YMotion

User libraries:

- DataTypes_Toolbox (v200 or higher)
- Math_Toolbox (v202 or higher)
- PLCopen_Toolbox (v205 or higher)

Using the Gantry Toolbox

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

Gantry Revision History

Current Version:

New for Gantry 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

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

(* Created from Gantry_Toolbox_v203_d_KH

PathPointArray increased to 2047.

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) Gantry_Power - Removed Alarm and Warning outputs.

12) Gantry_Power - Added support for a Tangent axis.

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

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

15) PathGenerator - Added support for a tangent axis

16) PathGenerator - Added support for intermittent motion and pauses

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

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

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

Previous Versions:

Data Types

Data Types for Gantry Toolbox

The following is a complete list of all DataTypes included in the Gantry Toolbox. The list is arranged to separate those that are used internally, and not useful outside of their particular function, and those that an application program must incorporate when the programmer wishes to use the associated Function Block.

Data Type	Usage
DataTypes not used directly with any of the Gantry Toolbox function blocks.	
GantryPositions	Can be used to store absolute positions within the coordinate system.
DataTypes for external use with Gantry Toolbox function blocks	
AXIS_REF	Identifies an axis
GantryStruct	Contains all information pertaining to a gantry system.
PathDetails	For Use with PathGenerator FB
DataTypes that support other DataTypes (no need for direct use by the application programmer)	
PathIdStruct	For use with PathGenerator and MovePath FBs
PathPairs	For use by PathGenerator FB
PathPointArray	For use by PathDetails STRUCT in PathGenerator FB
PathStruct	For Use with PathGenerator FB
SegmentArray	For use with MovePath FB
SegmentDetails	For use with MovePath FB
SegmentStruct	For use with MovePath FB
WPos	Supporting structure for GantryPositions
XPos	Supporting structure for GantryPositions
YPos	Supporting structure for GantryPositions
ZPos	Supporting structure for GantryPositions

Data Type: AXIS_REF

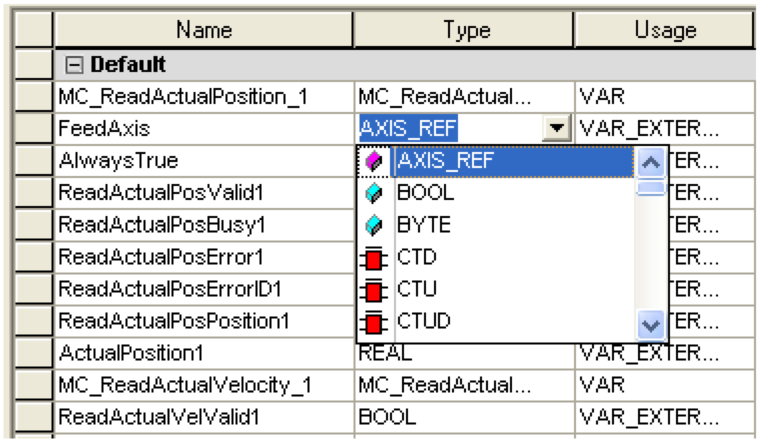
The AXIS_REF data type identifies an axis and thus provides the interface to the hardware or virtual axes. AXIS_REF is used as VAR_IN_OUT in all Motion Control Function Blocks described in this Online help. It is represented as an input and an output connected by a horizontal line in the graphical representation of a function block.

The value of AxisNum is determined by the logical axis number assigned in the Hardware Configuration. See the Configuration tab under each axis.

Data Type Declaration

```
TYPE

AXIS_REF:STRUCT

AxisNum:UINT;

END_STRUCT;

END_TYPE
```

Variable Declaration Example



Name	Type	Usage
⊟ **Default**		
MC_ReadActualPosition_1	MC_ReadActual...	VAR
FeedAxis	AXIS_REF	VAR_EXTER...
AlwaysTrue		ER...
ReadActualPosValid1		ER...
ReadActualPosBusy1		ER...
ReadActualPosError1		ER...
ReadActualPosErrorID1		ER...
ReadActualPosPosition1		ER...
ActualPosition1	REAL	VAR_EXTER...
MC_ReadActualVelocity_1	MC_ReadActual...	VAR
ReadActualVelValid1	BOOL	VAR_EXTER...

Code Example

```
AxisX.Number:=UINT#0;
MCMoveAbsoluteX(Axis:=AxisX, Execute:=FALSE);
AxisX:=MCMoveAbsolutX.Axis;
AxisY.Number:=UINT#0;
```

```
MCMoveAbsoluteY(Axis:=AxisY, Execute:=FALSE);
AxisX:=MCMoveAbsolutY.Axis;
```

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

```
TYPE

GantryPositions: STRUCT (*   Structure of three dimensional locations for positioning a gantry system  *)

X:XPos;

Y:YPos;

Z:ZPos;

W:WPos;

END_STRUCT;

END_TYPE;
```

Data Type: GantryStruct

This datatype contains all information pertaining to a gantry system.

Data Type Declaration

```
TYPE

GantryStruct:STRUCT       (*  DataType to be used in the application code  *)

ID:INT;             (*  Can be used to uniquely identify more than one gantry in a system   *)

Virtual: AxisStruct; (*  All data pertaining to the Virtual axis   *)

X: AxisStruct;      (*  All data pertaining to the X axis   *)

Y: AxisStruct;      (*  All data pertaining to the Y axis   *)

Z: AxisStruct;      (*  All data pertaining to the Z axis   *)

W: AxisStruct;      (*  All data pertaining to the W axis   *)

XPrime: AxisStruct;  (*  All data pertaining to the XPrime axis   *)

YPrime: AxisStruct;  (*  All data pertaining to the YPrime axis   *)

ZPrime: AxisStruct;  (*  All data pertaining to the ZPrime axis   *)

Opened:BOOL;        (*  Gripper status   *)

Closed:BOOL;        (*  Gripper status   *)

OpenCommand:BOOL;   (*  Gripper open request  *)

CloseCommand:BOOL;  (*  Gripper close request  *)

GripperValue:INT;   (*  Constant that equates to the gripper *)

Pick:INT;           (*  Commanded picking location row or column to be used

                    as array index to actual position   *)

Place:INT;          (*  Commanded picking location row or column to be used

                    as array index to actual position   *)

Up:LREAL;           (*  mm Position of the vertical axis when "UP."
                    Alternate usage: ZPosition  *)

Down:LREAL;           (*   mm Position of the vertical axis when "Down."

                    Alternate usage ZPosition  *)

Velocity:LREAL;      (*   Velocity of the gantry workpiece  *)

Accel:LREAL;         (*   Acceleration of the gantry workpiece  *)

Decel:LREAL;         (*   Deceleration of the gantry workpiece  *)

ZVelocityUp:LREAL;   (*   Velocity of the vertical axis  *)

ZVelocityDown:LREAL; (*   Velocity of the vertical axis  *)

ZAccel:LREAL;        (*   Acceleration of the vertical axis  *)

ZDecel:LREAL;        (*   Deceleration of the vertical axis  *)

END_STRUCT;

END_TYPE
```

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

```
TYPE

    PathIDStruct:STRUCT

     XAxisTable:UINT;               (*  The CamTableID for the X axis   *)

     YAxisTable:UINT;               (*  The CamTableID for the Y axis   *)

     PathLength:LREAL;               (*  The total length of the path motion of the

                                toolpoint, the distance the virtual master will

                                travel to complete the path.  *)

   END_STRUCT;

END_TYPE
```

Data Type: PathPairs

For use with the PathGenerator Function Block

Data Type Declaration

```
PathPairs: ARRAY[0..1024] OF UDINT;  (*  For use internally by the PathGenerator FB  *)
```

Data Type: PathPointArray

For use with the PathGenerator Function Block

Data Type Declaration

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

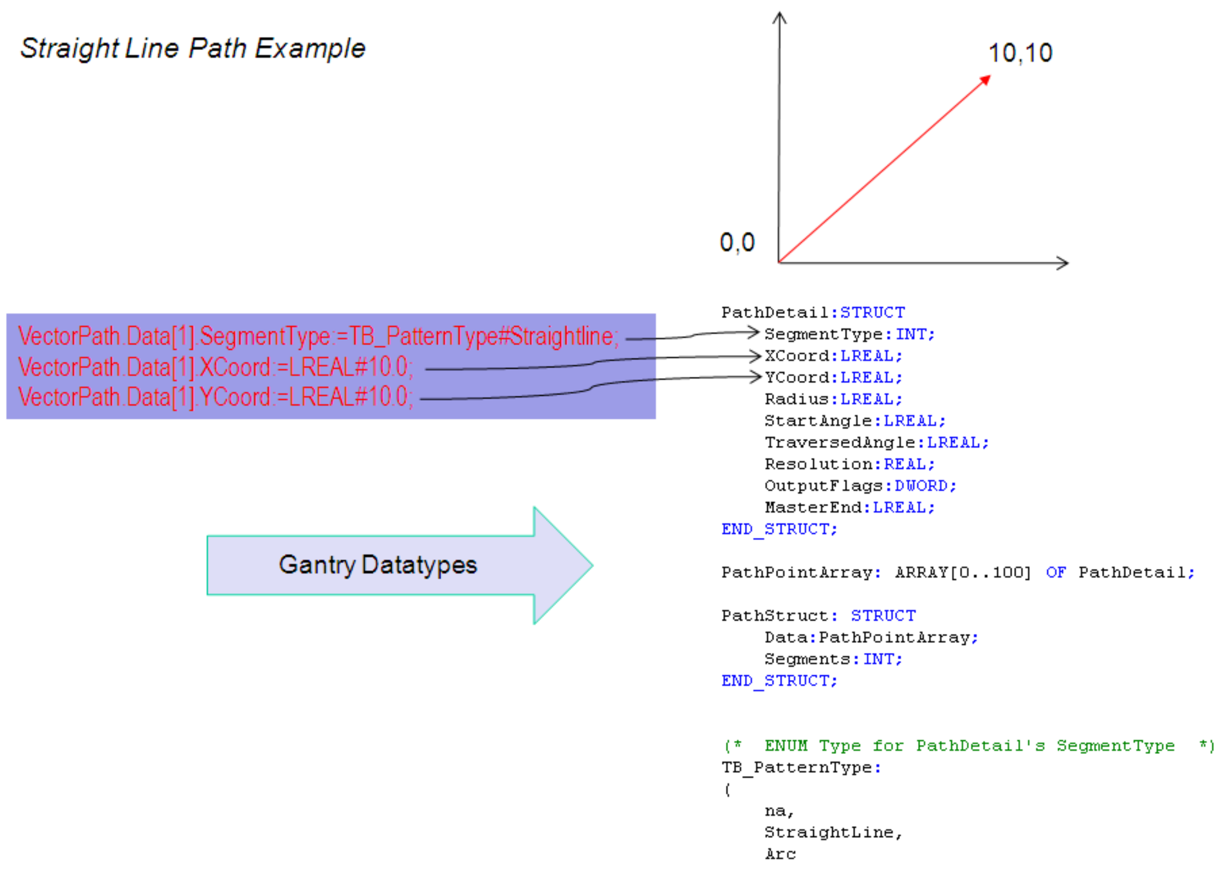
Data Type: PathStruct

For use with the PathGenerator Function Block

Data Type Declaration

```
PathStruct: STRUCT      (*  Data structure used with the PathGenerator function block  *)

    Data:PathPointArray;

    Segments:INT;       (*  Total datapoints specified in the path.  If you need more than defined in the
PathPointArray, just increase  *)

END_STRUCT;
```

PathStruct Example 1

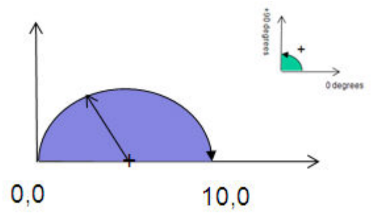


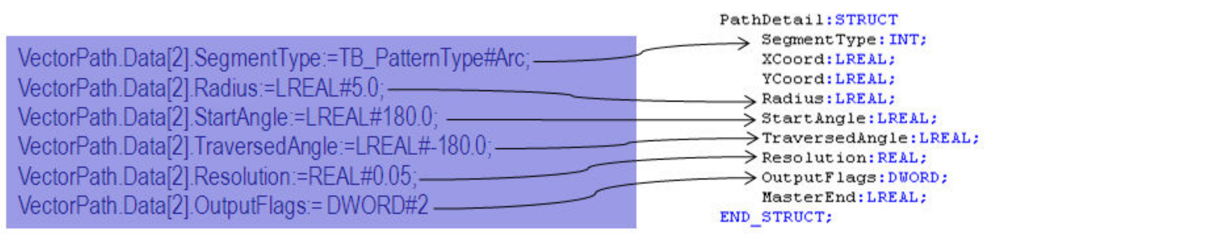
```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#10.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
```

```
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

Arc Path Example





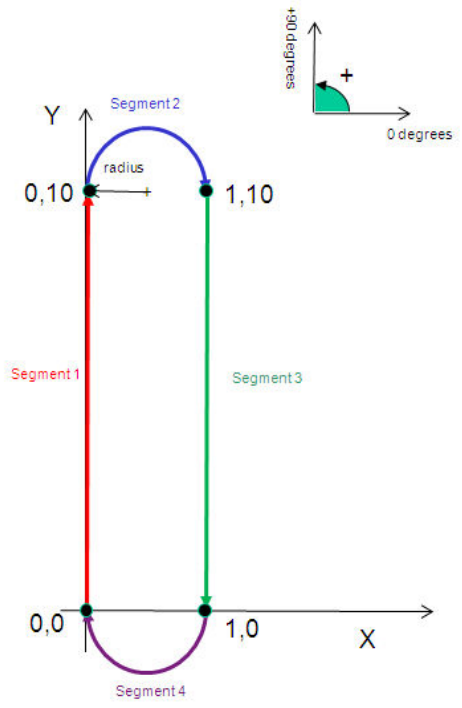
```
VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[2].Radius:=LREAL#5.0;
VectorPath.Data[2].StartAngle:=LREAL#180.0;
VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[2].Resolution:=REAL#0.05;
VectorPath.Data[2].OutputFlags:= DWORD#2
```

```
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

```
TYPE

    SegmentStruct: STRUCT

        Segment: SegmentArray;

        LastSegment: INT;

    END_STRUCT;

END_TYPE
```

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

Enumerated Type: TB_PatternType

ENUM Type for PathDetails' SegmentType

Data Type Declaration

```
TB_PatternType:

(

na,          (*  INT#0  - Not a valid PatternType  *)

StraightLine, (*  INT#1  - Straight Line *)

Arc          (*  INT#2  - Arc *)

);
```

Enumerated Type: TB_PatternType

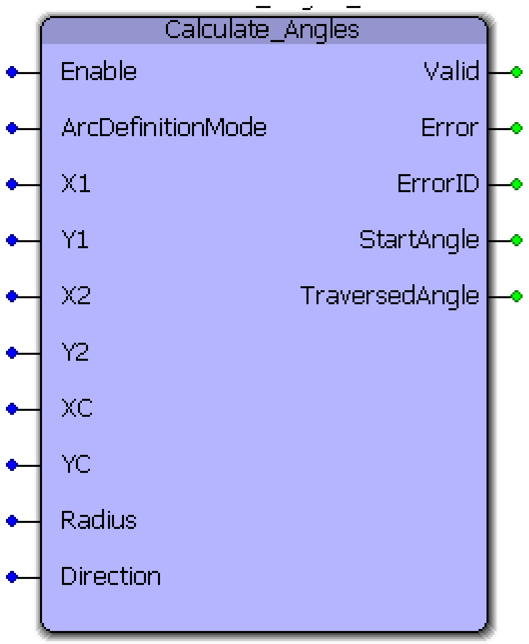
ENUM Type for PathDetails' SegmentType

Data Type Declaration

```
TB_PatternType:

(

na,          (*  INT#0  - Not a valid PatternType  *)

StraightLine, (*  INT#1  - Straight Line *)

Arc          (*  INT#2  - Arc *)

);
```

Function Blocks

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

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute while enable is held high.	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	0
B	X1	LREAL	X coordinate of the first coordinate	0.0
B	Y1	LREAL	Y coordinate of the first coordinate	0.0
B	X2	LREAL	X coordinate of the second coordinate	0.0
B	Y2	LREAL	Y coordinate of the second coordinate	0.0
B	XC	LREAL	X coordinate of the center coordinate	0.0
B	YC	LREAL	Y coordinate of the center coordinate	0.0

B	Radius	LREAL	Radius of arc	0.0
B	Direction	MC_Direction	0: clockwise, 1: counter clockwise	0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
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.	
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

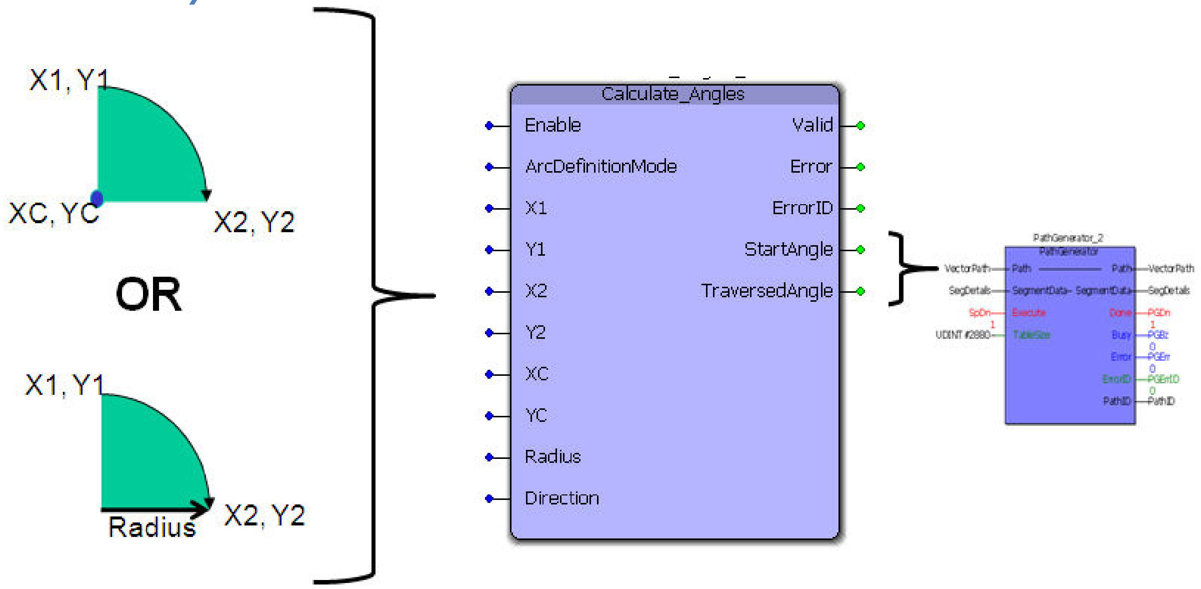
- See Yaskawa's Youtube channel for more info, details, and examples.

Error Description

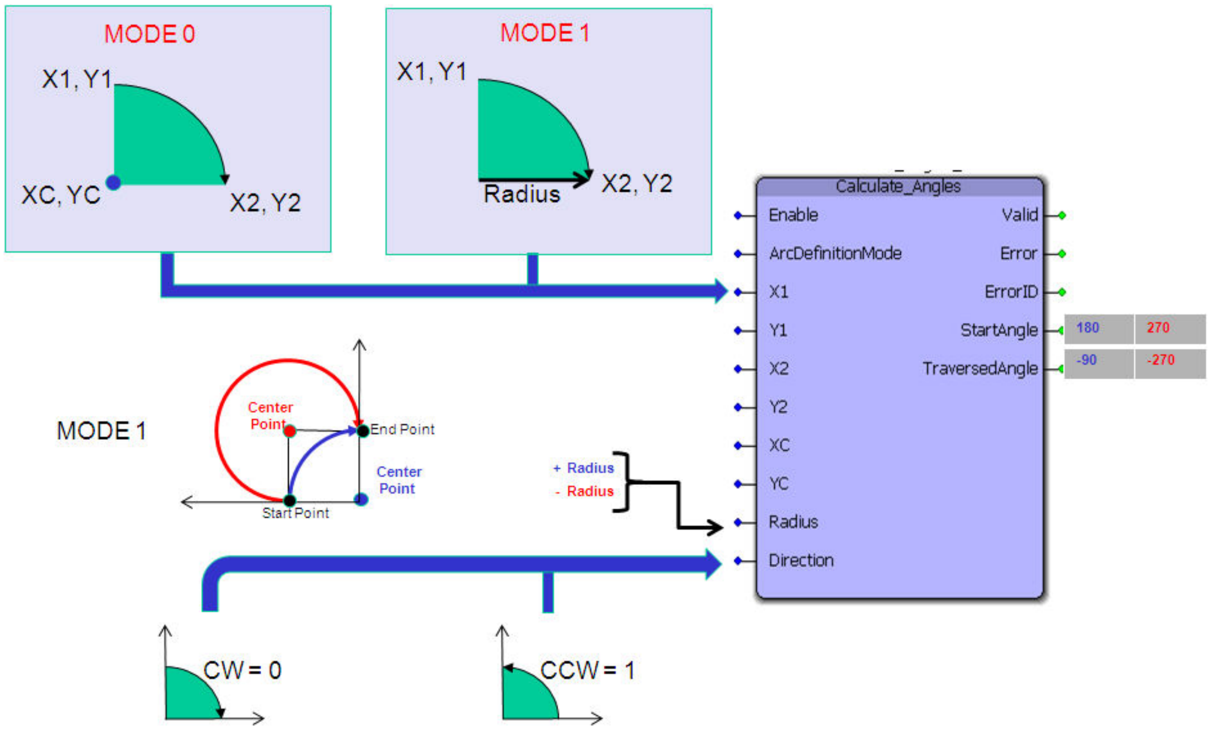
ErrorID	Meaning
0	No error
10130	The center to co-ordinate distance for the two input co-ordinates are not the same
10131	Zero radius is invalid
10132	Only modes 0 (center + 2 co-ordinates) and 1 (radius + 2 coordinates) are supported
10133	The coordinates of the two data points are the same
10140	Must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



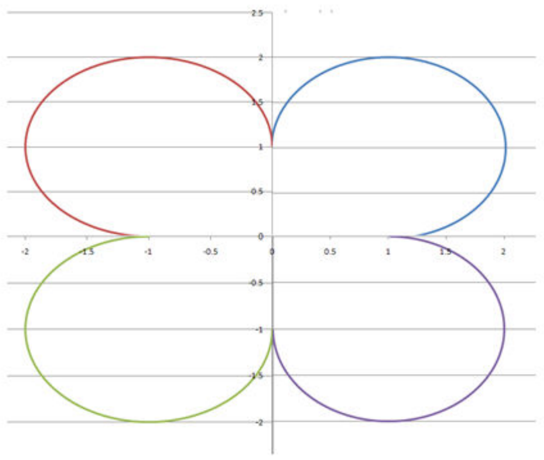
The two modes of data entry are shown in detail below. Mode 0: 2 coordinates + center coordinate, Mode 1: 2 coordinates + radius. If the user plans to use Mode 1, the sign of the radius is important. this is illustrated in the figure below. The two arcs (red and blue) have the same start and end coordinates 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

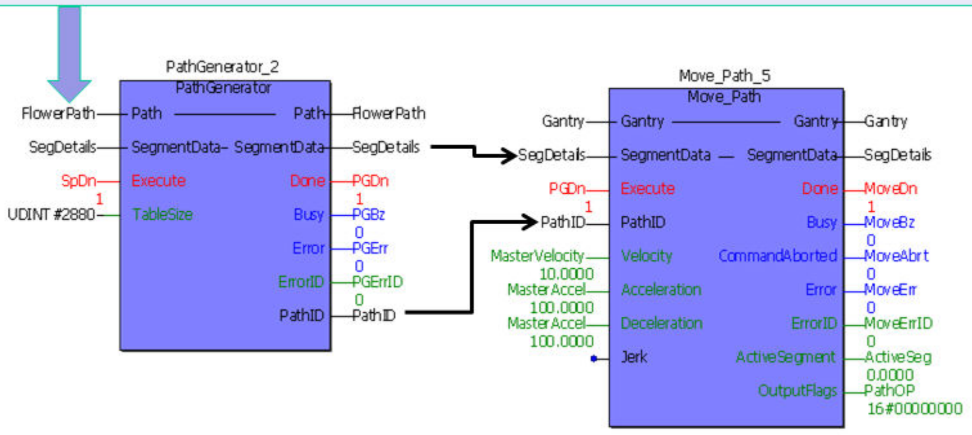
Step1: Using Calculate_Angles to calculate start and traverse angles for the flower path shown below

```
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);
```

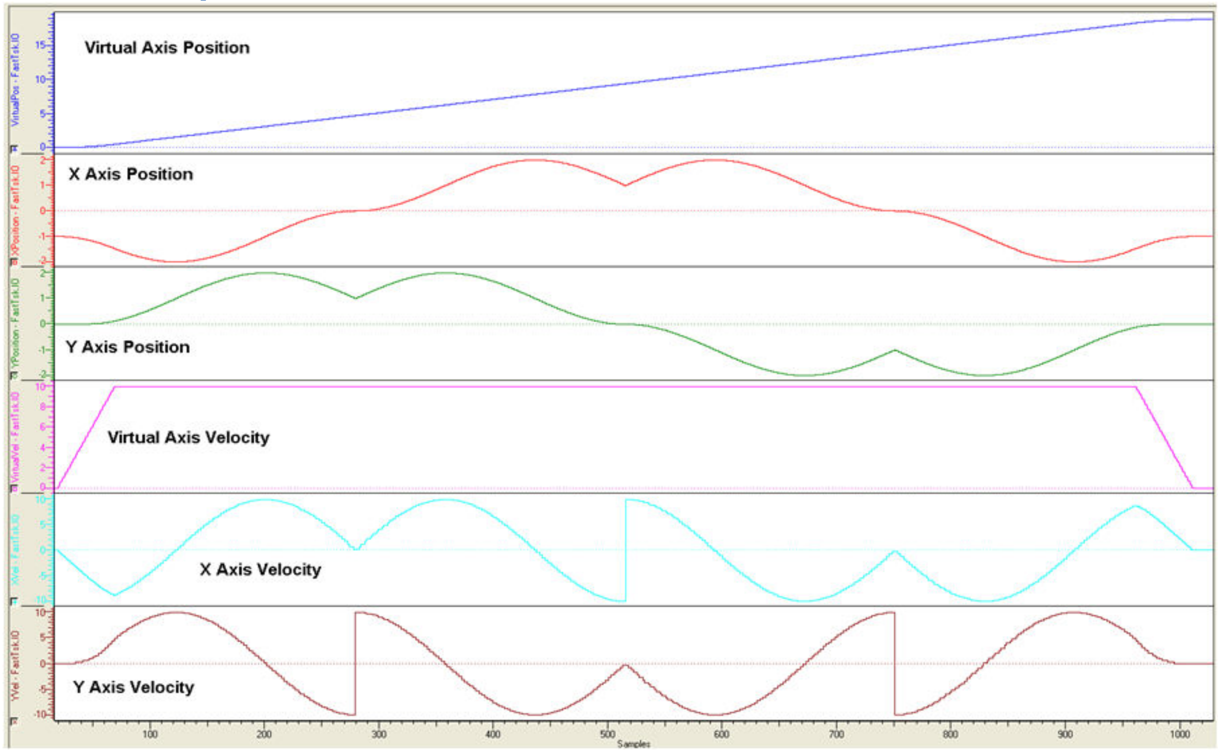


Step 2: Use PathGenerator to create the path and Move_Path to implement XY motion

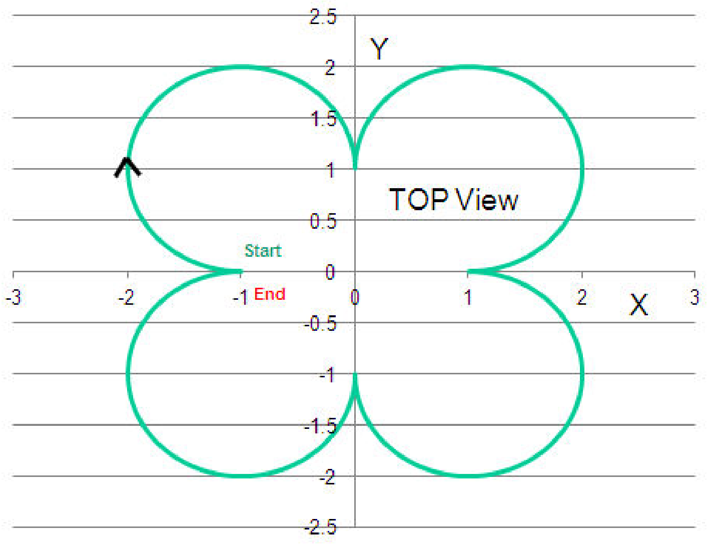
```
FlowerPath.Data[1].SegmentType := TB_PatternType#Arc;
FlowerPath.Data[1].Radius:=LREAL#1.0;
Calculate_Angles_1(Execute:=TRUE,ArcDefinitionMode := INT#1, X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);
FlowerPath.Data[1].StartAngle :=Calculate_Angles_1.StartAngle;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



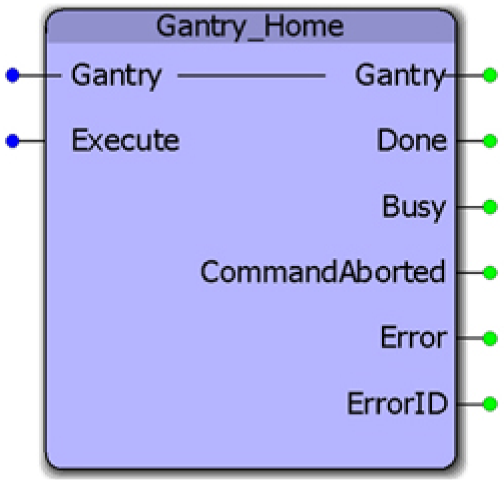
Step 3: Validation using logic analyzer



Step 4: Result on XY system



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.

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 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	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 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.

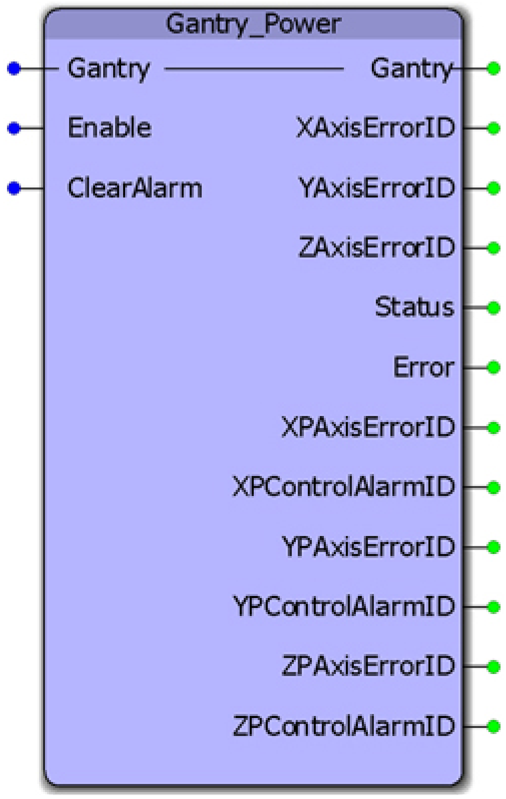
Error Description

ErrorID	Meaning
0	No error
1	Time limit exceeded
2	Distance limit exceeded
3	Torque limit exceeded
4369	The move could not be buffered because the axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4378	The function block is not applicable for the external axis specified
4379	A homing sequence is already in progress.
4380	MC_SetPosition can not be executed while the axis is moving.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
4382	When the axis is in 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.
4390	Position cannot be defined while the axis is the cam master of other axes.
4391	The function block cannot be used with a virtual axis.
4396	Axis latch function already in use.
4397	Over travel limit still ON after attempting to move away from it.
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.
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.
61713	An internal assertion in the motion kernel failed indicating the controller is not in a stable state. Please report this error to Yaskawa America Incorporated.

i

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.

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 while enable is held high.	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	

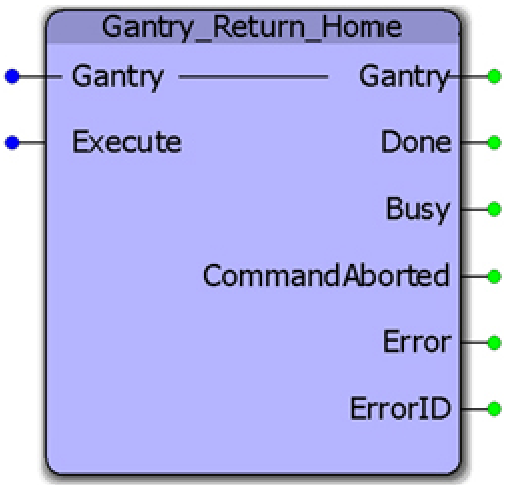
V	ZAxisErrorID	UINT	ErrorID on the Z axis
B	Status	BOOL	TRUE if the drive is enabled. This output is derived from the Status output of MC_Power.
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
V	XPAxisErrorID	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. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4371	The servo drive failed to enable or disable. Check the amplifier wiring for L1 / L2 / L3
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 (HHB) is preventing the drive from enabling.
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.
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.
61713	An internal assertion in the motion kernel failed indicating the controller is not in a stable state. Please report this error to Yaskawa America Incorporated.

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.

Parameters

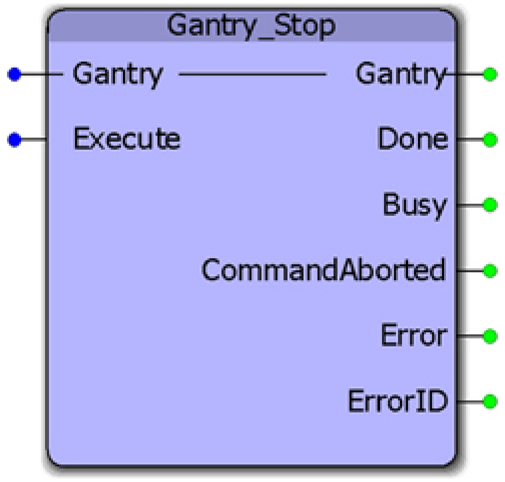
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has 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	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 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 axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

Gantry_Stop



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

Parameters

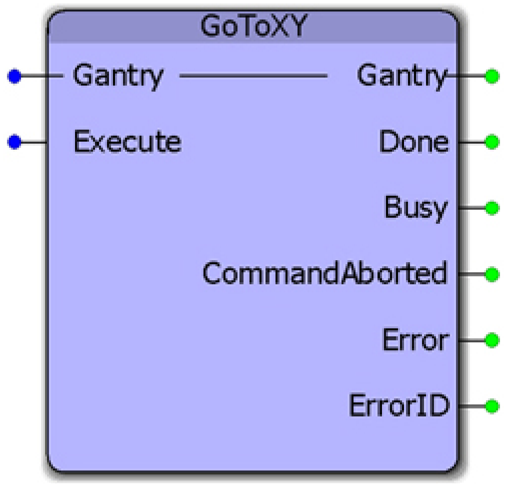
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has 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	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 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
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

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.

Parameters

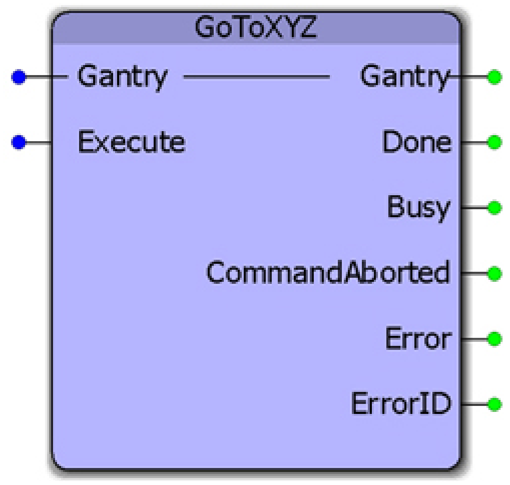
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has 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	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 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 axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

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.

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 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	CommandAborted	BOOL	Set high if motion is aborted by another motion command or MC_Stop. This output is cleared with the same behavior	

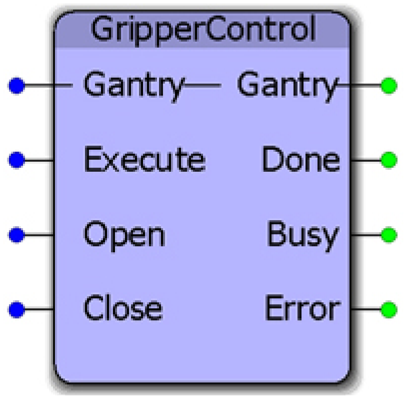
			as the Done output.
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

Error Description

ErrorID	Meaning
0	No error
4369	The move could not be buffered because the axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

.

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.

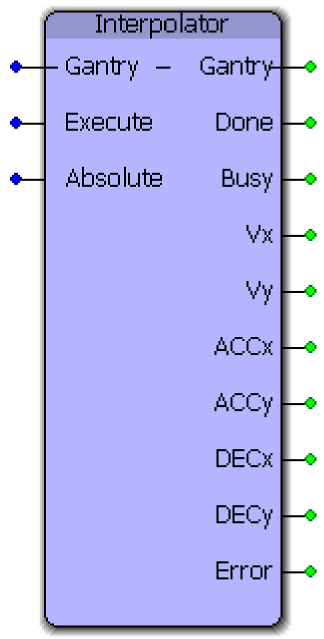
Parameters

*	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	Open	BOOL	Command to open the gripper	
B	Close	BOOL	Command to close the gripper	
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has been completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when execute goes low.	
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	

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.

Parameters

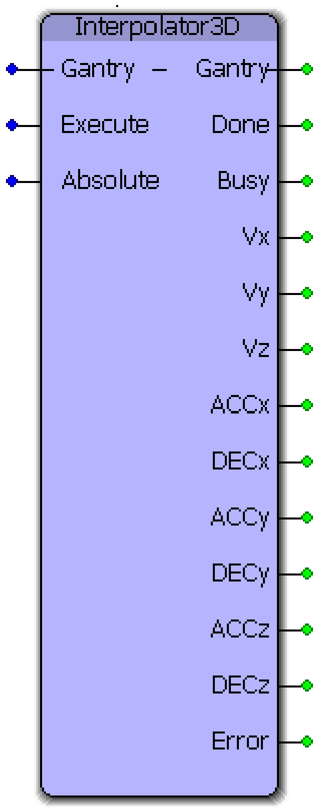
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				Default
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger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

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.

Parameters

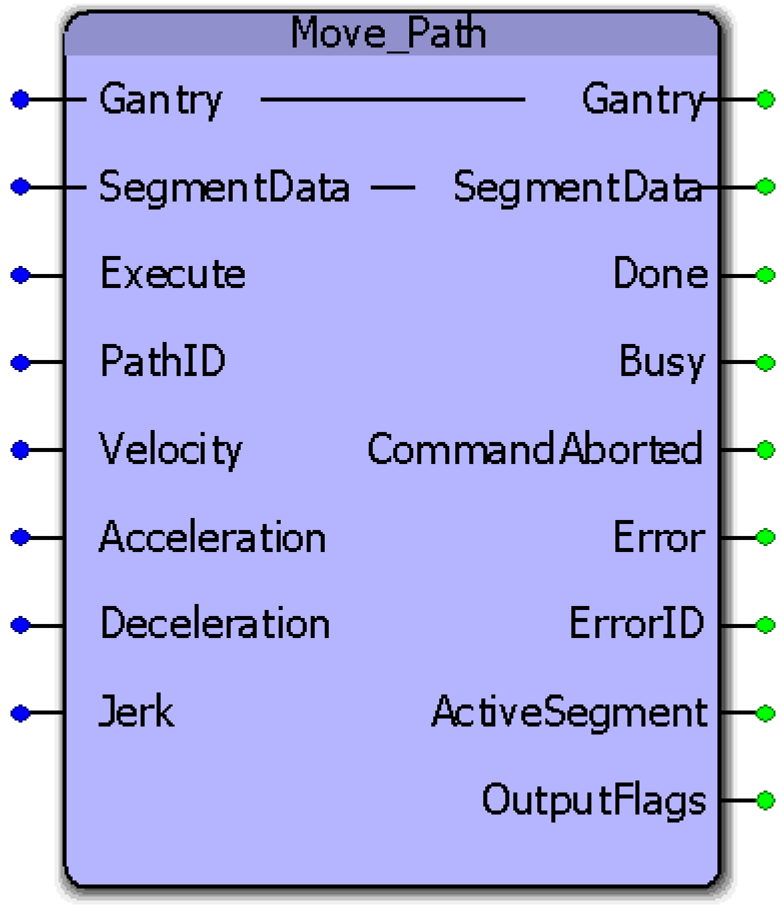
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and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

Error Description

ErrorID	Meaning
0	No error
10034	Interpolation calculation error.

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 controller along the path.

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	FALSE

			initiated. To modify an input, change the value and re-trigger the execute input.	
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*	*Not supported; reserved for future use. Value of the jerk in [user units / second^3].*	*LREAL#0.0*
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	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 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 absolute relative from the start point of the move. The axes can be moved using other motion blocks prior to executing Move_Path to account for offsets.

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

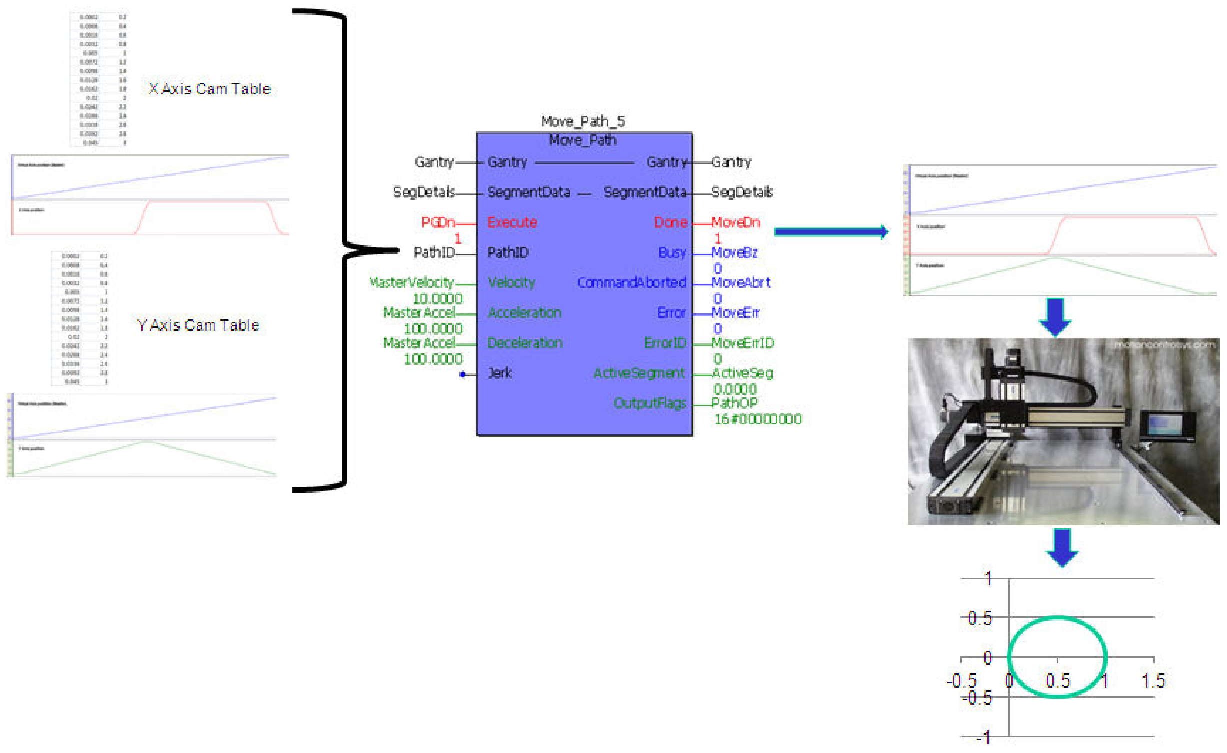
Error Description

ErrorID	Meaning
0	No error
4369	The move could not be buffered because the axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4378	The function block is not applicable for the external axis specified
4380	MC_SetPosition can not be executed while the axis is moving.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
4382	When the axis is in 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 the cam master of other axes.
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 cams Machine Cycle. (0 to Prm1502, the last master position in the active cam table.)
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.
4626	The master slave relationship is defined. A slave cannot be a master to another axis.
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.
57874	Argument data is NULL. The EngageData input must be connected.

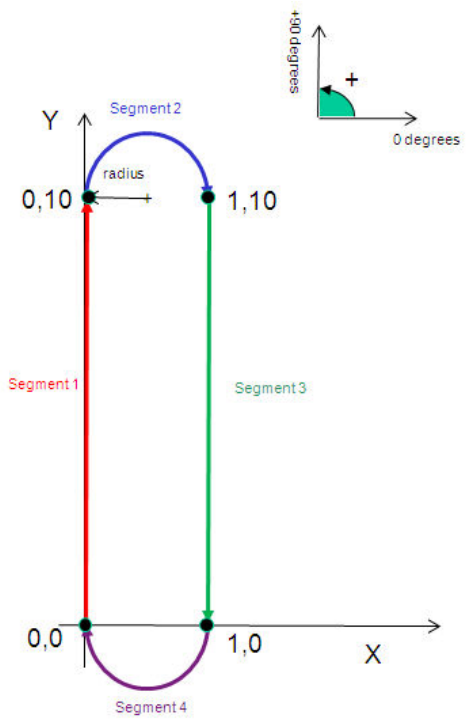
Example

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.

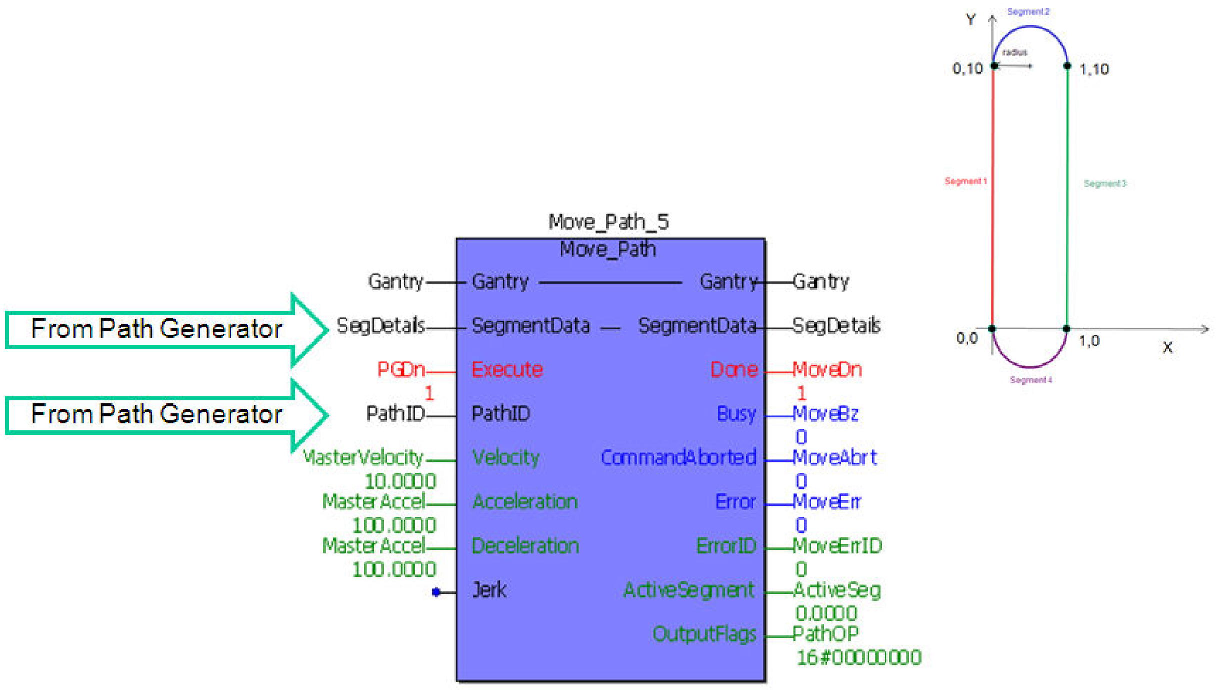


Consider the following contour:

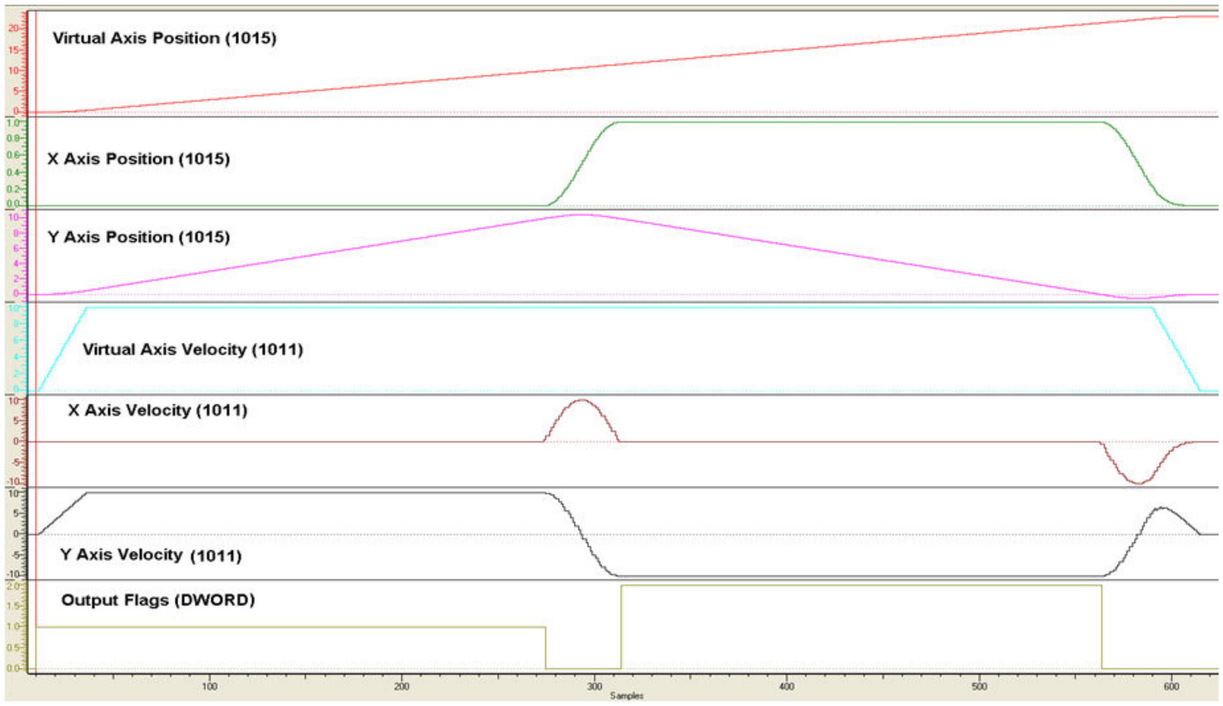
```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#0.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
VectorPath.Data[1].OutputFlags:=DWORD#1;

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;
```



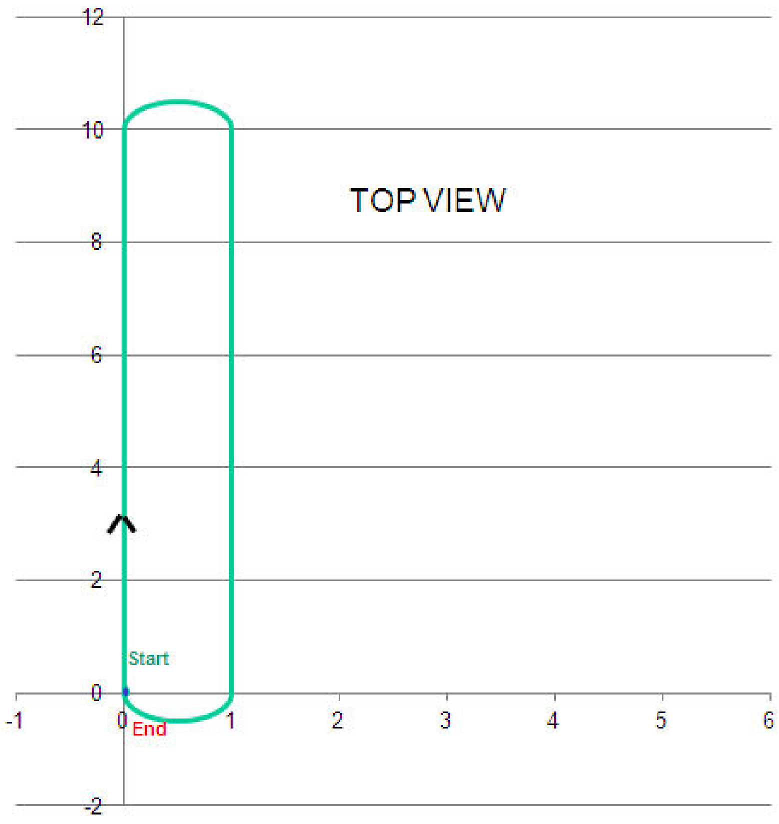
The MovePath function block uses SegmentData and PathID from the [PathGenerator](#) function block and executes moves on the X and Y axes. If a profile is made up of multiple segments (4 in the example below), the active segment output indicates which segment is being run. Output flags can be set from 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.


Move_Path_5
Move_Path
From Path Generator
From Path Generator
Gantry — Gantry — Gantry — Gantry
SegDetails — SegmentData — SegmentData — SegDetails
PGDn — Execute — Done — MoveDn
1 — 1
PathID — PathID — Busy — MoveBz
0
MasterVelocity — Velocity — CommandAborted — MoveAbrt
10.0000 — 0
MasterAccel — Acceleration — Error — MoveErr
100.0000 — 0
MasterAccel — Deceleration — ErrorID — MoveErrID
100.0000 — 0
Jerk — ActiveSegment — ActiveSeg
0.0000
OutputFlags — PathOP
16#00000000
Y
Segment 2
radius
0,10
1,10
Segment 1
Segment 3
0,0
1,0
X
Segment 4

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)

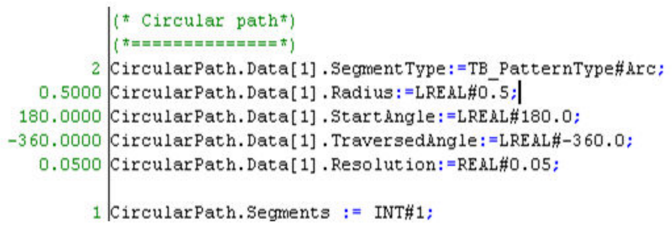


The actual profile plotted by the XY system is shown below

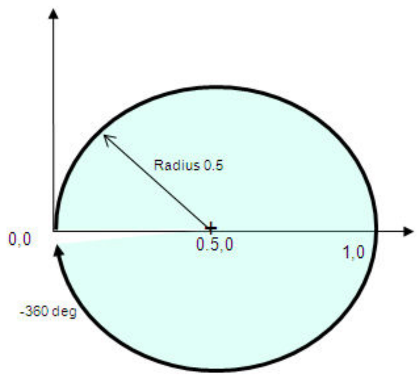


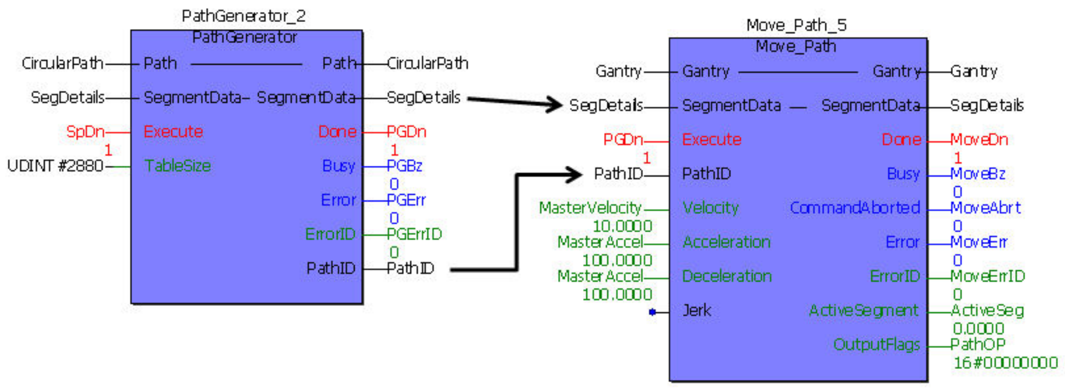
Code Example 2

Consider the following circular profile

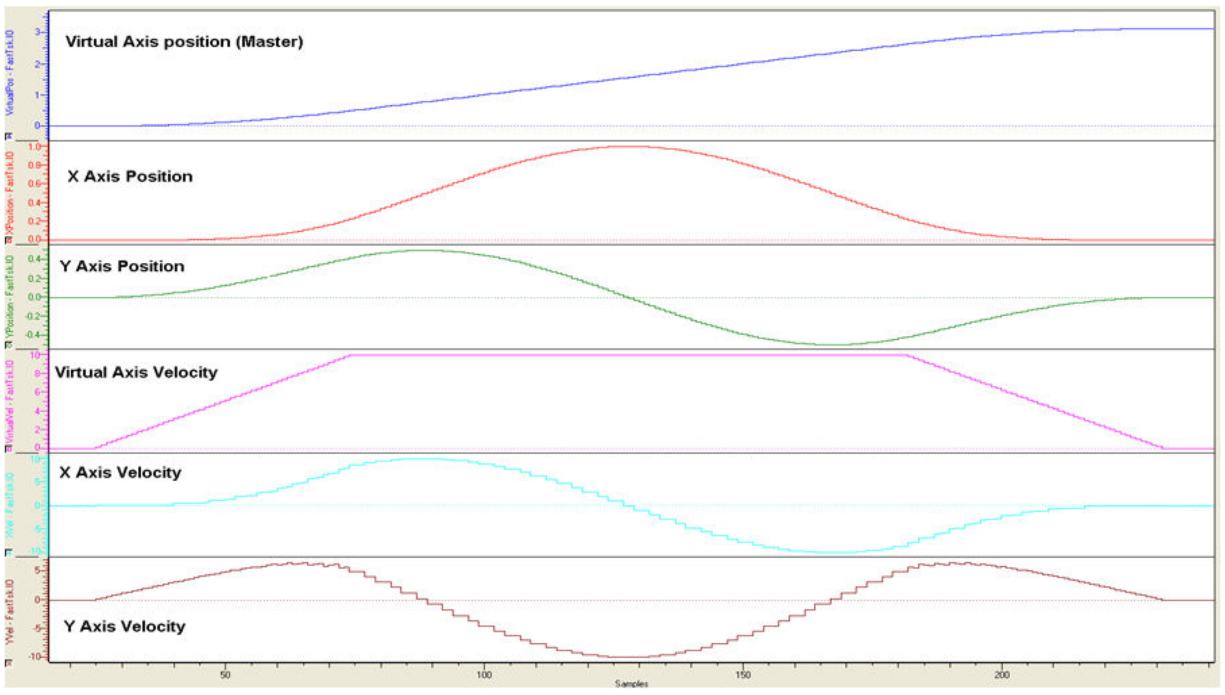


```
          (* Circular path*)
          (*==============*)
        2 CircularPath.Data[1].SegmentType:=TB_PatternType#Arc;
   0.5000 CircularPath.Data[1].Radius:=LREAL#0.5;
 180.0000 CircularPath.Data[1].StartAngle:=LREAL#180.0;
-360.0000 CircularPath.Data[1].TraversedAngle:=LREAL#-360.0;
   0.0500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

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

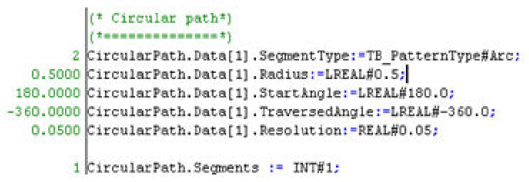




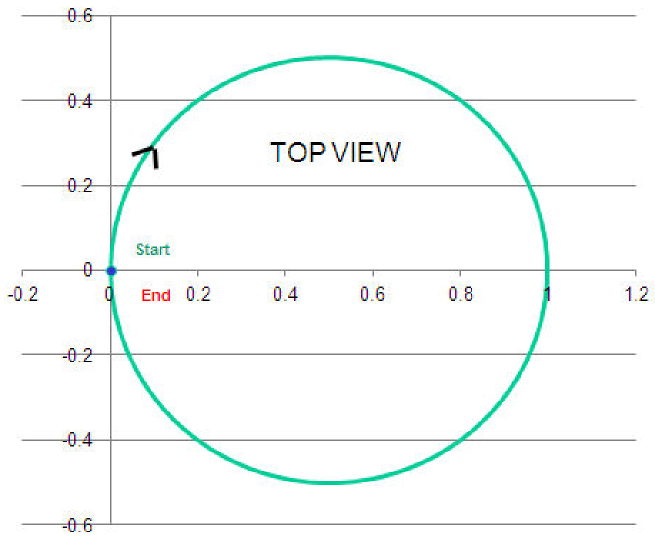
The logic analyzer traces from individual axes while Move_Path was busy is shown in the plot below



The actual profile plotted by the XY system is:



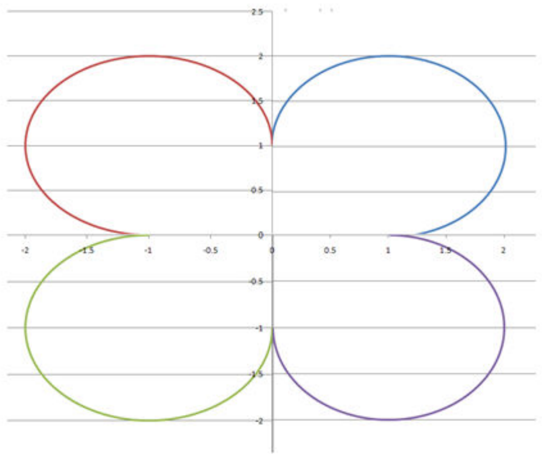
```
          (* Circular path*)
          (*==============*)
        2 CircularPath.Data[1].SegmentType:=TB_PatternType#Arc;
   0.5000 CircularPath.Data[1].Radius:=LREAL#0.5;
 180.0000 CircularPath.Data[1].StartAngle:=LREAL#180.0;
-360.0000 CircularPath.Data[1].TraversedAngle:=LREAL#-360.0;
   0.0500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

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



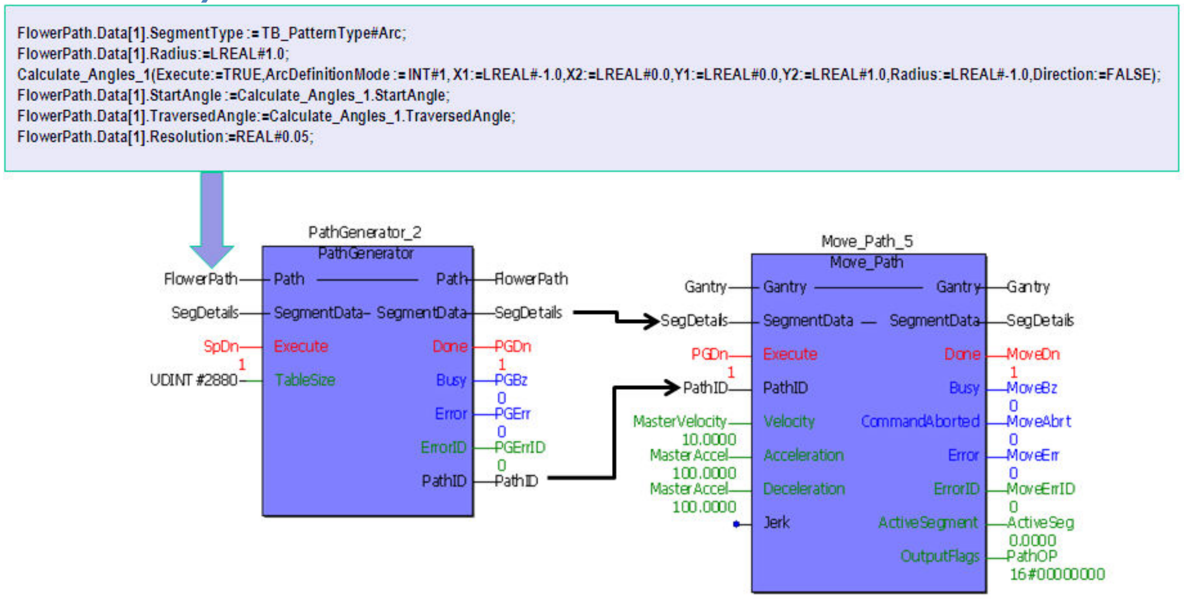
Application Example

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);
```



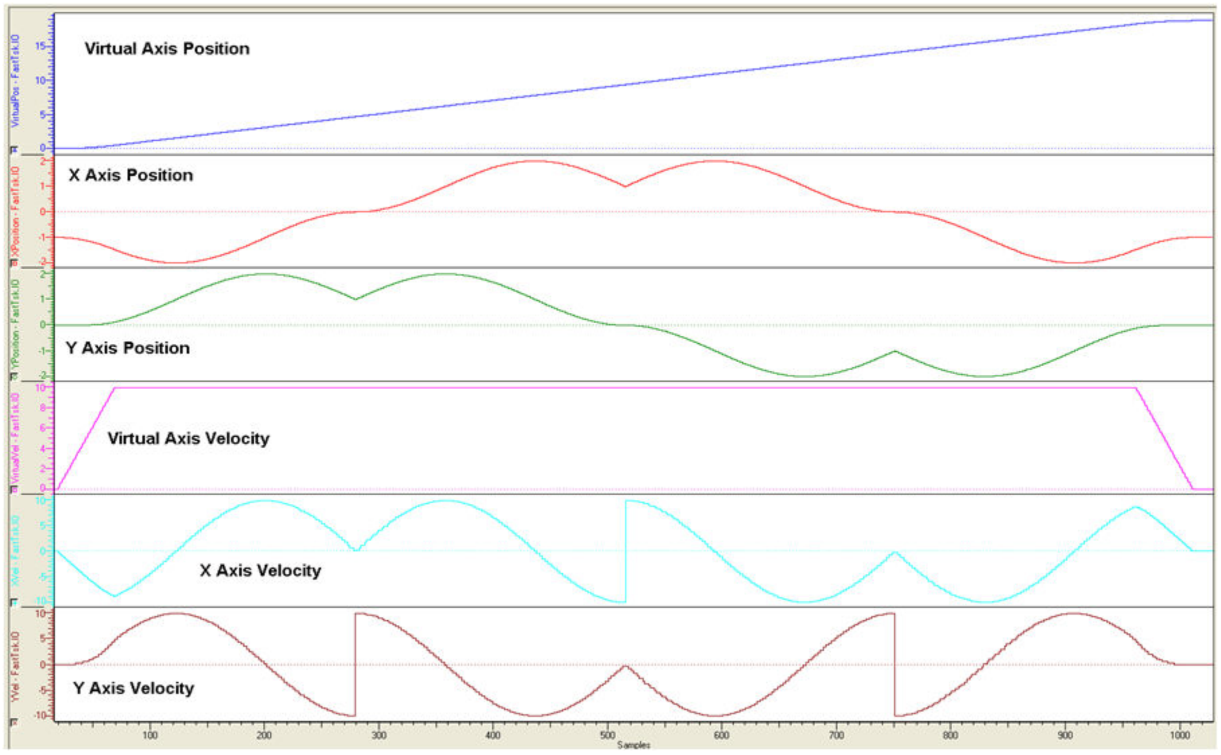
Step 2: Use the PathGenerator FB to create the path and the Move_Path FB to implement XY motion.



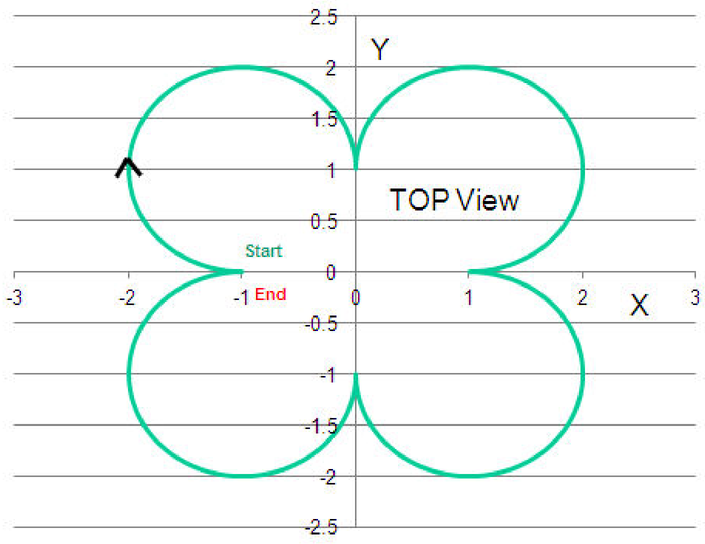
```
FlowerPath.Data[1].SegmentType :=TB_PatternType#Arc;
FlowerPath.Data[1].Radius:=LREAL#1.0;
Calculate_Angles_1(Execute:=TRUE,ArcDefinitionMode:=INT#1, X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);
FlowerPath.Data[1].StartAngle:=Calculate_Angles_1.StartAngle;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

PathGenerator_2
PathGenerator
FlowerPath — Path — Path — FlowerPath
SegDetails — SegmentData — SegmentData — SegDetails
SpDn — Execute — Done — PGDn 1
1
UDINT #2880 — TableSize — Busy — PGBz 0
Error — PGErr 0
ErrorID — PGErrID 0
PathID — PathID
Move_Path_5
Move_Path
Gantry — Gantry — Gantry — Gantry
SegDetails — SegmentData — SegmentData — SegDetails
PGDn — Execute — Done — MoveDn 1
1
PathID — PathID — Busy — MoveBz 0
MasterVelocity 10.0000 — Velocity — CommandAborted — MoveAbrt 0
MasterAccel 100.0000 — Acceleration — Error — MoveErr 0
MasterAccel 100.0000 — Deceleration — ErrorID — MoveErrID 0
Jerk — ActiveSegment — ActiveSeg 0.0000
OutputFlags — PathOP 16#00000000

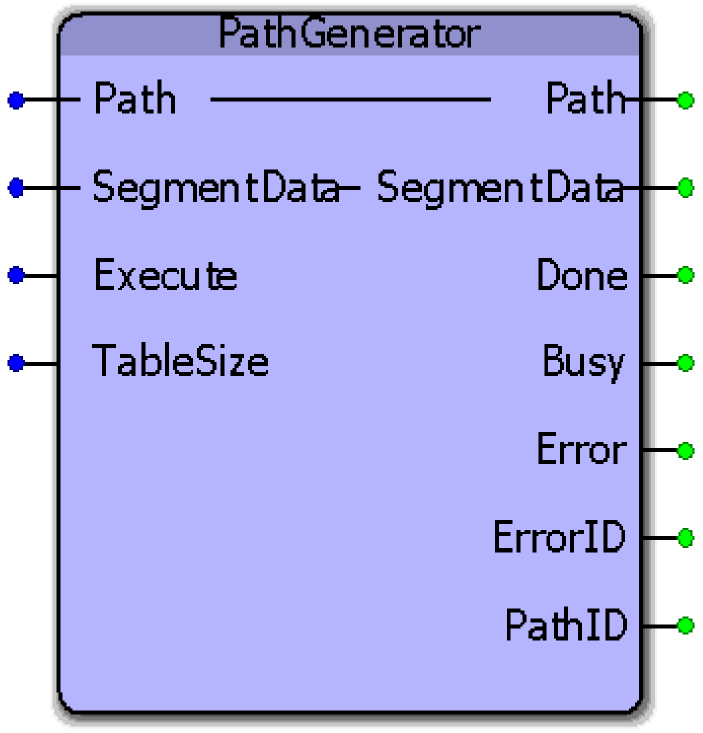
Step 3: Validation using logic analyzer.



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, Theta, and a Tangent axis are provided.

Parameters

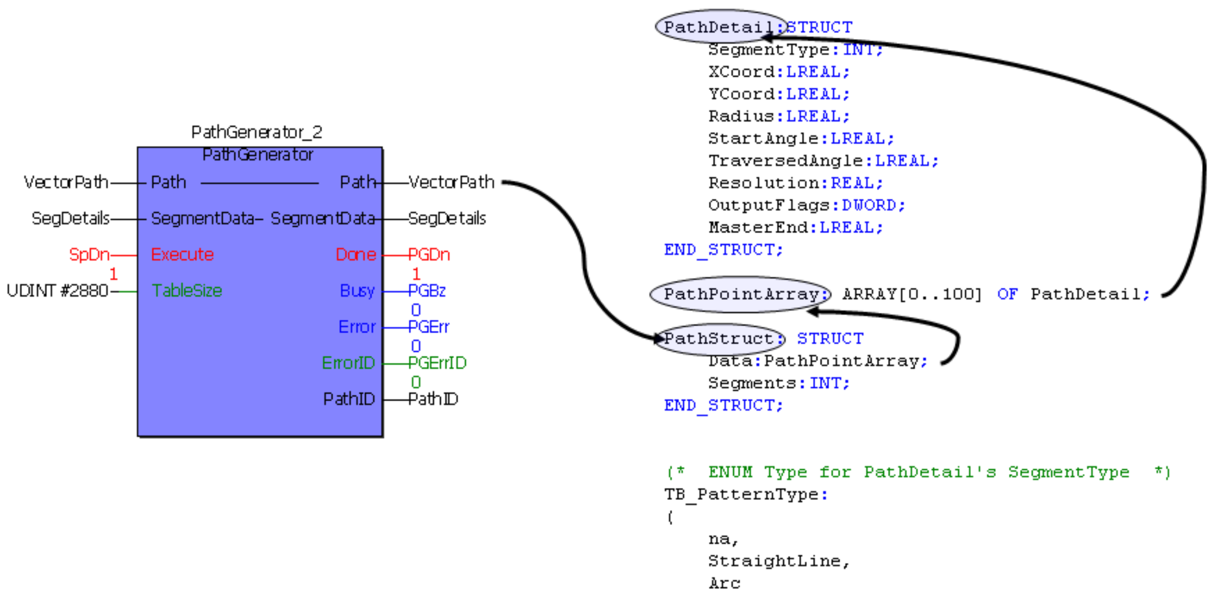
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Path	PathStruct	Structure of data that describes a motion path containing straight lines and arc segments.	
V	SegmentData	SegmentStruct	Structure of data that contains the segment number, output code, and tool path endpoint for each segment in the motion 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 been completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when execute goes low.
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.
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 function block.

Notes

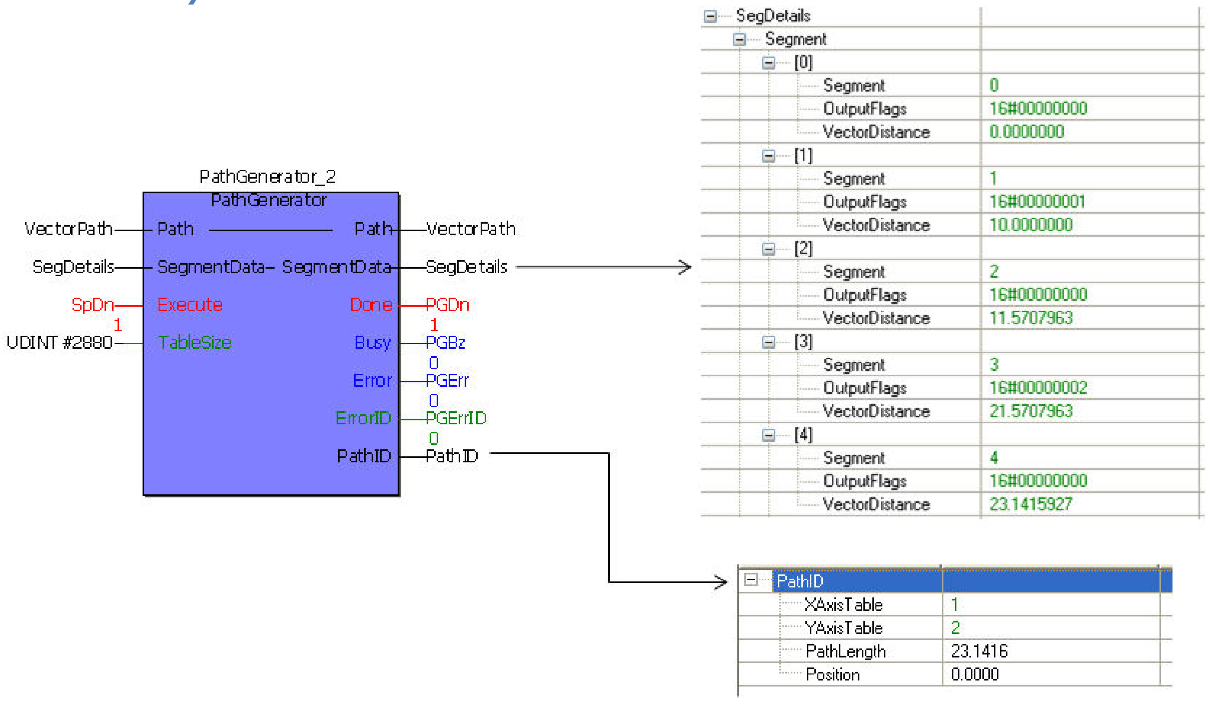
This function converts user defined straight line, arc segment, input and output data into cam files which produce coordinated motion. The cam files are loaded into the motion engine ready for use.

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

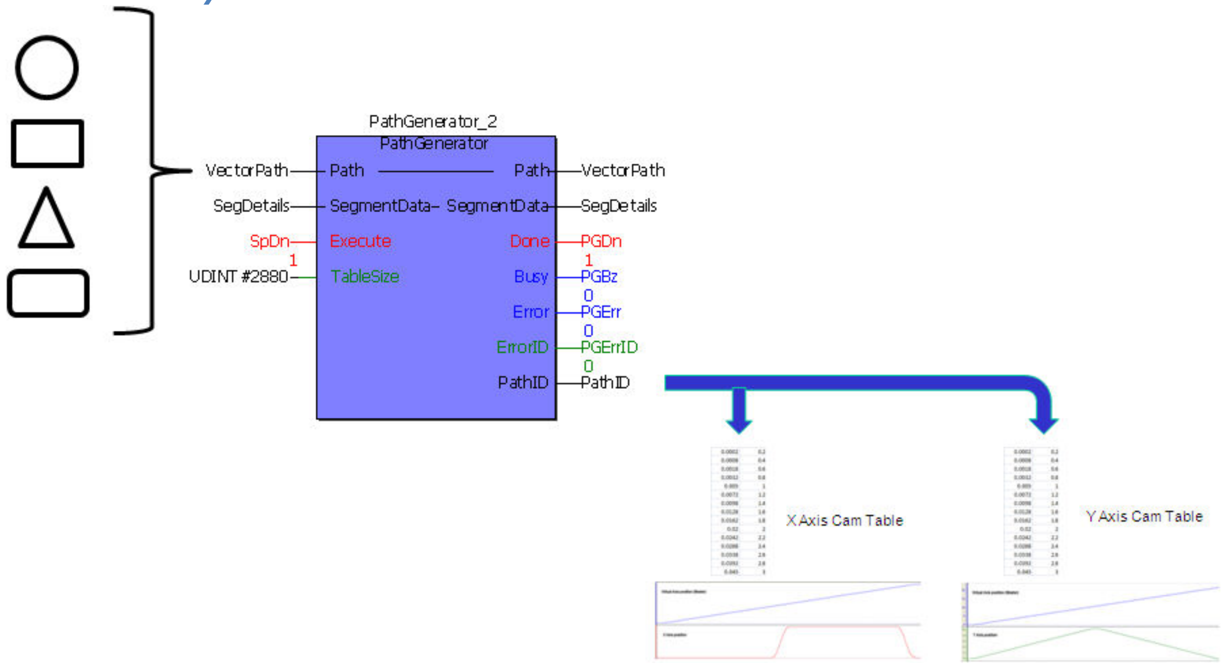
PathID	
XAxisTable	1
YAxisTable	2
PathLength	23.1416
Position	0.0000

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

Error Description

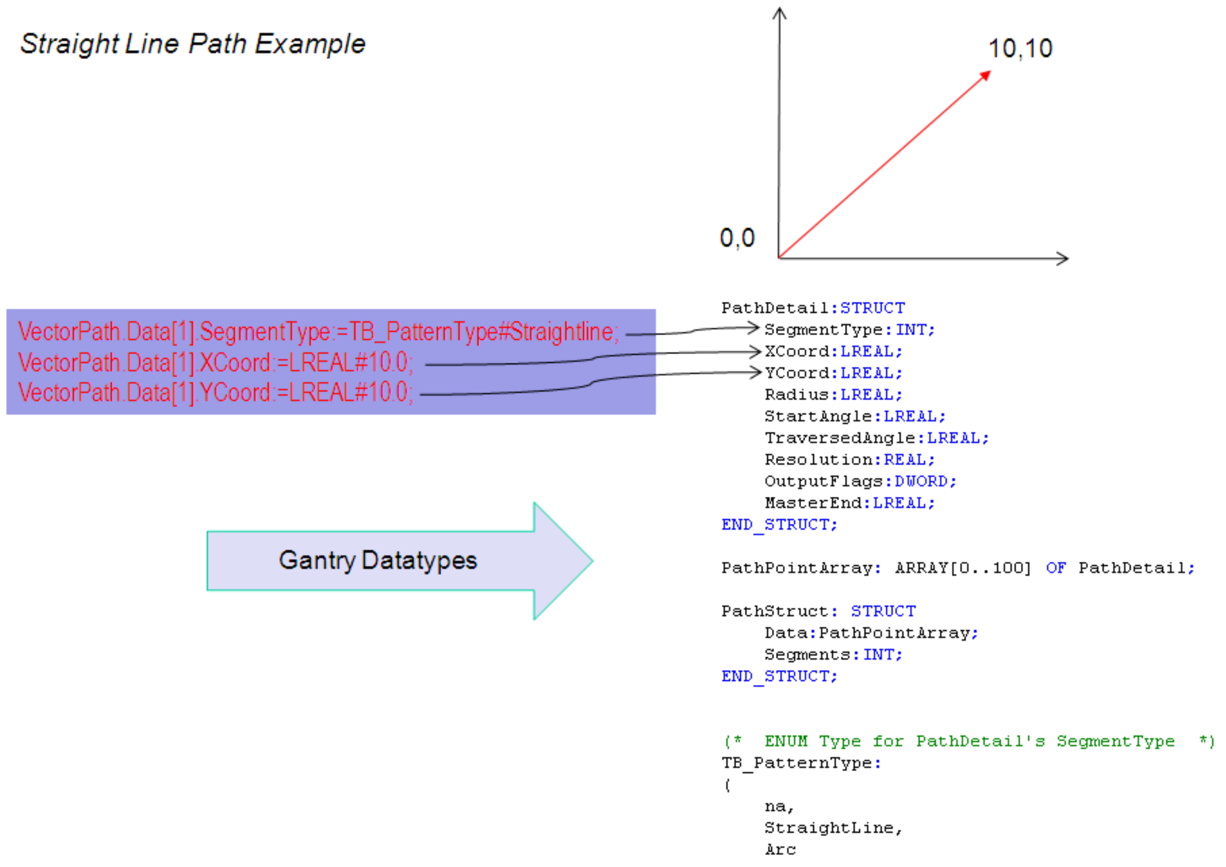
ErrorID	Meaning
0	No error
10038	CamData.LastSegment must be greater than 0 and less than 400, or whatever value has been declared as the ARRAY size in the CTB_Types file.
10053	DataPoint Error
10054	One of the segments in the path has an invalid Segment Type. Path.Data[Segment].SegmentType must be coded as either being a line (INT#1) or an arc (INT#2).
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

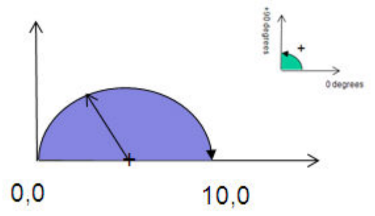


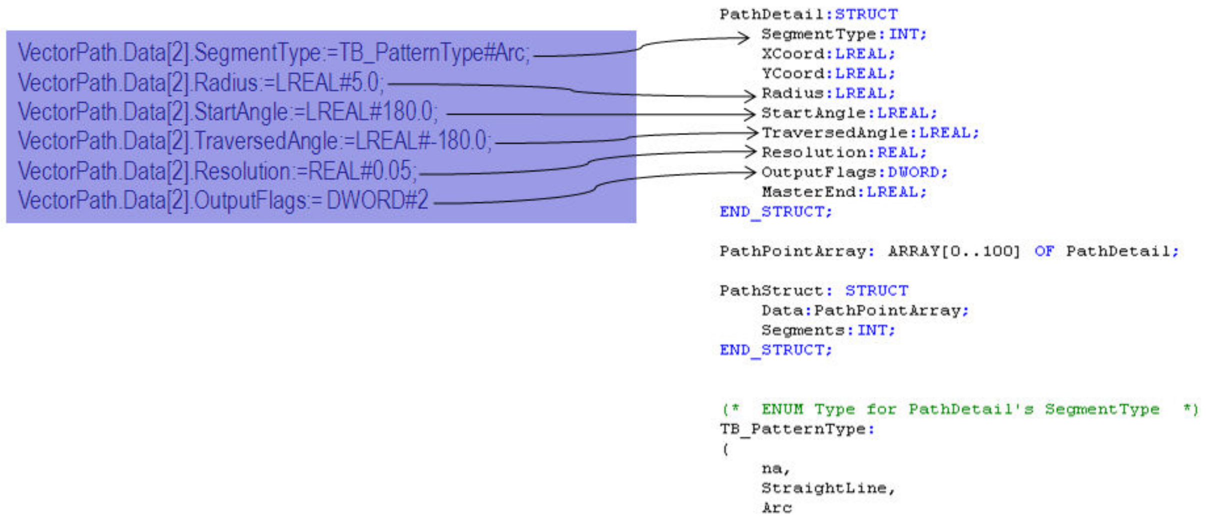
```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#10.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
```

```
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

Arc Path Example





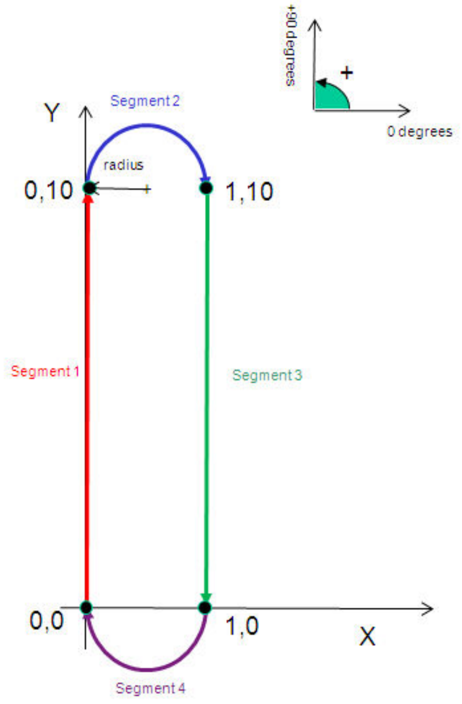
```
VectorPath.Data[2].SegmentType:=TB_PatternType#Arc;
VectorPath.Data[2].Radius:=LREAL#5.0;
VectorPath.Data[2].StartAngle:=LREAL#180.0;
VectorPath.Data[2].TraversedAngle:=LREAL#-180.0;
VectorPath.Data[2].Resolution:=REAL#0.05;
VectorPath.Data[2].OutputFlags:= DWORD#2
```

```
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

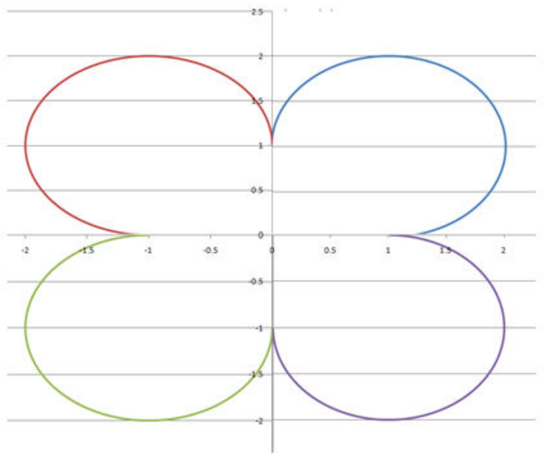
```
VectorPath.Data[1].SegmentType:=TB_PatternType#Straightline;
VectorPath.Data[1].XCoord:=LREAL#0.0;
VectorPath.Data[1].YCoord:=LREAL#10.0;
VectorPath.Data[1].OutputFlags:=DWORD#1;

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

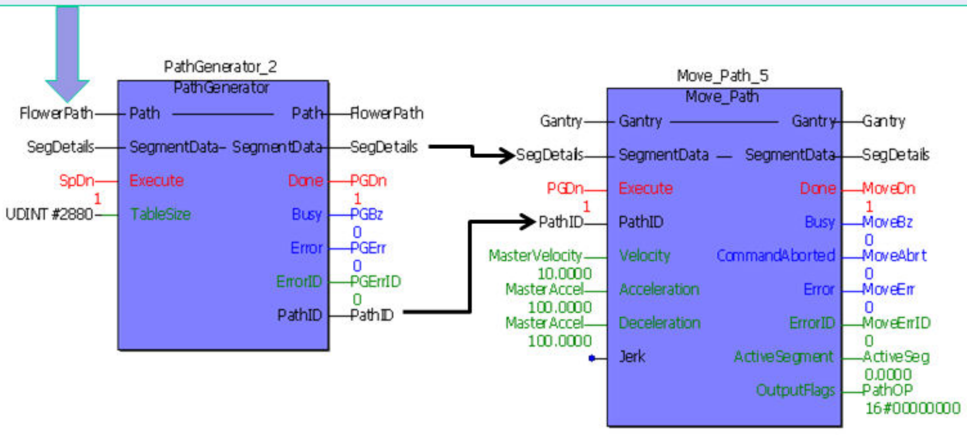
Step1: Using Calculate_Angles to calculate start and traverse angles for the flower path shown below

```
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);
```

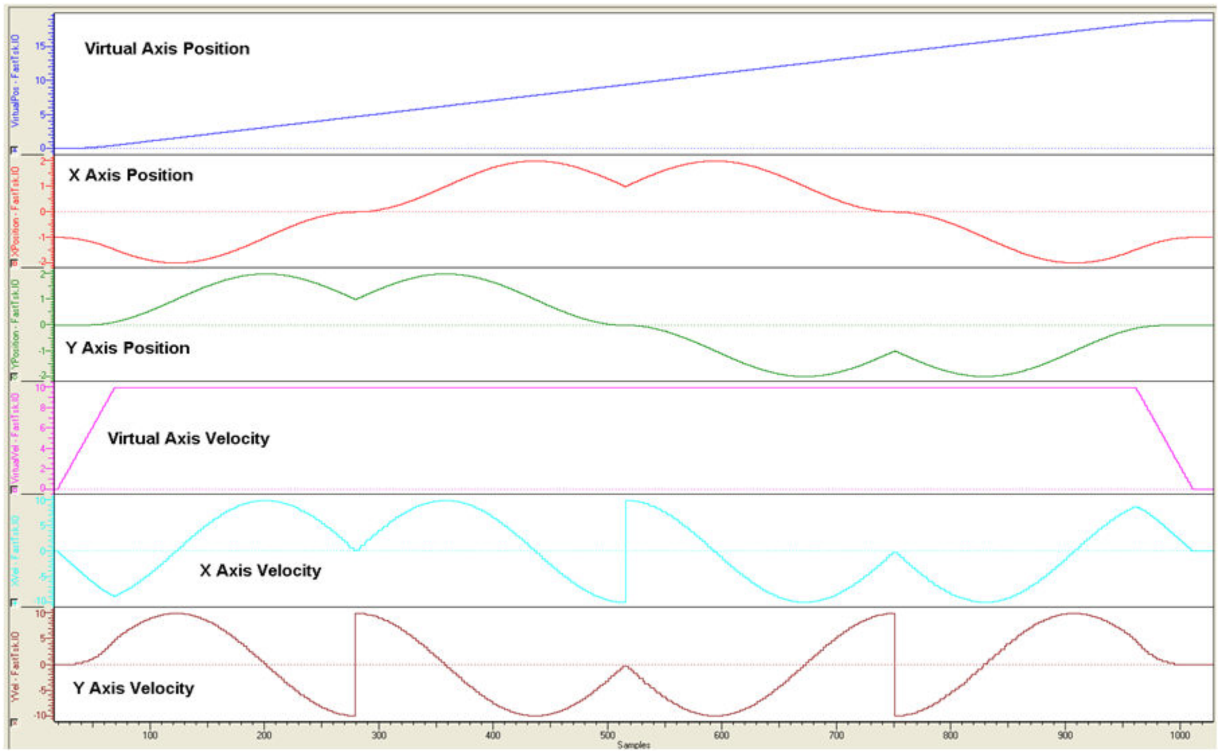


Step 2: Use PathGenerator create the path and Move_Path to implement XY motion

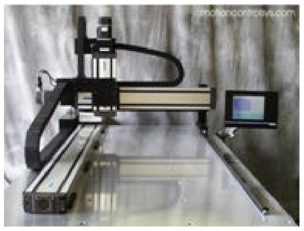
```
FlowerPath.Data[1].SegmentType:=TB_PatternType#Arc;
FlowerPath.Data[1].Radius:=LREAL#1.0;
Calculate_Angles_1(Execute:=TRUE,ArcDefinitionMode:=INT#1,X1:=LREAL#-1.0,X2:=LREAL#0.0,Y1:=LREAL#0.0,Y2:=LREAL#1.0,Radius:=LREAL#-1.0,Direction:=FALSE);
FlowerPath.Data[1].StartAngle:=Calculate_Angles_1.StartAngle;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

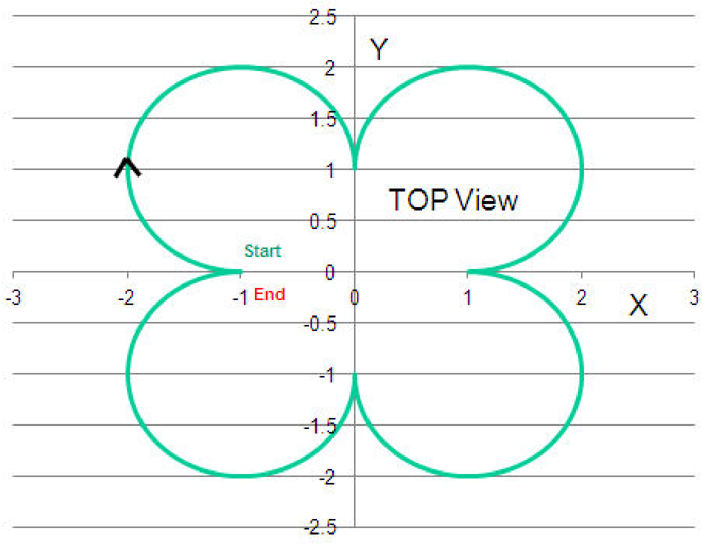


Step 3: Validation using logic analyzer

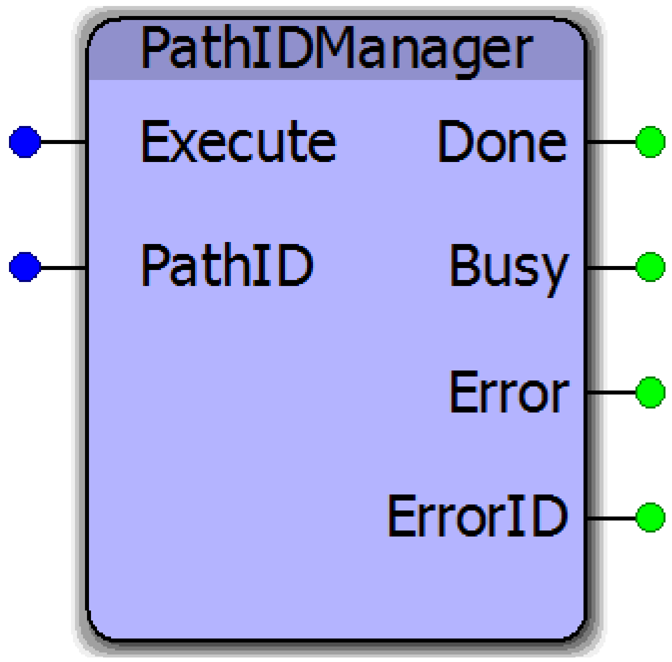


Step 4: Result on XY system




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

PathIDManager



This function block serves as a FIFO buffer for PathID's. Each time a new PathID is created, it will delete the memory allocated to the oldest set of CamTableIDs used for a PathID by using the Y_RemoveCamTable function block from the PLCopenPlus firmware library. This function block cleans up memory in IEC applications which build new paths on the fly. A circular buffer of four PathID tables is maintained in the PathIDManager. When the function block is executed a fifth time, it releases the memory area of the oldest PathID. The controller can allocate this memory area for new Paths or application code.

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	UINT	The most recent PathID created by Y_CamFileSelect or Y_CamStructSelect	UINT#0
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has been completed successfully. If another block takes control before the action is completed, the Done output will not be set. This output is reset when execute goes low.	

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

Notes

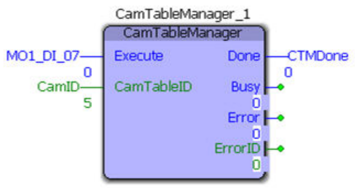
- This function block is unnecessary in applications which use a single, static PathID.

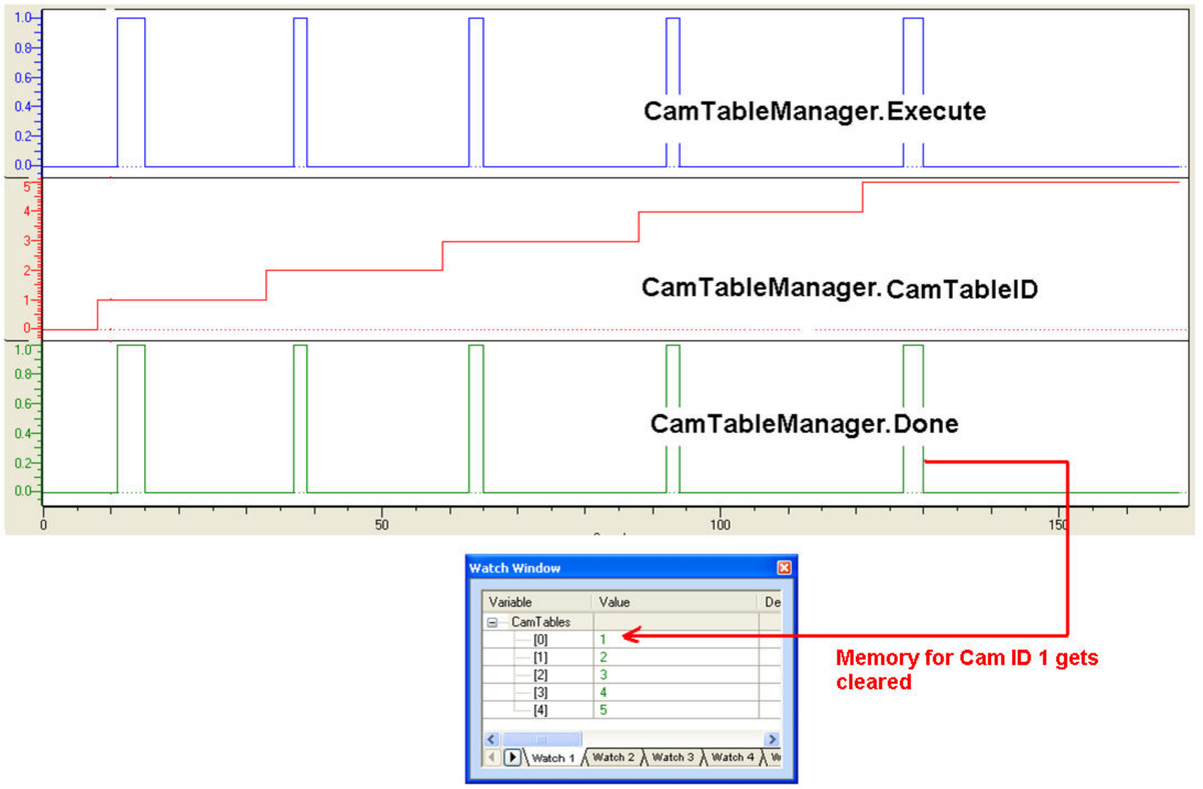
Error Description

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

Example 1

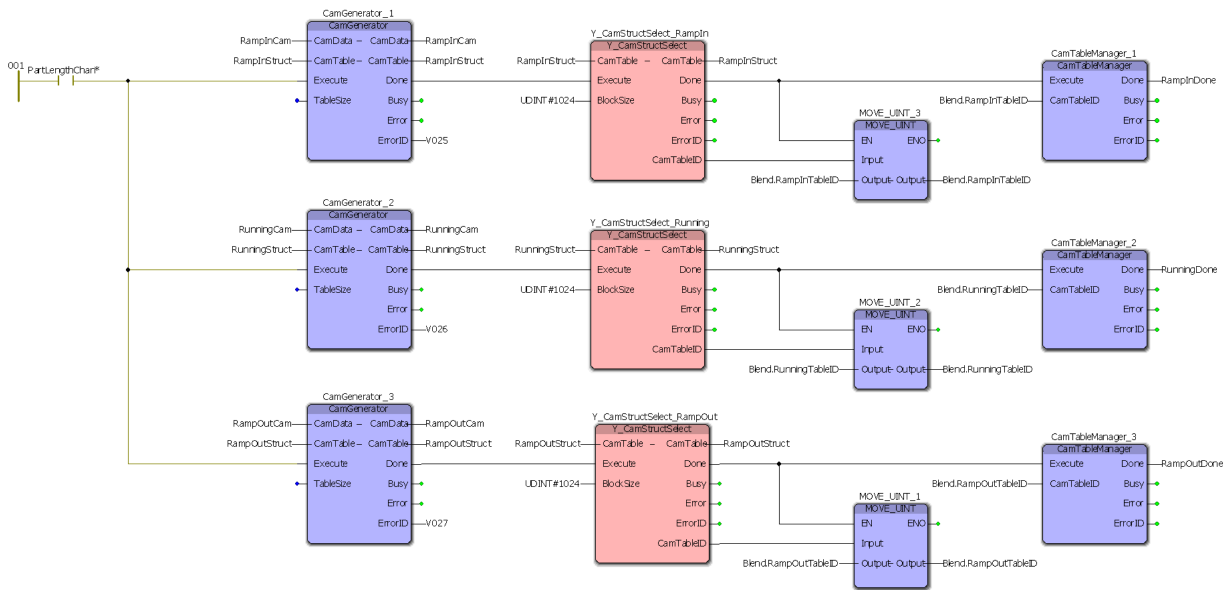
An example of using the CamTableManager is shown below; it operates very similarly to the PathIDManager function block. On the fifth execute of the PathIDManager block, the memory for the oldest Path ID gets released. In the example shown below, the memory for PathID 1 gets released. The next execution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



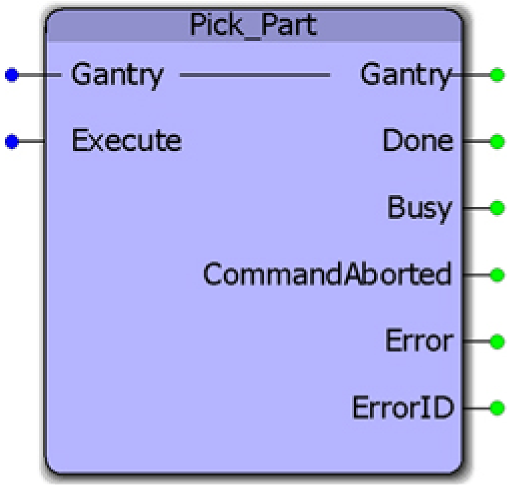


Variable	Value	De
CamTables		
[0]	1	
[1]	2	
[2]	3	
[3]	4	
[4]	5	

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.

Parameters

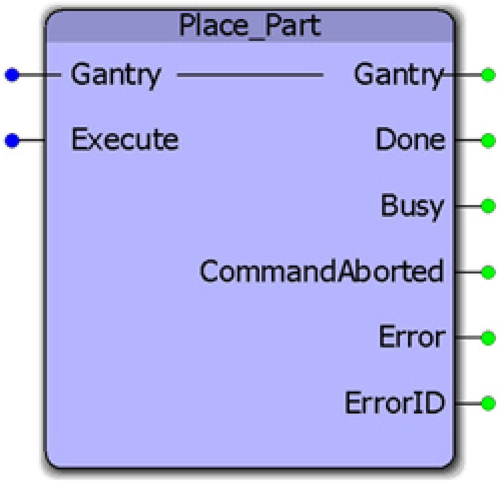
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has 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	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 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 axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

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

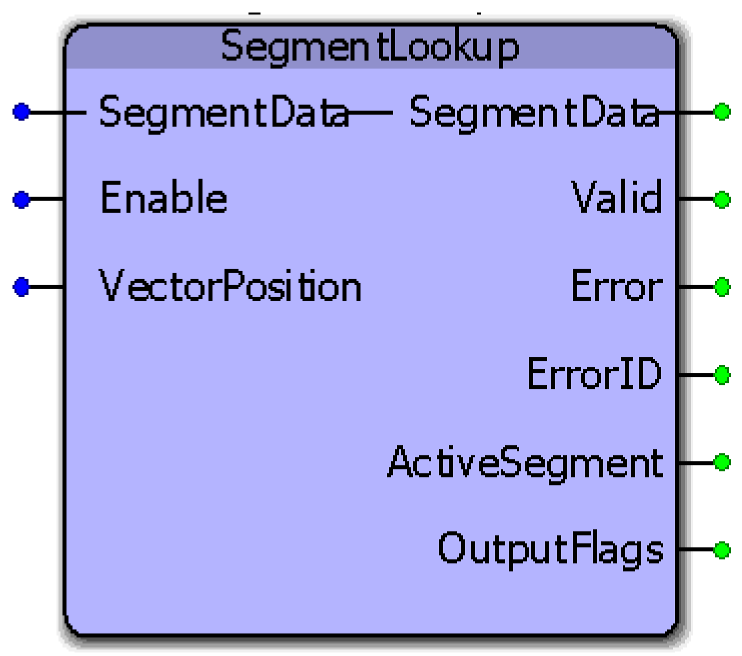
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	Gantry	GantryStruct	Contains all information pertaining to a gantry system.	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
VAR_OUTPUT				
B	Done	BOOL	Set high when the commanded action has 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	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 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 axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

SegmentLookup



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

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 while enable is held high.	FALSE
B	VectorPosition	BOOL	Position of the master vector (master axis)	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
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.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

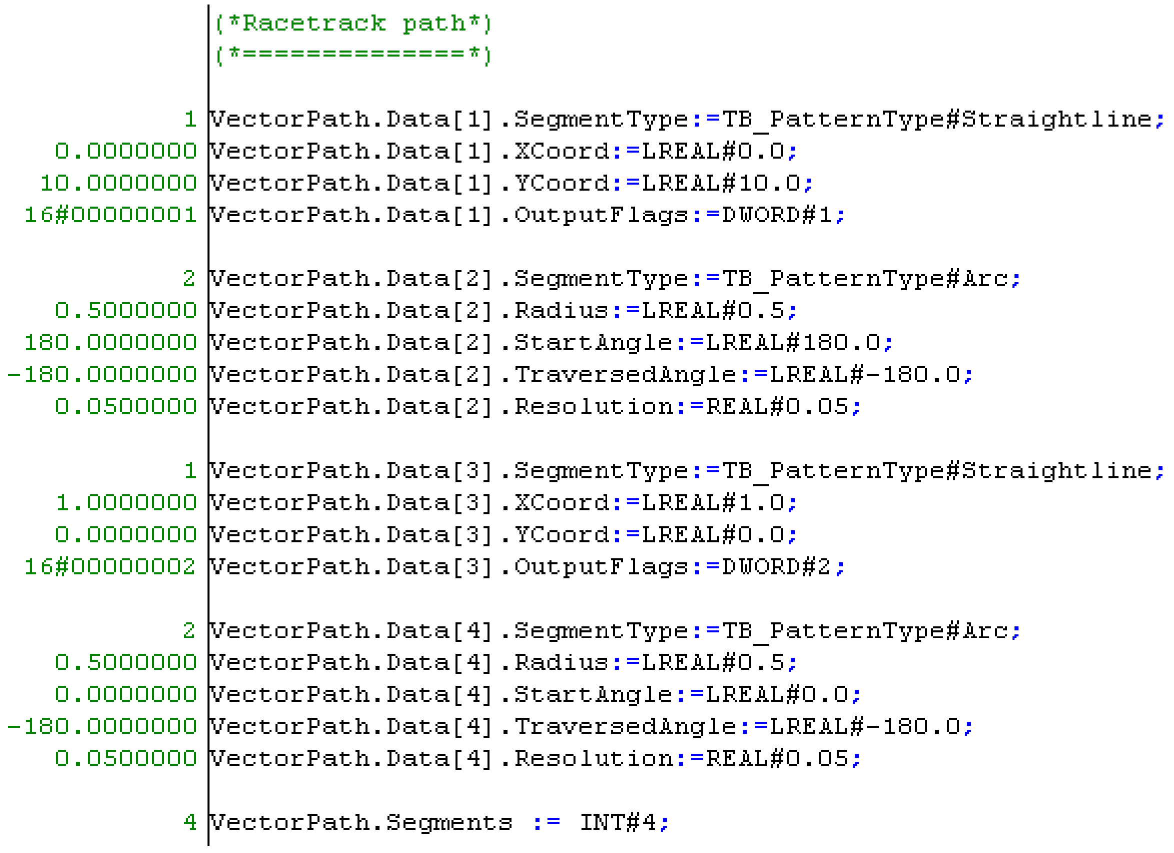
B	ActiveSegment	INT	Current active segment
B	OutputFlags	DWORD	Outputs DWORD that can be used to control digital output patterns during segments

Error Description

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

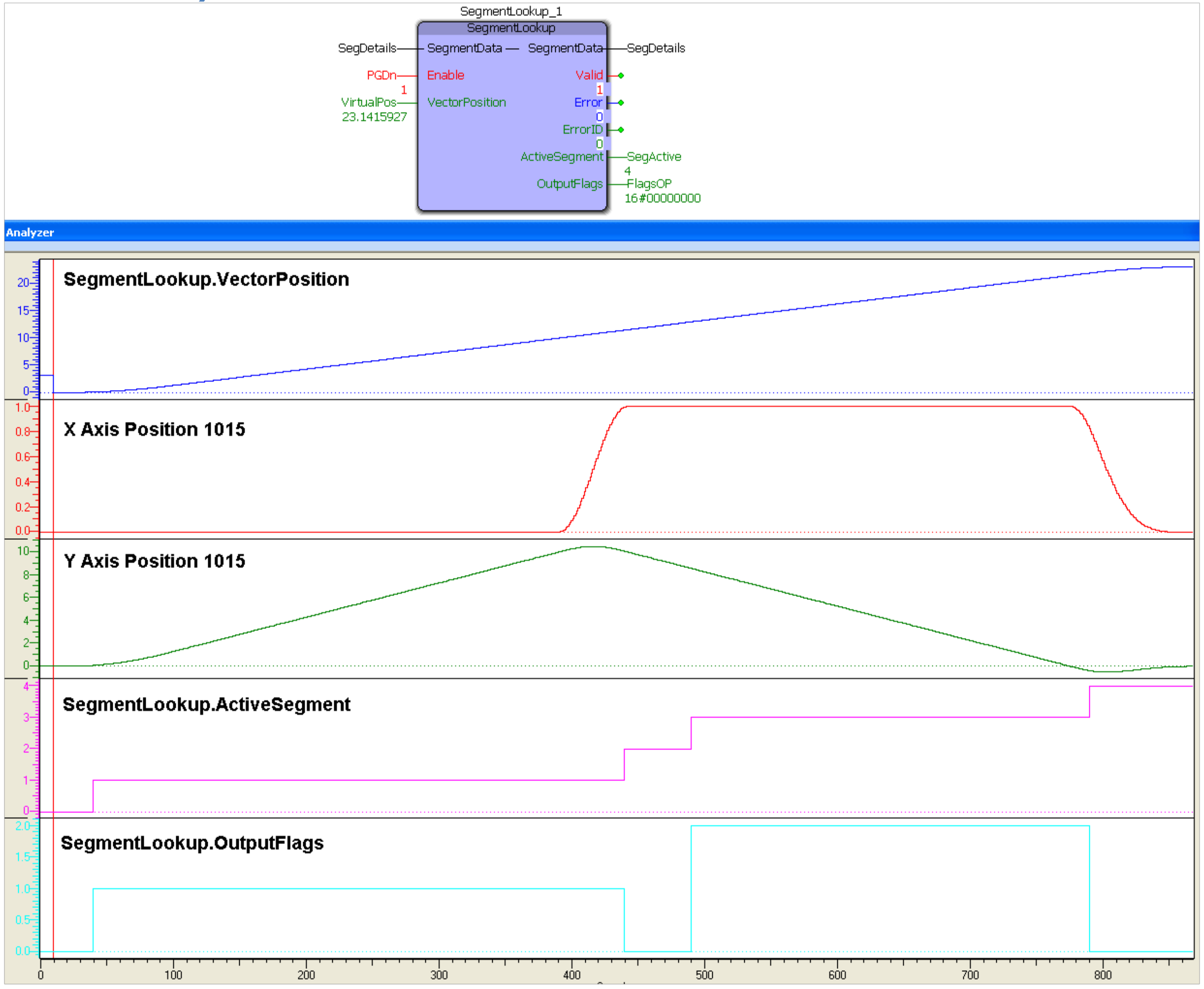
Example

Consider the profile shown below:

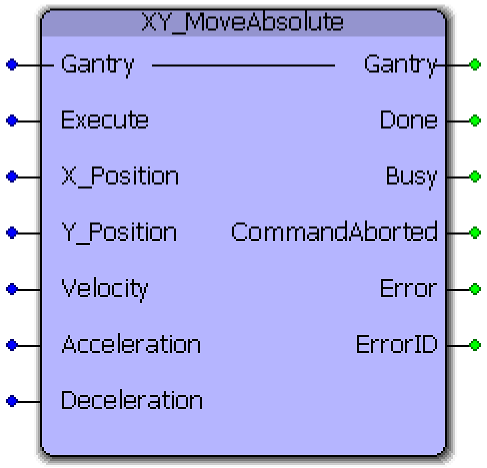


```
               (*Racetrack path*)
               (*==============*)

            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
VectorPosition
Error
23.1415927
0
ErrorID
0
ActiveSegment
SegActive
4
OutputFlags
FlagsOP
16#00000000
Analyzer
SegmentLookup.VectorPosition
X Axis Position 1015
Y Axis Position 1015
SegmentLookup.ActiveSegment
SegmentLookup.OutputFlags

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.

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 tool tip	0.0
B	Y_Position	LREAL	Target Y coordinate of the tool tip	0.0
B	Velocity	LREAL	Velocity of the tool tip	0.0
B	Acceleration	LREAL	Acceleration of the tool tip	0.0
B	Deceleration	LREAL	deceleration of the tool tip	0.0
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	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 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.

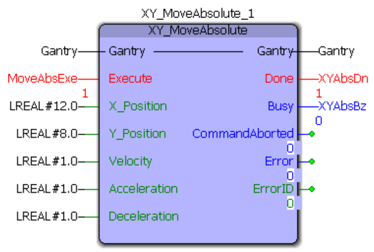
Error Description

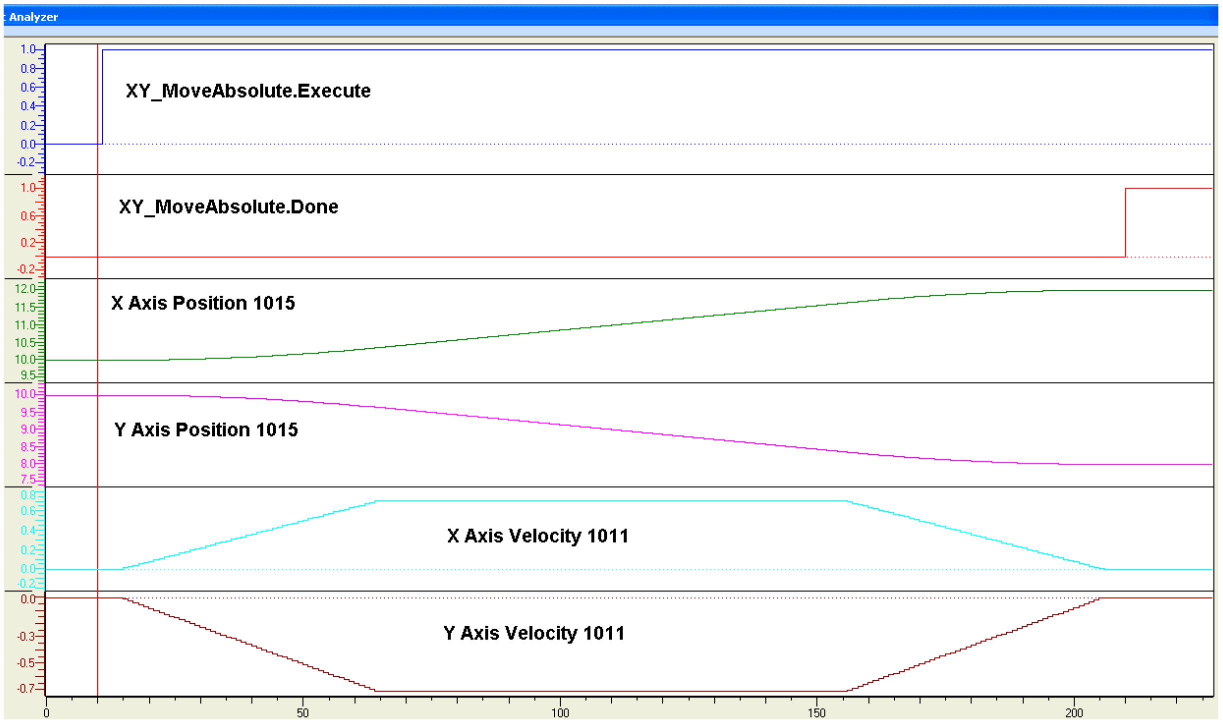
ErrorID	Meaning
0	No error
4369	The move could not be buffered because the axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

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.

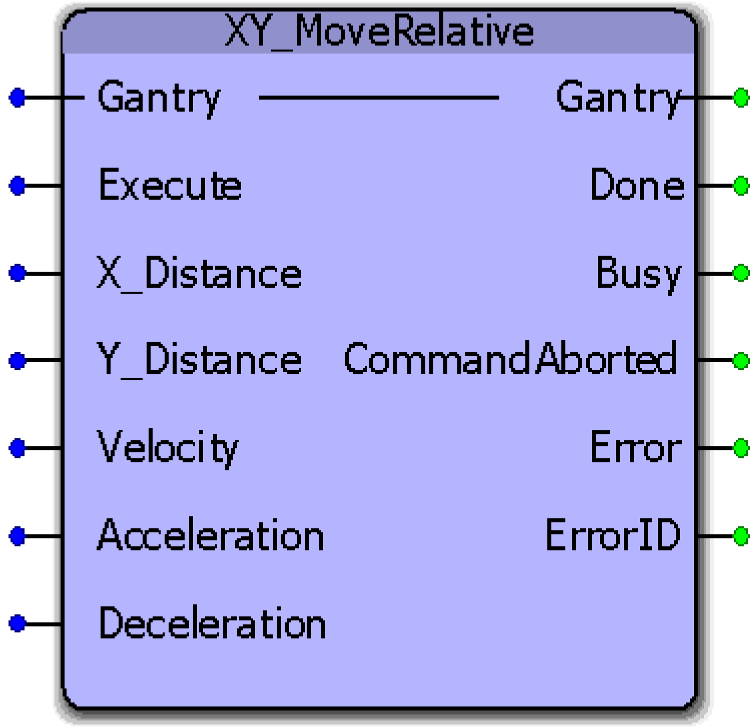
The velocities, accelerations and decelerations of the two axes are calculated (in XY_MoveAbsolute) such that the individual axes start and stop at the same time instant.





"

XY_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.

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_Distance	LREAL	X coordinate distance to be moved	0.0
B	Y_Distance	LREAL	Y coordinate distance to be moved	0.0
B	Velocity	LREAL	Velocity of the tool tip	0.0

B	Acceleration	LREAL	Acceleration of the tool tip	0.0
B	Deceleration	BOOL	Deceleration of the tool tip	0.0
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	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 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.	

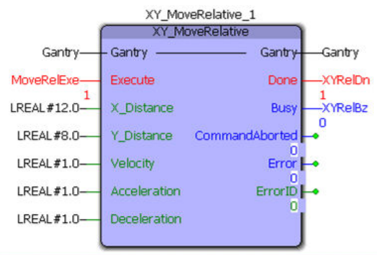
Error Description

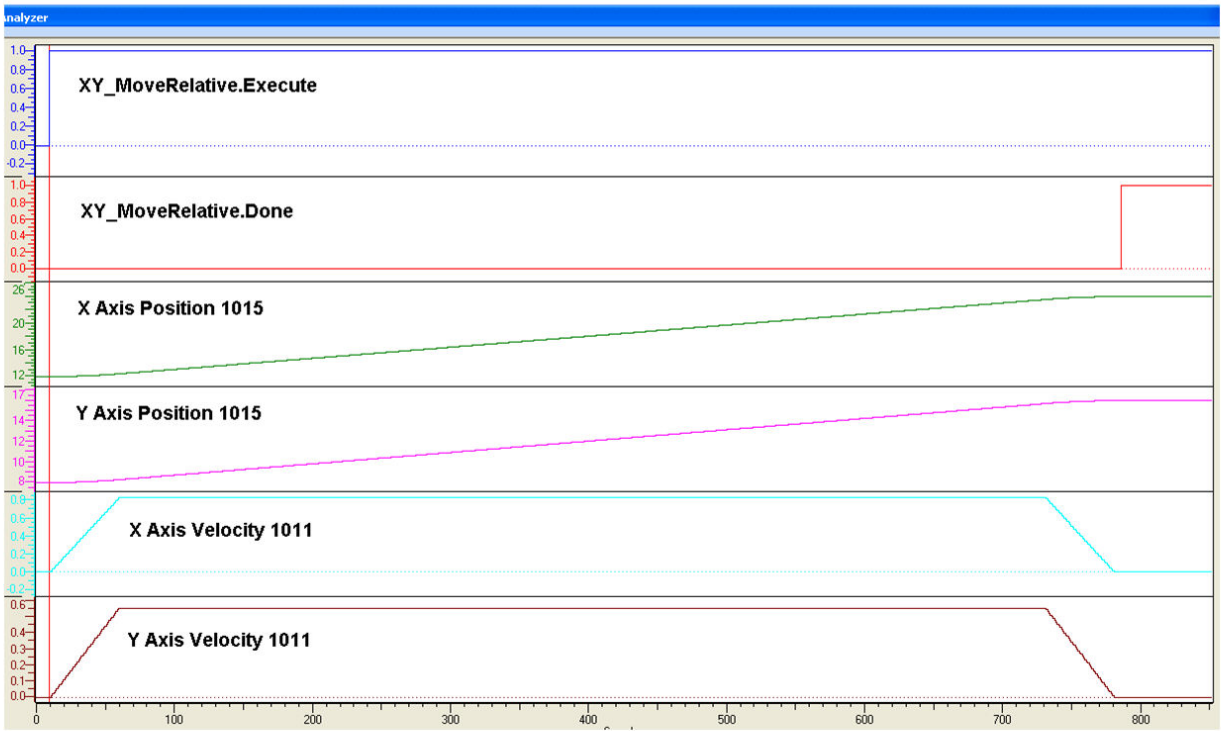
ErrorID	Meaning
0	No error
4369	The move could not be buffered because the axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

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.





Math Toolbox

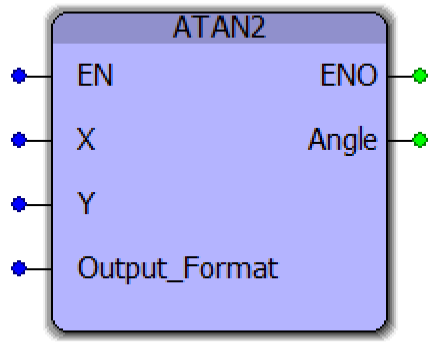
Math Toolbox

The Math toolbox contains many functions that already exist in the MotionWorks IEC Edit Wizard. The purpose for this duplication was originally to provide compatibility and support for the MP2600iec controller with its PLC operating system called eCLR. As of firmware version 1.2.3, the eCLR operating system supports EN / ENO input and outputs, but this Toolbox is still maintained for legacy support.

In addition to the many basic functions duplicated in this toolbox, some additional functionality is also provided.

Function Blocks

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.

Parameters

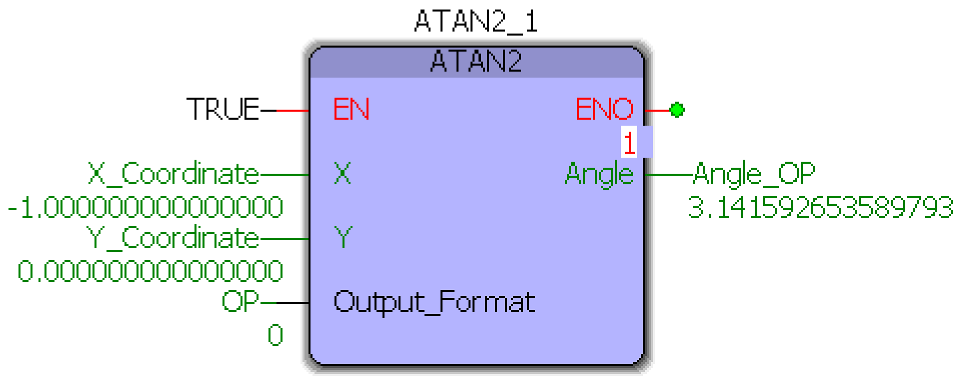
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	EN	BOOL	This function will continue to calculate the ATAN2 result while EN is held high.	FALSE
V	X	LREAL	X coordinate	LREAL#0.0
V	Y	LREAL	Y coordinate	LREAL#0.0
V	Output_Format	INT	Format of the output value. 0: radians (-pi, pi] 1: radians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

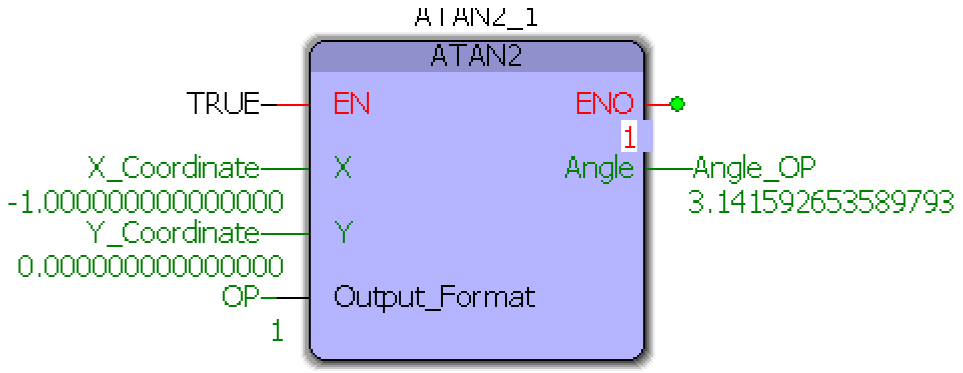
Notes

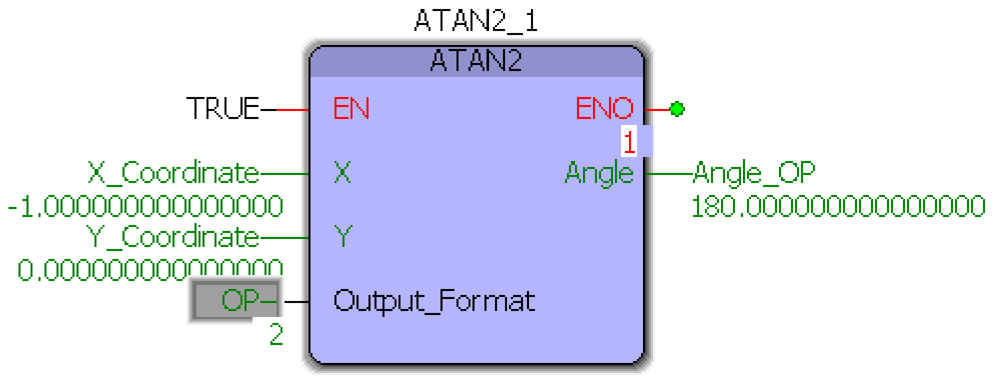
This is a function, not a function block and only provides one output. If ENO is not high when EN is high, this function cannot calculate the Angle.

Example

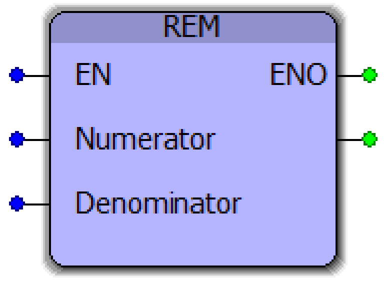
ATAN2 used with various output formats:







REM



This function block returns the modulo division result of two LREAL inputs. It is useful for determining the position within a MachineCycle.

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	This output will be high if EN is high and thisfunction is able to calculate the remainder with no errors.	
V	REM	BOOL	This output contains the calculated remainder	

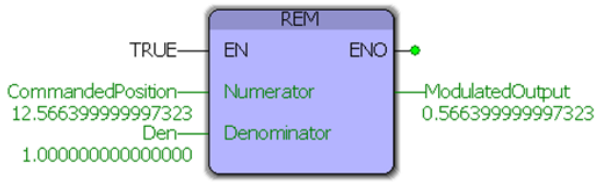
Error Description

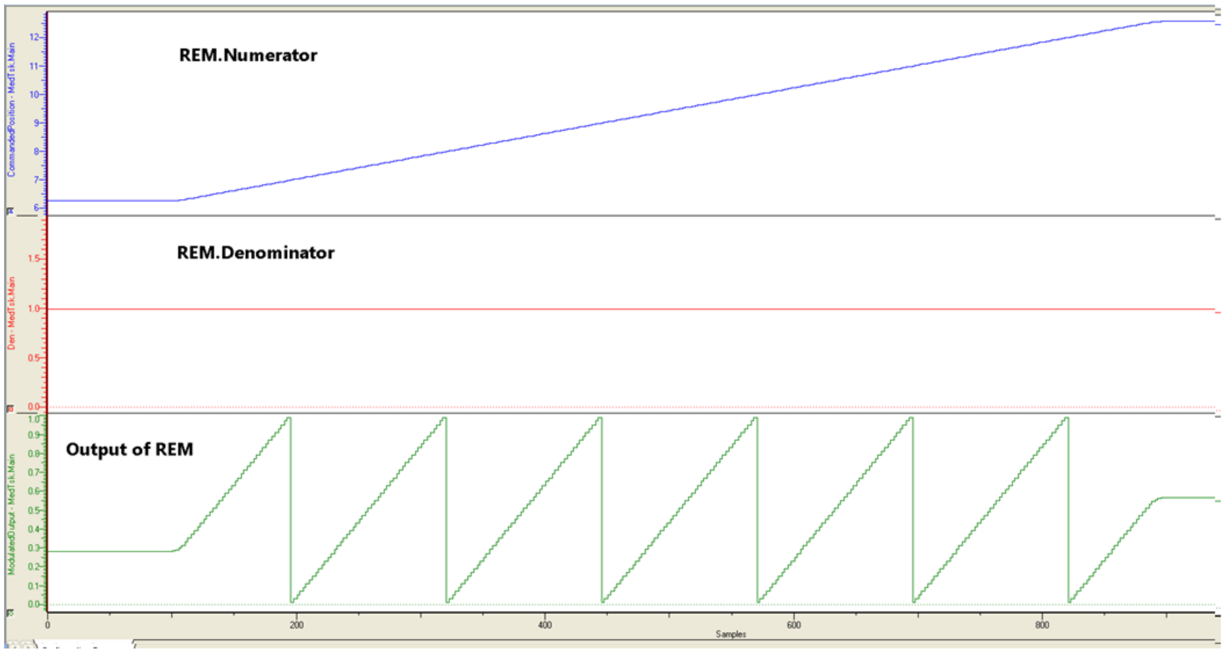
This is a function, not a function block and only provides one output. If ENO is not high when EN is high, this function cannot calculate the remainder. Verify that the Denominator is not zero.

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





Pack ML Toolbox

Getting Started: PackML

Requirements for v202

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

Firmware libraries:

- PROCONOS

User libraries:

- Math_Toolbox (v201 or higher)
- Yaskawa_Toolbox (v201 or higher)

Using the PackML Toolbox

See Yaskawa's Understanding PackML Webinar for an in depth look into this toolbox.

PackML Revision History

Current Version:

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

Previous Versions:

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

Enumerated Types

Enumerated Type: PackMLState

ENUM Type for indicating the PackML state.

Data Type Declaration

```
PackMLState:(Undefined, Clearing, Stopped, Starting, Idle, Suspended,
Execute, Stopping, Aborting, Aborted, Holding, Held, UnHolding, Suspending,
UnSuspending, Resetting, Completing, Complete);
```

- (* Defined for PackMLState*)
- (* 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 *)

Enumerated Type: PackMLState

ENUM Type for indicating the PackML state.

Data Type Declaration

```
PackMLState:(Undefined, Clearing, Stopped, Starting, Idle, Suspended,
Execute, Stopping, Aborting, Aborted, Holding, Held, UnHolding, Suspending,
UnSuspending, Resetting, Completing, Complete);
```

- (* Defined for PackMLState*)
- (* 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 *)

DataTypes

Data Type: PackML_Commands_STRUCT

Supporting structure for PackTags_Commands_STRUCT

Data Type Declaration

```
PackML_Commands_STRUCT :  STRUCT

Mode : DINT;          (*  Mode command, Mode's can be customized according to the PackML standard or for the
user's needs. See template documentation for more on mode customization *)

Reset : BOOL;         (*  Command to Reset the Machine  *)

Start : BOOL;         (*  Command to Start the Machine  *)

Stop : BOOL;          (*  Command to Stop the Machine   *)

Hold : BOOL;          (*  Command to Hold the Machine   *)

UnHold : BOOL;        (*  Command to UnHold the Machine  *)

Suspend : BOOL;       (*  Command to Suspend the Machine  *)

UnSuspend : BOOL;     (*  Command to UnSuspend the Machine  *)

Abort : BOOL;         (*  Command to Abort the Machine  *)

Clear : BOOL;         (*  Command to Clear the Machine  *)

StateComplete : BOOL; (*  Command to enter the Completing State  *)

END_STRUCT;
```

Data Type: EquipmentModule_STRUCT

Supporting data type used by EquipmentModule_ARRAY.

Data Type Declaration

```
EquipmentModule_STRUCT: STRUCT

EnabledCMs : INT;              (*  Number of enabled Control Modules contained in the Equipment Module  *)

CMs_Active : WORD;             (*  Every bit in this word indicates if a control module is active  *)

CMs_NotDone : WORD;            (*  Every bit in this word indicates if a control module is done  *)

CM_InactiveMask : WORD;        (*  Every bit in this word indicates if a control module is Inactive  *)

CM : ControlModule_ARRAY;      (*  Array containing the Commands, Status and Active bits for the 16 Control
Modules contained in the Equipment module *)

Cmd_Reset : BOOL;              (*  Command to Reset the machine  *)

Sts_Resetting_SC : BOOL;       (*  When set, the machine is in the resetting state  *)

Cmd_Start : BOOL;              (*  Command to Start the machine  *)

Sts_Starting_SC : BOOL;        (*  When set, the machine is in the Starting state  *)

Cmd_Stop : BOOL;               (*  Command to Stop the machine  *)

Sts_Stopping_SC : BOOL;        (*  When set, the machine is in the Stopping state  *)

Cmd_Hold : BOOL;               (*  Command to Hold the machine  *)

Sts_Holding_SC : BOOL;         (*  When set, the machine is in the Holding state  *)

Cmd_UnHold : BOOL;             (*  Command to Unhold the machine  *)

Sts_UnHolding_SC : BOOL;       (*  When set, the machine is in the UnHolding state  *)

Cmd_Suspend : BOOL;            (*  Command to Suspend the machine  *)

Sts_Suspending_SC : BOOL;      (*  When set, the machine is in the Suspending state  *)

Cmd_UnSuspend : BOOL;          (*  Command to UnSuspend the machine  *)

Sts_UnSuspending_SC : BOOL;    (*  When set, the machine is in the UnSuspending state  *)

Cmd_Abort : BOOL;              (*  Command to Abort the machine  *)
```

```
Sts_Aborting_SC : BOOL;       (*  When set, the machine is in the Aborting state  *)

Cmd_Clear : BOOL;             (*  Command to Clear the machine  *)

Sts_Clearing_SC : BOOL;       (*  When set, the machine is in the Clearing state  *)

Sts_Executing_SC : BOOL;      (*  When set, the machine is in the Executing state  *)

Cmd_StateComplete : BOOL;     (*  Command to enter the Completing State  *)

Sts_Completing_SC : BOOL;     (*  When set, the machine is in the Completing state  *)

ModuleActive : BOOL;          (*  Indicates if the module is active to receive commands  *)

END_STRUCT;
```

Data Type: PackML_States_STRUCT

Supporting structure for PackTags_Status_STRUCT

Data Type Declaration

```
PackML_States_STRUCT :  STRUCT

Clearing : BOOL;        (*  Indicates the machine is in the Clearing State  *)

Stopped : BOOL;         (*  Indicates the machine is in the Stopped State  *)

Starting : BOOL;        (*  Indicates the machine is in the Starting State  *)

Idle : BOOL;            (*  Indicates the machine is in the Idle State  *)

Suspended : BOOL;       (*  Indicates the machine is in the Suspended State  *)

Execute : BOOL;         (*  Indicates the machine is in the Execute State  *)

Stopping : BOOL;        (*  Indicates the machine is in the Stopping State  *)

Aborting : BOOL;        (*  Indicates the machine is in the Aborting State  *)

Aborted : BOOL;         (*  Indicates the machine is in the Aborted State  *)

Holding : BOOL;         (*  Indicates the machine is in the Holding State  *)

Held : BOOL;            (*  Indicates the machine is in the Held State  *)

UnHolding : BOOL;       (*  Indicates the machine is in the UnHolding State  *)

Suspending : BOOL;      (*  Indicates the machine is in the Suspending State  *)

UnSuspending : BOOL;    (*  Indicates the machine is in the UnSuspending State  *)

Resetting : BOOL;       (*  Indicates the machine is in the Resetting State  *)

Completing : BOOL;      (*  Indicates the machine is in the Completing State  *)

Complete : BOOL;        (*  Indicates the machine is in the Complete State  *)

END_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: UNitMachine_STRUCT

Contains all the information about the machine's current state for each EM and CM

Data Type Declaration

```
UNitmachine_STRUCT: STRUCT

PackML_StateControlReady : BOOL; (* Indicates when the PackML_State_Diagram function block is ready to
control the machine *)

EnabledEMs : INT; (* Number of enabled equipment modules in the machine *)

EMs_Active : WORD; (* Every bit in this word indicates which equipment modules are Active *)

EMs_NotDone : WORD; (* Every bit in this word indicates which equipment modules are Not Done*)

EM_InactiveMask : WORD; (* Every bit in this word indicates which equipment modules are Inactive *)

EM : EquipmentModule_ARRAY; (* Array containing the Commands, Status and Active bits for the 16 Equipment
Modules contained in the Machine*)

Sts_Resetting_SC : BOOL; (* When set, the machine is in the resetting state *)

Sts_Starting_SC : BOOL; (* When set, the machine is in the Starting state *)

Sts_Stopping_SC : BOOL; (* When set, the machine is in the Stopping state *)

Sts_Holding_SC : BOOL; (* When set, the machine is in the Holding state *)

Sts_UnHolding_SC : BOOL; (* When set, the machine is in the UnHolding state *)

Sts_Suspending_SC : BOOL; (* When set, the machine is in the Suspending state *)

Sts_UnSuspending_SC : BOOL; (*When set, the machine is in the UnSuspending state*)

Sts_Aborting_SC : BOOL; (* When set, the machine is in the Aborting state *)

Sts_Clearing_SC : BOOL; (* When set, the machine is in the Clearing state *)

Sts_Executing_SC : BOOL; (* When set, the machine is in the Executing state *)

Sts_Completing_SC : BOOL; (* When set, the machine is in the Completing state *)

END_STRUCT;
```

Data Type: PackTags_Admin_STRUCT

Data Type Declaration

```
PackTags_Admin_STRUCT : STRUCT

Alarm : EventHistoryArray; (* Array of Event information *)

StateCurrentTime : DINT; (* Amount of time spent in the current state *)

StateCumulativeTime : StateCumulativeArray; (* Array containing all the times spent in the different states *)

ModeCurrentTime : DINT; (* Amount of time spent in the current mode *)

ModeCumulativeTime : DINT_Array32; (* Array containing all the times spent in the different modes *)

AccumTimeSinceReset : DINT; (* Time since the cumulative and current times have been reset *)

ResetAllTimes : BOOL; (* Command to reset all timers *)

ResetCurrentModeTimes : BOOL; (* Command to reset all Current Times being tracked *)

TimeRollover : BOOL; (* Warning when the timer is approaching a roll over *)

ProdProcessed : DINT; (* Cumulative number of primary packages processed since the machine’s counters and
timers were reset *)

DefectiveProd : DINT; (* Cumulative number of defective packages processed since the machine’s counters and
timers were reset *)

ReWorkProd : DINT; (* Cumulative number of re-workable primary packages processed *)

UpstreamMessage : DINT;

DownstreamMessage : DINT;

CurrentUpstreamNodeID : DINT;

CurrentDownstreamNodeID : DINT;

END_STRUCT;
```

Data Type: PackTags_Status_STRUCT

Data Type Declaration

```
PackTags_Status_STRUCT :  STRUCT

CommandRejected : BOOL; (* If an invalid request is given and rejected, this bit will be set *)

UnitModeCurrent : DINT; (*Current Machine Mode*)

UnitModeCurBit : DWORD; (*Current Machine Mode Bit*)

UnitModeCurrentName : STRING; (*Current Machine Mode Name*)

UnitModeRequested : BOOL; (*[1 = Acknowledges that a unit mode change has been requested]*)

UnitModeChangeInProcess : BOOL; (*[1 = Requested unit mode change in process]*)

ProcModeCurrent : DINT; (*Current Procedure Mode*)

ProcModeRequested : BOOL; (*[1 = Acknowledges that a procedure mode change has been requested]*)

ProcModeChangeInProcess : BOOL; (*[1 = Requested procedure mode change in process]*)

StateCurrent : DINT; (*Current Machine State*)

StateCurBit : DWORD;

StateCurrentName : STRING; (*Current Machine State Name*)

StateRequested : BOOL; (*[1 = Acknowledges that a state change has been requested]*)

StateChangeInProcess : BOOL; (*[1 = Requested state change in process]*)

StateChangeProgress : DINT; (* Percent Complete of current state *)

StateLastCompleted : DINT; (* Machine state last completed *)

SeqNumber : DINT;

CurMachSpd : DINT; (*Current Machine Speed In Primary Line Packages Per Minute*)

MatReady : DWORD; (*Material Interlocks*)

MatLow : DWORD; (*Material Interlocks*)

MachDesignSpeed : REAL; (* Speed the machine is designed to operate at in it's installed environment *)

State : PackML_States_STRUCT;
```

ModeChangeNotAllowed : BOOL; (* This bit is set if an invalid mode change is requested and ignored *)

MachCycle : DINT; (* Indicates the number of completed machine cycles with or without product *)

ProdRatio : DINT; (* Quantity of primary packages per current package being produced *)

Dirty : BOOL; (* Set when the machine becomes dirty and machine must run through a cleaning cycle before production continues *)

Clean : BOOL; (* Bit is set after a cleaning cycle and reset once production begins again *)

TimeToDirty : DINT; (* Time remaining until machine becomes dirty again *)

EquipmentAllocatedToUnitModeID : DINT; (* Allocating a machine to operating a different mode than another duplicate machine *)

MachineReusableForUnitModeID : DINT; (* Indicates machine does not require immediate cleaning and can resume production in a specific time window *)

MachineReusableTimeLeft : DINT; (* Amount of time left for a system to be reusable for a specific Unit mode *)

MachineStoringProductID : DINT; (* For machines that have a storing capability *)

MachineTransferringProductID : DINT; (* For machines used in conveying, compacting and/or separating product and transferring it to other machinery *)

(* THE FOLLOWING FIELDS COME INITIALLY COMMENTED OUT TO SAVE MEMORY WHEN NOT USED *)

(* Node : Node_ARRAY; (*Node (machine) interface & ID structure*)

(* ProcessVariables : ProcessVariable_ARRAY; (* Machine Engineering Parameters *)

(* Product : Product_ARRAY; (* Machine Product/Recipe Parameters *)

(* Limits : Limit_ARRAY; (* Machine Parameter Prograble Limits *)

END_STRUCT;

Data Type: PackTags_Commands_STRUCT

Data Type Declaration

```
PackTags_Commands_STRUCT   : STRUCT

UnitMode : DINT; (*Unit Mode Commanded*)

UnitModeChangeRequest : BOOL; (*[1 = Change Machine Mode to Commanded Value]*)

ProcMode : DINT; (*Procedure Mode Commanded*)

ProcModeChangeRequest : BOOL; (*[1 = Change Procedure Mode to Commanded Value]*)

CurMachSpeed : DINT; (*Machine Speed - In Primary Line Packages*)

MatReady : DWORD; (*Material Interlocks*)

MatLow : DWORD; (*Material Interlocks*)

ResetPackMLTimes : BOOL; (*[1 = Reset PackML Current Mode and State Current/Cumulative Times]*)

CntrlCmd : DINT; (* provides an alternate method of moving through the state diagram *)

StateCmd : PackML_Commands_STRUCT; (* A structure for Coordinating machine nodes *)

StateChangeRequest : BOOL; (* Indicates the state machine should proceed to the target state *)

CfgRemoteCmdEnable : BOOL;

RemoteModeCmd : DINT;

RemoteModeCmdChgReq : BOOL;

RemoteStateCmd : DINT;

RemoteStateCmdChgReq : BOOL;

TargetDownstreamNodeID : DINT;

TargetUpstreamNodeID : DINT;

ChangeNodeServicedUpstream : DINT;

ChangeNodeServicedDownstream: DINT;


(* THE FOLLOWING FIELDS COME INITIALLY COMMENTED OUT TO SAVE MEMORY WHEN NOT USED *)
(* Node : Node_ARRAY; (*Node (machine) interface & ID structure*)

(* ProcessVariables : ProcessVariable_ARRAY; (* Machine Engineering Parameters *)

(* Product : Product_ARRAY; (* Machine Product/Recipe Parameters *)

(* Limits : Limit_ARRAY; (* Machine Parameter Prograble Limits *)

END_STRUCT;
```

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: PackML_Module_Commands_STRUCT

Supporting data type used by ControlModule_ARRAY

Data Type Declaration

```
PackML_Module_Commands_STRUCT: STRUCT

Cmd_Reset : BOOL; (* Command to Reset the machine *)

Sts_Resetting_SC : BOOL; (* When set, the machine is in the resetting state *)

Cmd_Start : BOOL; (* Command to Start the machine *)

Sts_Starting_SC : BOOL; (* When set, the machine is in the Starting state *)

Cmd_Stop : BOOL; (* Command to Stop the machine *)

Sts_Stopping_SC : BOOL; (* When set, the machine is in the Stopping state *)

Cmd_Hold : BOOL; (* Command to Hold the machine *)

Sts_Holding_SC : BOOL; (* When set, the machine is in the Holding state *)

Cmd_UnHold : BOOL; (* Command to Unhold the machine *)

Sts_UnHolding_SC : BOOL; (* When set, the machine is in the UnHolding state *)

Cmd_Suspend : BOOL; (* Command to Suspend the machine *)

Sts_Suspending_SC : BOOL; (* When set, the machine is in the Suspending state *)

Cmd_UnSuspend : BOOL; (*Command to UnSuspend the machine *)

Sts_UnSuspending_SC : BOOL; (* When set, the machine is in the UnSuspending state *)

Cmd_Abort : BOOL; (* Command to Abort the machine *)

Sts_Aborting_SC : BOOL; (* When set, the machine is in the Aborting state *)

Cmd_Clear : BOOL; (* Command to Clear the machine *)

Sts_Clearing_SC : BOOL; (* When set, the machine is in the Clearing state *)

Sts_Executing_SC : BOOL; (* When set, the machine is in the Executing state *)

Cmd_StateComplete : BOOL; (* Command to enter the Completing State *)

Sts_Completing_SC : BOOL; (* When set, the machine is in the Completing state *)
```

```
ModuleActive : BOOL; (* Indicates if the module is active to receive commands *)

END_STRUCT;
```

Data Type: Parameter_STRUCT

Supporting Structure for Parameter_ARRAY

Data Type Declaration

```
Parameter_STRUCT : STRUCT

ID : DINT; (* ID value assigned to the parameter *)

Name : STRING; (* Literal description of the parameter *)

Unit : STRING_5; (*  Unit associated with the given parameter *)

Value : REAL; (*Numeric value associated with the given parameter *)

END_STRUCT;
```

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: ProcessVariable_STRUCT

Supporting structure for ProcessVariable_ARRAY

Data Type Declaration

```
ProcessVariable_STRUCT : STRUCT

ID : DINT; (* ID value assigned to the parameter *)

Name : STRING; (* Literal description of the parameter,  can also be displayed on an HMI screen *)

Unit : STRING_5; (*  Unit associated with the given parameter, can also be displayed on an HMI screen *)

Value : REAL; (*Numeric value associated with the given parameter, can also be displayed on an HMI screen *)

END_STRUCT;
```

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: Node_STRUCT

Supporting structure for Node_ARRAY.

Data Type Declaration

```
Node_STRUCT : STRUCT

Number : INT; (* A chosen unique number of the Upstream/Downstream PackML machine *)

ControlCmdNumber : INT; (* User defined command to be sent from one node on the network to another *)

CmdValue : INT; (* A value to be associated with the ControlCmdNumber such as speed, or the mode requested to change to *)

Parameter : Parameter_ARRAY; (* An array of parameter names, values, and units of the parameter *)

END_STRUCT;
```

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: Ingredient_STRUCT

A structure of parameters containing information for a specific ingredient. Support structure for Ingredient_ARRAY.

Data Type Declaration

```
Ingredient_STRUCT : STRUCT

ID : INT; (* ID value assigned to the ingredient *)

Parameter : Parameter_ARRAY; (* An array of parameters used for the specified Ingredient *)

END_STRUCT;
```

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: Product_STRUCT

A structure containing product information

Data Type Declaration

```
Product_STRUCT : STRUCT
```

ProductID : INT; (* Used to indicate to the machine what product it is producing, also displayed on all HMI screens *)

ProcessVariables : ProcessVariable_ARRAY; (* Array of information containing parameters for multiple machines *)

Ingredients : Ingredient_ARRAY; (* An array containing all information regarding an ingredient *)

END_STRUCT;

Data Type: Product_ARRAY

An array containing product information

Data Type Declaration

```
Product_ARRAY : ARRAY[0..9] OF Product_STRUCT;
```

Data Type: Limit_STRUCT

Supporting structure for Limit_ARRAY.

Data Type Declaration

```
Limit_STRUCT : STRUCT

ID : INT; (* User defined ID for the limit, 0000 reserved for no limit assigned *)

Name : STRING; (* Literal name for the limit *)

Unit : STRING_5; (* Unit of the limit value *)

Value : REAL; (* Value assigned to the limit *)

END_STRUCT;
```

Data Type: Limit_ARRAY

An array containing user defined machine limits.

Data Type Declaration

```
Limit_ARRAY : ARRAY[0..9] OF Limit_STRUCT;
```

Supporting Arrays

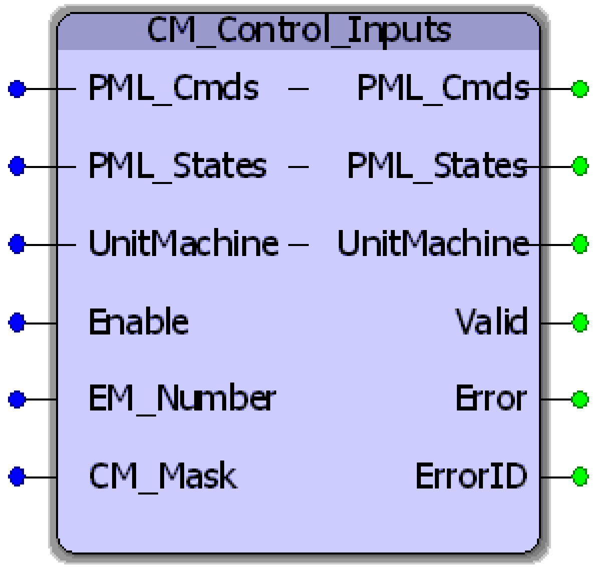
Arrays used by function blocks and other data types in the PackML Toolbox.

Data Type Declaration

```
TYPE

    DINT_Array18 : ARRAY[0..17] OF DINT;

    DINT_Array32 : ARRAY[0..31] OF DINT;

    DINT_Array7 : ARRAY[0..6] OF DINT;

    STRING_Array32 : ARRAY[0..31] OF STRING;

    STRING_Array18 : ARRAY[0..17] OF STRING;

    StateCumulativeArray : ARRAY[0..6] OF DINT_Array18; (* Default to max 6 Modes. Increase up to ..31 if
more Modes are defined *)

    STRING_5 : STRING(5);

    STRING_40 : STRING(40);

    STRING_200 : STRING(200);

    BOOL_16 : ARRAY[0..15] OF BOOL;

END_TYPE
```

Function Blocks

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.

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 the machine is operating in	
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 while the enable is held high	FALSE
V	EM_Number	INT	The EM number corresponding to the EM in which this FB is located	0
V	CM_Mask	WORD	Mask to deactivate CMs. When a CM is	16#0000

			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	
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Enable' goes low.	

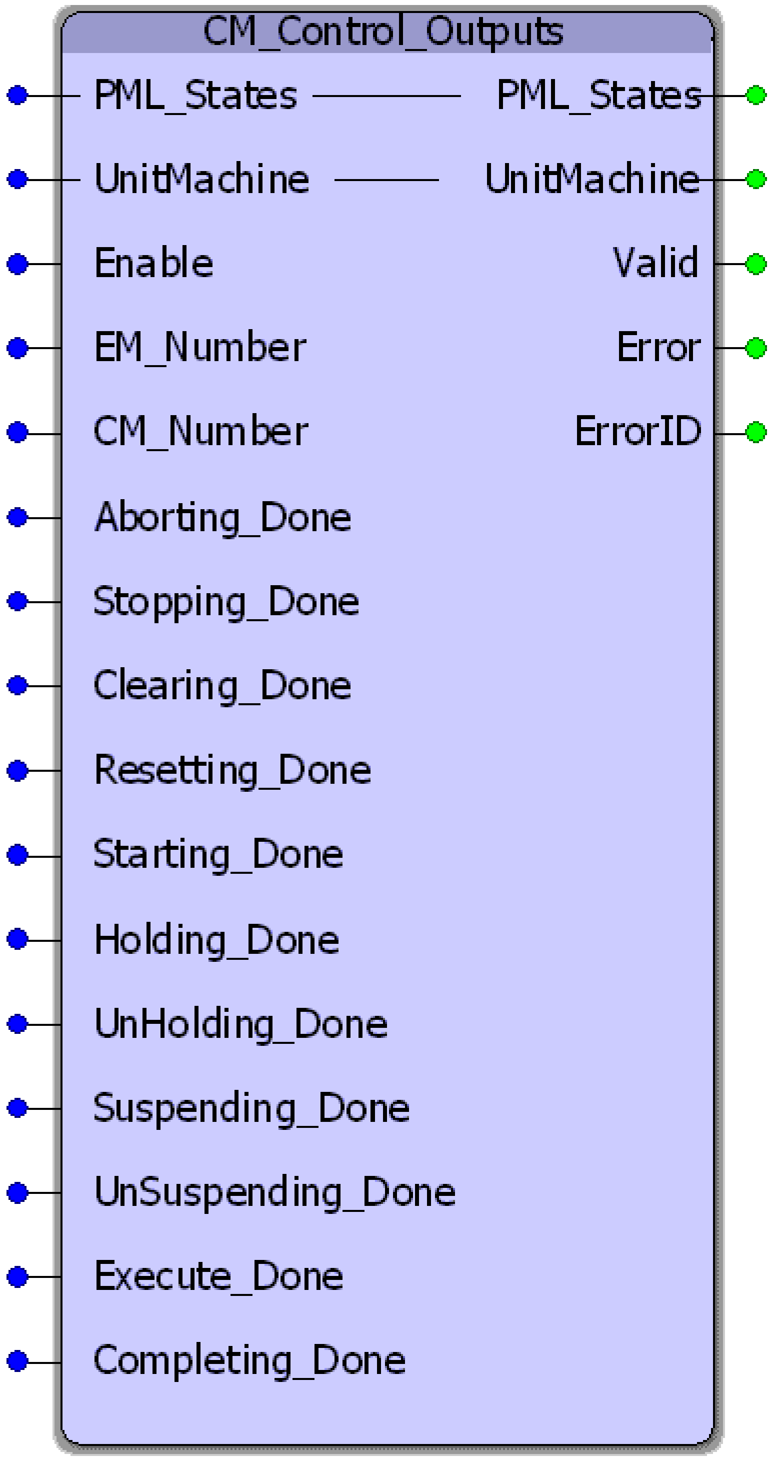
Notes

- 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

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.

Parameters

*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	PML_States	PackML_States_STRUCT	Structure containing information about the current state the machine is operating in	
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 while the enable is held high	FALSE
V	EM_Number	INT	The EM number corresponding to the EM in which this FB is located	0
V	CM_Number	WORD	The CM number corresponding to the CM in which this FB is located	0
B	Aborting_Done	BOOL	Setting this bit indicates that the current CM is done 'Aborting' and is ready to move to the next state	FALSE
B	Stopping_Done	BOOL	Setting this bit indicates that the current CM is done 'Stopping' and is ready to move to the next state	FALSE
B	Clearing_Done	BOOL	Setting this bit indicates that the current CM is done 'Clearing' and is ready to move to the next state	FALSE
B	Resetting_Done	BOOL	Setting this bit indicates that the current CM is done 'Resetting' and is ready to move to the next state	FALSE
B	Starting_Done	BOOL	Setting this bit indicates that the current CM is done 'Starting' and is ready to move to the next state	FALSE
B	Holding_Done	BOOL	Setting this bit indicates that the current CM is done 'Holding' and is ready to move to the next state	FALSE
B	UnHolding_Done	BOOL	Setting this bit indicates that the current CM is done	FALSE

			'UnHolding' and is ready to move to the next state	
B	Suspending_Done	BOOL	Setting this bit indicates that the current CM is done 'Suspending' and is ready to move to the next state	FALSE
B	UnSuspending_Done	BOOL	Setting this bit indicates that the current CM is done 'UnSuspending' and is ready to move to the next state	FALSE
B	Execute_Done	BOOL	Setting this bit indicates that the current CM is done 'Executing' and is ready to move to the next state	FALSE
B	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 outputs of the function are valid	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Enable' goes low.	

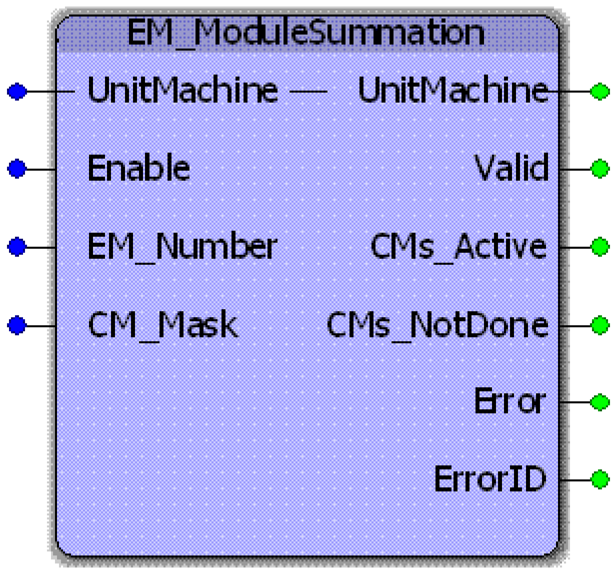
Notes

- 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

EM_ModuleSummation



The EM_Module_Summation function block rolls up all the Control Module State Complete bits for active and enabled CMs. The result is an overall Equipment Module State Complete bit that is transferred to the UN_ModuleControl Worksheet.

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	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while the enable is held high	FALSE
V	EM_Number	INT	The EM number corresponding to the EM in which this FB is located	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 =TRUE)	16#0000
VAR_OUTPUT				

B	Valid	BOOL	Indicates that the outputs of the function are valid
B	CMs_Active	WORD	The list of active CMs. Same bit scheme as CM_Mask. (Example: if CMs_Active.X4 = TRUE
B	CMs_NotDone	WORD	A compilation of which Control Modules have not completed the transition task.
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Enable' goes low.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Enable' goes low.

Notes

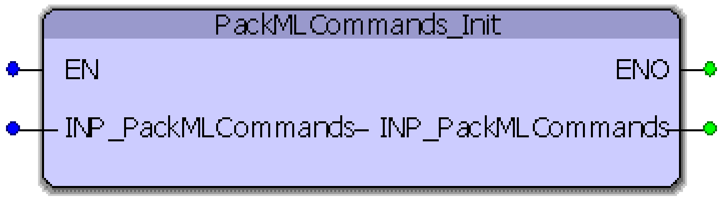
• The user can identify those CMs stuck in transition by comparing the outputs CMs_Active and CMs_NotDone. These outputs are of WORD datatype, with each bit [x] representing the active status and transition status of CM[x]. Example: If the PackML command STOP was given, and CM[1] was enabled and active, but not finished stopping yet, the output of CMs_Active would be ...00111 while the output of CMs_NotDone would be ...00010. The user then knows that the process is stuck in CM[1]. The user would then go to the EM00_CM00_Control_Outputs worksheet to further drill into the problem.

• 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

PackMLCommands_Init



The PackMLCommands_Init function block clears all commands and sets the machine to be in the stopped state.

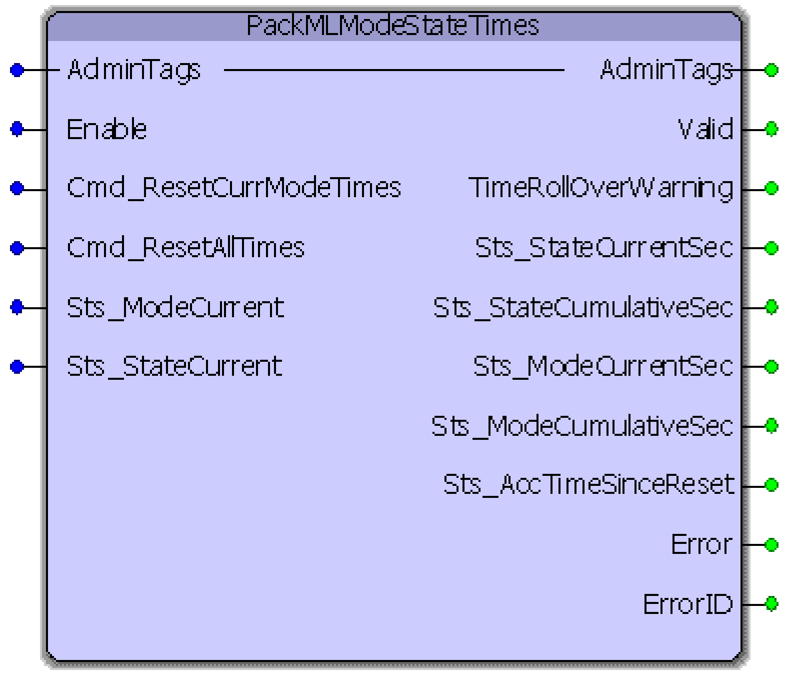
Parameters

*	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	The function will continue to execute while the enable is held high	FALSE
VAR_OUTPUT				
B	ENO	BOOL	Indicates that the outputs of the function are valid	

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.

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 while the enable is held high	FALSE
B	Cmd_ResetCurrModeTimes	BOOL	When set, all time counting will be stalled and all of the times	FALSE

			being counted for the Sts_ModeCurrent will be cleared.	
B	Cmd_ResetAllTimes	BOOL	When set, all times being monitored will be reset to zero. Time counting will also be stalled as long as this input is held high	FALSE
V	Sts_ModeCurrent	DINT	The current mode the machine is operating in	0
V	Sts_StateCurrent	DINT	The current state the machine is operating in	0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid	
B	TimeRollOverWarning	BOOL	A warning is sent when any of the time accumulators is approaching rolling over	
B	Sts_StateCurrentSec	DINT	Time (in seconds) spent in the current state	
V	Sts_StateCumulativeSec	StateCumulativeArray	An array containing the times spent operating in different modes and states	
B	Sts_ModeCurrentSec	DINT	Time (in seconds) spent in the current mode	
V	Sts_ModeCumulativeSec	DINT_Array32	An array of times spent in each mode	
B	Sts_AccTimeSinceReset	DINT	Accumulated time since Cmd_ResetAllTimes went high or the program was stopped for any reason.	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Enable' goes low.	

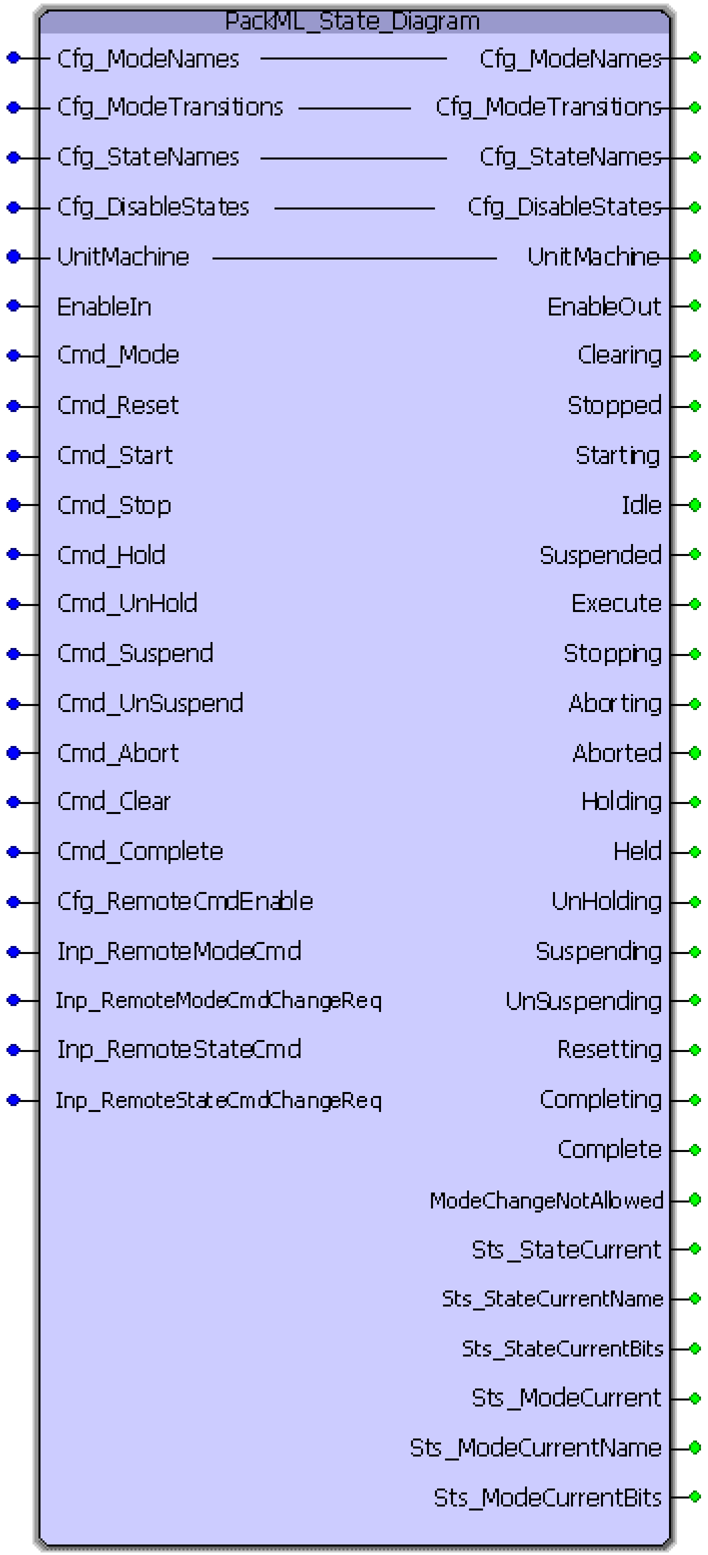
Notes

- See template documentation for further details on recommended usage.

Error Description

ErrorID	Meaning
0	No Error
12563	Time rollover warning

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.

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
B	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	0
B	Cmd_Reset	BOOL	Setting this bit sends the 'Restart' command to all enabled and active EMs if it is a legal transition from	FALSE
B	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
B	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
B	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
B	Cmd_UnHold	BOOL	Setting this bit sends the 'UnHold' command to all enabled and	FALSE

			active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	
B	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
B	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
B	Cmd_Abort	BOOL	Setting this bit sends the 'Abort' command to all enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	FALSE
B	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
B	Cmd_Complete	BOOL	Setting this bit sends the 'Complete' command to all	FALSE

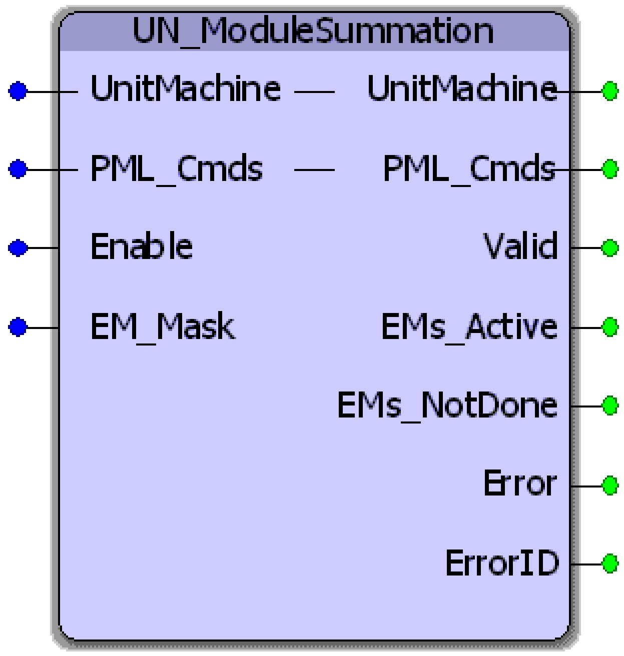
			enabled and active EMs if it is a legal transition from the current machine state, otherwise the command will be ignored	
V	Cfg_RemoteModeCmd	DINT	The remotely requested mode to transition to	0
B	Inp_RemoteModeCmdChangeReq	BOOL	When this input is set, the machine will transition to the mode set by Cfg_RemoteModeCmd 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
B	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	
B	Clearing	BOOL	When this bit is set, the machine is in the ‘Clearing’ state	
B	Stopped	BOOL	When this bit is set, the machine is in the ‘Stopped’ state	
B	Starting	BOOL	When this bit is set, the machine is in the ‘Starting’ state	
B	Idle	BOOL	When this bit is set, the machine is in the ‘Idle’ state	
B	Suspended	BOOL	When this bit is set, the machine is in the ‘Suspended’ state	
B	Execute	BOOL	When this bit is set, the machine is in the ‘Execute’ state	

B	Stopping	BOOL	When this bit is set, the machine is in the 'Stopping' state
B	Aborting	BOOL	When this bit is set, the machine is in the 'Aborting' state
B	Aborted	BOOL	When this bit is set, the machine is in the 'Aborted' state
B	Holding	BOOL	When this bit is set, the machine is in the 'Holding' state
B	Held	BOOL	When this bit is set, the machine is in the 'Held' state
B	UnHolding	BOOL	When this bit is set, the machine is in the 'UnHolding' state
B	Suspending	BOOL	When this bit is set, the machine is in the 'Suspending' state
B	UnSuspending	BOOL	When this bit is set, the machine is in the 'UnSuspending' state
B	Resetting	BOOL	When this bit is set, the machine is in the 'Resetting' state
B	Completing	BOOL	When this bit is set, the machine is in the 'Completing' state
B	Complete	BOOL	When this bit is set, the machine is in the 'Complete' state
B	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	DWORD	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.

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.

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 while the enable is held high	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	16#0000

			deactivate EM_3, set EM_Mask.X3 =TRUE)	
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid	
B	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)	
B	EMs_NotDone	WORD	A compilation of which Equipment Modules have not completed the transition task.	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Enable' goes low.	

Error Description

ErrorID	Meaning
0	No Error

PLCopen Toolbox

PLCopen Toolbox

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 definition.

Certain versions of this toolbox refer to the Math Toolbox for additional functionality. You must also include the Math Toolbox in your project to avoid compile errors.

See the PLCopen_Toolbox eLearning Modules on Yaskawa's Youtube channel for video tutorials and examples.

The PLCopen Toolbox consists of the following:

Data Types:

Data Type	Description
AXIS_REF	Identifies an axis
AxisParamData	Supporting structure for AxisPrmArray. Used by the ReadAxisParameters function block
AxisParameterStruct	For use with the ReadAxisParameters function block
AxisPrmArray	Used by the ReadAxisParameters function block
AxisStruct	For use as a container for all axis related data. (Customizable)
CAMSWITCH_ARRAY	Supporting structure for CAMSWITCH_REF. Used by the Y_DigitalCamSwitch function block
CAMSWITCH_REF	Used by the Y_DigitalCamSwitch function block
CAMSWITCH_STRUCT	Supporting structure for CAMSWITCH_ARRAY. Used by the Y_DigitalCamSwitch function block
HomeStruct	For use with any HOME_*** function block
IndividualParamDetails	Used by the ReadAxisParameters function block
LatchBufferArray	Supporting structure for ProductBufferStruct Used by the ReadAxisParameters function block
MoveStruct	For use with MC_MoveAbsolute, MC_MoveRelative, and MC_MoveVelocity
ProductBufferStruct	For use with the ProductBuffer function block
SWERROR_STRUCT	Used by the Y_DigitalCamSwitch function block
TRACK_ARRAY	Supporting structure for TRACK_REF. Used by the Y_DigitalCamSwitch function block
TRACK_REF	Used by the Y_DigitalCamSwitch function block
TRACK_STRUCT	Supporting structure for TRACK_ARRAY. Used by the Y_DigitalCamSwitch

	function block

Enumerated Types:

Enumerated Type	Description
MC_Direction	For use with the Reverse_MC_Direction function block to select positive or negative direction for LimitDirection and PulseDirection
TB_AxisType	For use with the ReadAxisParameters function block to select the type of axis, such as servo, external encoder, VFD, etc.

Programs:

Program	Description
Initialize	A template of code which can be copied to reduce the time required to enter initialization code into your project.

Function Blocks:

Function Block	Description
AbsolutePositionManager	This function block can serve as a central point for monitoring, clearing, and defining the position of an absolute encoder.
AccDecLimits	Manages the parameters associated with enabling/disabling the acceleration and deceleration limits.
Axes_Interlock	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.
AxisControl	Combines MC_Power, MCReadAxisError, and MC_Reset and provides separate outputs for controller and drive alarms and warnings.
AxisStatus	Uses MC_ReadAxisError to provide further breakdown of the ErrorClass and AxisErrorID.
ControllerAlarm	Provides a BOOL output to indicate if there is a controller alarm not related to an axis.
HighSpeedOutput	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.
Home_LS	Combines the PLCopen function blocks MC_StepLimitSwitch,

	MC_MoveRelative, and MC_SetPosition.
Home_LS_Pulse	Combines the PLCopen function blocks MC_StepLimitSwitch, MC_StepRefPulse, MC_MoveRelative, and MC_SetPosition.
Home_Pulse	Combines the PLCopen function blocks MC_StepRefPulse, MC_MoveRelative, and MC_SetPosition.
Jog	Combines the PLCopen functions MC_MoveVelocity and MC_Stop.
MoveRelative_ByTime	Converts the MoveTime input into acceleration, velocity, and deceleration.
PositionLimits	Enables or disables the position limit function.
ProductBuffer	Uses MC_TouchProbe and provides an array of recorded latch positions for the axis specified.
ReadAxisParameters	Reads all the commonly updated axis parameters that may be used within an application and copies them to an AxisParameterStruct.
Reverse_MC_Direction	Changes the enumerated type MC_Direction#positive_direction to MC_Direction#negative_direction or vice versa.
VelocityLimits	Enables or disables the velocity limit function.
Y_DigitalCamSwitch	Commands a group of discrete output bits analogous to a set of mechanical cam controlled switches driven by a rotating shaft.

Getting Started: PLCopen

Requirements for v205

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:

- DataTypes_Toolbox (v200 or higher)
- Math_Toolbox (v202 or higher)
  - Only required if using the ProductBuffer function block

PLCopen Revision History

Current Version:

New for PLCopen v205 – 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

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

Previous Versions:

(***************** 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 Types

Data Types for PLCopen Toolbox

The following is a complete list of all DataTypes included in the PCLopen toolbox. The list is arranged to separate those that are used internally, and not useful outside of their particular function, and those that an application program must incorporate when the programmer wishes to use the associated Function Block.

Data Type	Usage
DataTypes for use with function blocks in the PLCopen firmware library	
HomeStruct	For use with any HOME_*** function block
MoveStruct	For use with MC_MoveAbsolute, MC_MoveRelative, and MC_MoveVelocity
DataTypes for external use with the PLCopen Toolbox function blocks	
AXIS_REF	Identifies an axis
AxisParamStruct	For use with the CamSlave_FeedToLength and CamSlave_WindowCheck function blocks.
AxisStruct	For use as a container for all axis related data. (Customizable)
MC_Direction	ENUM type for indicating positive or negative direction for LimitDirection and PulseDirection of the Reverse_MC_Direction function block
ProductBufferStruct	For use with the ProductBuffer function block
TB_AxisType	ENUM type for indicating the axis type for the ReadAxisParameters function block
DataTypes that support other DataTypes (no need for direct use by the programmer)	
AxisParamData	Supporting structure for AxisPrmArray. Used by the ReadAxisParameters function block
CAMSWITCH_ARRAY	Supporting structure for CAMSWITCH_REF. Used by the Y_DigitalCamSwitch function block
CAMSWITCH_STRUCT	Supporting structure for CAMSWITCH_ARRAY. Used by the Y_DigitalCamSwitch function block.
LatchBufferArray	Supporting structure for ProductBufferStruct Used by the ReadAxisParameters function block
TRACK_ARRAY	Supporting structure for TRACK_REF. Used by the Y_DigitalCamSwitch function block
TRACK_STRUCT	Supporting structure for TRACK_ARRAY. Used by the Y_DigitalCamSwitch function block
DataTypes used internally by PLCopen Toolbox function blocks	
AxisPrmArray	Used by the ReadAxisParameters function block
CAMSWITCH_REF	Used by the Y_DigitalCamSwitch function block
IndividualParamDetails	Used by the ReadAxisParameters function block
SWERRORSTRUCT	Used by the Y_DigitalCamSwitch function block
TRACK_REF	Used by the Y_DigitalCamSwitch function block

Data Type: AXIS_REF

The AXIS_REF data type identifies an axis and thus provides the interface to the hardware or virtual axes. AXIS_REF is used as VAR_IN_OUT in all Motion Control Function Blocks described in this Online help. It is represented as an input and an output connected by a horizontal line in the graphical representation of a function block.

The value of AxisNum is determined by the logical axis number assigned in the Hardware Configuration. See the Configuration tab for each axis.

Data Type Declaration

TYPE

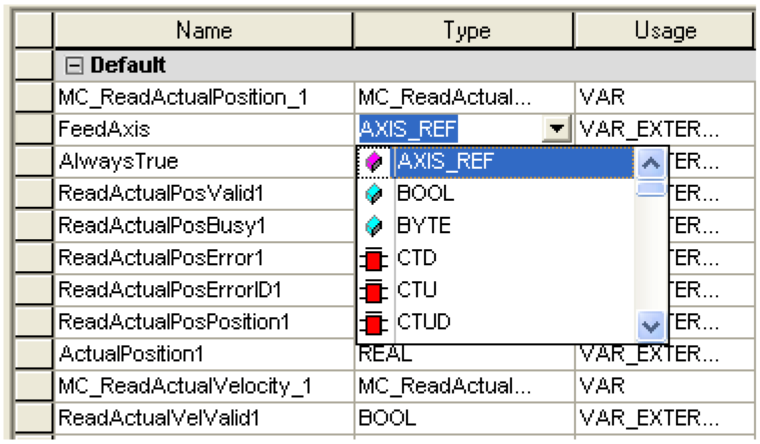
AXIS_REF:STRUCT

AxisNum:UINT;

END_STRUCT;

END_TYPE

Variable Declaration Example



Name	Type	Usage
⊟ **Default**		
MC_ReadActualPosition_1	MC_ReadActual...	VAR
FeedAxis	AXIS_REF	VAR_EXTER...
AlwaysTrue		...ER...
ReadActualPosValid1		...ER...
ReadActualPosBusy1		...ER...
ReadActualPosError1		...ER...
ReadActualPosErrorID1		...ER...
ReadActualPosPosition1		...ER...
ActualPosition1	REAL	VAR_EXTER...
MC_ReadActualVelocity_1	MC_ReadActual...	VAR
ReadActualVelValid1	BOOL	VAR_EXTER...

Code Example

```
AxisX.Number:=UINT#0;
MCMoveAbsoluteX(Axis:=AxisX, Execute:=FALSE);
AxisX:=MCMoveAbsolutX.Axis;
AxisY.Number:=UINT#0;
MCMoveAbsoluteY(Axis:=AxisY, Execute:=FALSE);
AxisX:=MCMoveAbsolutY.Axis;
```

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 ReadAxisParameters function block.

Data Type Declaration

```
TYPE

AxisParameterStruct:STRUCT

ActualPosition:LREAL;                (* 1000  *)

ActualPositionCyclic:LREAL;          (* 1005  *)

ActualPositionNonCyclic:LREAL;       (* 1006  *)

ActualTorque:LREAL;                  (* 1004  *)

ActualVelocity:LREAL;                (* 1001  *)

AtVelocity:BOOL;                     (* 1141  *)

BufferedMotionBlocks:LREAL;          (* 1600  *)

CamMasterCycle:LREAL;                (* 1512  *)

CamMasterPosition:LREAL;             (* 1500  *)

CamMasterShiftedCyclic:LREAL;        (* 1502  *)

CamMasterShiftedPosition:LREAL;      (* 1501  *)

CamMasterScale:LREAL;                (* 1510  *)

CamMasterShift:LREAL;                (* 1511  *)

CamOffset:LREAL;                     (* 1531  *)

CamScale:LREAL;                      (* 1530  *)

CamShiftRemaining:LREAL;             (* 1513  *)

CamState:LREAL;                      (* 1540  *)

CamTableIDEngaged:LREAL;             (* 1541  *)

CamTableOutput:LREAL;                (* 1520  *)

CommandedAcceleration:LREAL;         (* 1012  *)
```

```
CommandedPosition:LREAL;          (*  1010   *)

CommandedPositionCyclic:LREAL;      (*  1015   *)

CommandedPositionNonCyclic:LREAL;   (*  1016   *)

CommandedTorque:LREAL;            (*  1014   *)

CommandedVelocity:LREAL;          (*  1011   *)

InPosition:BOOL;                  (*  1140   *)

LatchPositionNonCyclic:LREAL;       (*  1031   *)

PositionError:LREAL;              (*  1130   *)

PositionWindow:LREAL;             (*  1120   *)

END_STRUCT;

END_TYPE
```

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

```
TYPE

AxisStruct: STRUCT

Ref:AXIS_REF;         (*   Used with the Axis VAR_IN_OUT of many PLCopen function blocks   *)

JogSpeed:LREAL;       (*   In user units/sec as defined in the Hardware Configuration   *)

RunSpeed:LREAL;       (*   In user units/sec as defined in the Hardware Configuration   *)

Position:LREAL;       (*   In user units as defined in the Hardware Configuration   *)

Acceleration:LREAL;   (*   In user units/sec2 as defined in the Hardware Configuration   *)

Deceleration:LREAL;   (*   In user units/sec2 as defined in the Hardware Configuration   *)

Jerk:LREAL;           (*   In user units/sec/sec/sec as defined in the Hardware Configuration   *)

Status:BOOL;          (*   To indicate if the drive is enabled   *)

Warning:BOOL;

Alarm:BOOL;

DriveAlarmID:UINT;

DriveWarningID:UINT;

ControlAlarmID:UDINT;

Prm:AxisParameterStruct;

Home:HomeStruct;

Latch:RegistrationStruct;

Cam:CamStruct;

END_STRUCT;
```

```
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

TYPE

CAMSWITCH_REF:STRUCT

MasterType: INT; (* 0 = Infinite/Rotary, 1 = Finite/Linear *)

MachineCycle: LREAL;

(*This number should match the setting in the Hardware Configuration. Valid for Type = 0.*)

LastSwitch; INT; (* To limit the evaluation of the array *)

Switch:CAMSWITCH_ARRAY;

END_STRUCT;

END_TYPE

Data Type: CAMSWITCH_STRUCT

Supporting structure for CAMSWITCH_ARRAY. Used by the Y_DigitalCamSwitch function block.

Data Type Declaration

```
TYPE

CAMSWITCH_STRUCT:STRUCT

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. *)

FirstOnPosition:LREAL;

(* Lower boundary where the switch is ON. *)

LastOnPosition:LREAL;

(* Upper boundary where the switch is ON. If LastOnPosition < FirstOnPosition,

then the switch should be OFF between the positions (inverse cam switch) *)

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.  *)

CamSwitchMode:INT;

(* Position vs Time-Based output.  0 = Position. 1 = Time.  *)

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.*)

END_STRUCT;

END_TYPE
```

Data Type: HomeStruct

For use with all HOME_*** function blocks.

Data Type Declaration

```
TYPE

HomeStruct: STRUCT

Direction:INT;          (*  Used in conjunction with MC_StepLimit Function Block  *)

SwitchMode:INT;         (*  Configuration for action of the home sensor.  [See MC_SwitchMode]  *)

TorqueLimit:LREAL;      (*  Default if unused [ZERO] is 100.00% of rated torque   *)

ApproachVelocity:LREAL;

ApproachTimeLimit:LREAL; (*  In seconds  *)

ApproachDistanceLimit:LREAL;

AccDec:LREAL;

CreepVelocity:LREAL;

CreepTimeLimit:LREAL;   (*  In seconds  *)

CreepDistanceLimit:LREAL;

Offset:LREAL;           (*  Position offset to MOVE after finding the last input device

(sensor of C channel)  *)

OffsetVelocity:LREAL;

Position:LREAL;         (*  This is the location that will be defined when all homing

actions are complete, including the offset move.  *)

Homed:BOOL;             (*  Flag to indicate that the axis was successfully homed  *)

END_STRUCT;

END_TYPE
```

Data Type: IndividualParamDetails

Used by the ReadAxisParameters function block.

Data Type Declaration

```
IndividualParamDetails:STRUCT   (*   For internal use of the ReadAxisParameters Function Block   *)

Num:UINT;

BValue:BOOL;

DIValue:DINT;

LRValue:LREAL;

PType:INT;              (*  1=BOOL,  2=BYTE,  3=INT,  4=DINT,   5=LREAL   *)

AxisMask:WORD;

TypeMask:WORD;

END_STRUCT;
```

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

```
TYPE

MoveStruct:STRUCT

Position:LREAL;     (*   In user units as defined in the Hardware Configuration   *)

Velocity:LREAL;     (*   In user units/sec as defined in the Hardware Configuration   *)

Acceleration:LREAL; (*   In user units/sec2 as defined in the Hardware Configuration   *)

Deceleration:LREAL; (*   In user units/sec2 as defined in the Hardware Configuration   *)

Jerk:LREAL;         (*   In user units/sec/sec/sec as defined in the Hardware Configuration   *)

END_STRUCT;

END_TYPE
```

Data Type: ProductBufferStruct

For use with the ProductBuffer function block.

Data Type Declaration

TYPE

ProductBufferStruct: STRUCT

BufferSize:INT;   (* INPUT - Maximum number of registration marks to be tracked. (Circular buffer size). *)

BufferNonCyclic:LatchBufferArray; (* OUTPUT - Array (circular buffer) of all recorded registration marks (unmodularized latch values). *)

BufferCyclic:LatchBufferArray; (* OUTPUT - Array (circular buffer) of all recorded registration marks (modularized latch values). *)

Sensor:TRIGGER_REF;   (* INPUT - TRIGGER_REF for the axis which registration marks are to be detected. *)

SensorDistance:LREAL;   (* INPUT - Distance in units of the master axis from the registration sensor to the required synchronization point with a slave axis. *)

SensorOffset:LREAL;   (* INPUT - If the sensor is an exact multiple of machine cycles from cut position, this number would be zero. *)

ManualOffset:LREAL;   (* INPUT - Amount to adjust the synchronization point, typically comes from HMI. *)

LockoutDistance:LREAL;   (* INPUT - Distance after recording a latch that another latch would be ignored as potential noise. *)

ProductAwayDistance:LREAL;   (* INPUT - The distance the product travels from its initial detection until it is safely past the slave operation. *)

StorePointer:INT;   (* OUTPUT - Array Index of the latch data that was stored after MC_TouchProbe. *)

UsePointer:INT;   (* INPUT - Array Index of the latch data. *)

PrevUsePointer:INT;   (* INPUT - Array Index of the previous latch data. *)

```
END_STRUCT;

END_TYPE
```

NOTES:

INPUT - Indicates a value that the USER must supply to the ProductBuffer function block.

OUTPUT - Indicates a value that the ProductBuffer function block will write to the structure for use in the application.

The following structure values are not used by the ProductBuffer function block, but are defined in the ProductBufferStruct because typical applications that cam benefit from this function require this data for successful operation:

- SensorDistance
- SensorOffset
- ProductAwayDistance

Data Type: SWERROR_STRUCT

Used by the Y_DigitalCamSwitch function block

Data Type Declaration

```
TYPE

SWERROR_STRUCT: STRUCT

TrackNumber: INT;

(* The last switch number where an invalid setting for TrackNumber occurred *)

FirstOnPosition: INT;

(* The last switch number where an invalid setting for FirstOnPosition occurred *)

LastOnPosition: INT;

(* The last switch number where an invalid setting for LastOnPosition occurred *)

AxisDirection: INT;

(* The last switch number where an invalid setting for AxisDirection occurred *)

CamSwitchMode: INT;

(* The last switch number where an invalid setting for CamSwitchMode occurred *)

Duration: INT;

(* The last switch number where an invalid setting for Duration occurred *)

ImproperOnPosition: INT;

(* The last switch number where an improper relationship between FirstOnPosition

and LastOnPosition occurred *)

OnOffPositionError: INT;

(* The last switch number where the OnCompensationScaler and/or

OffCompensationScaler resulted in an improper relationship between

the modified FirstOn and LastOn positions. *)

END_STRUCT;
```

```
END_TYPE
```

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

```
TYPE

TRACK_STRUCT:STRUCT

OnCompensationScaler:LREAL;

(* Compensation for the FirstOnPosition of each switch on the track. + = advance, - = delay*)

OffCompensationScaler:LREAL;

(* SpeedCompensation  for the LastOnPosition of each switch on the track.*)

Value: BOOLEAN;

(* The resulting status of the track after evaluating and combining all switches that affect the track.*)

END_STRUCT;

END_TYPE
```

Enumerated Types

Enumerated Type: MC_Direction

ENUM type for indicating the axis type for the Reverse_MC_Direction function block.

Data Type Declaration

```
(*  ENUM Type for LimitDirection and PulseDirection  *)

MC_Direction:(positive_direction, negative_direction);
```

Enumerated Type: TB_AxisType

ENUM type for indicating the axis type for the ReadAxisParameters function block.

Data Type Declaration

```
(*  ENUM Type for AxisCode  *)

TB_AxisType:(Servo, VFD, Stepper, Virtual, External);
```

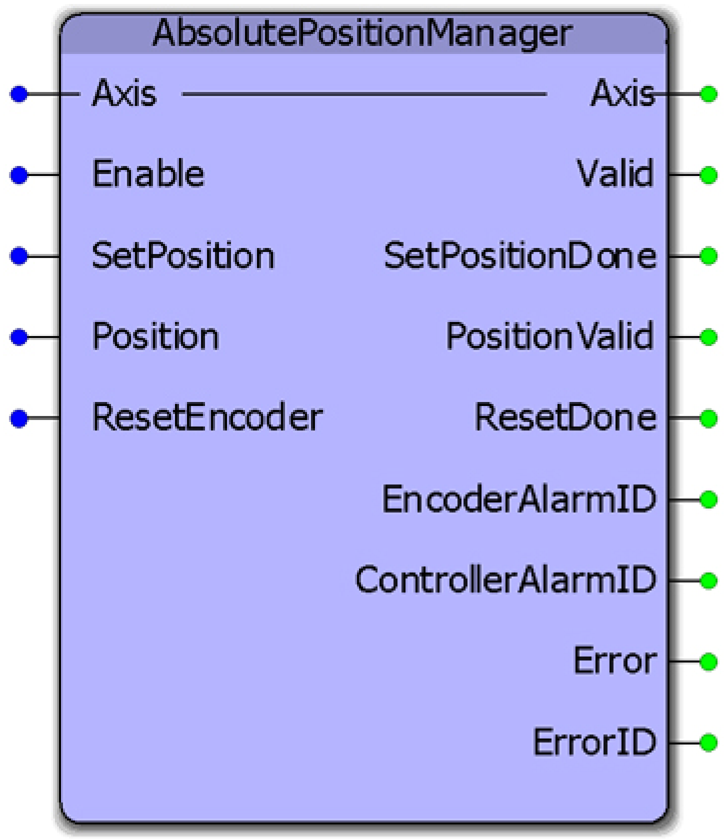
PTB_Initialize

This is not a function block but a Program POU in the 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.

Function Blocks

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.

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 while enable is held high.	FALSE
V	SetPosition	BOOL	Value of the absolute position [u] to be set when homing is done. The reference	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 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 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

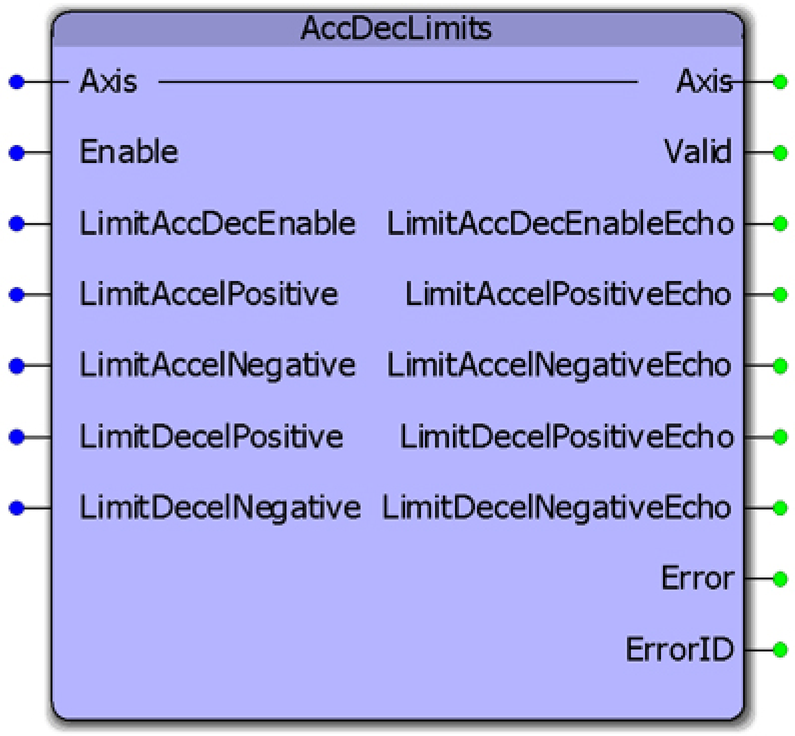
- Check the Hardware Configuration to ensure that the alarm format for Sigma III and higher drives is set for 3 digit alarm mode.
- See the AbsolutePositionManager eLearning Module on Yaskawa's YouTube channel.

Error Description

ErrorID	Meaning
4378	The function block is not applicable for the external axis specified
4380	MC_SetPosition can not be executed while the axis is moving.

4382	When the axis is in 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 the cam master of other axes.
4391	The function block cannot be used with a virtual axis.
4401	Axis latch function already in use.
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.
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.

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.

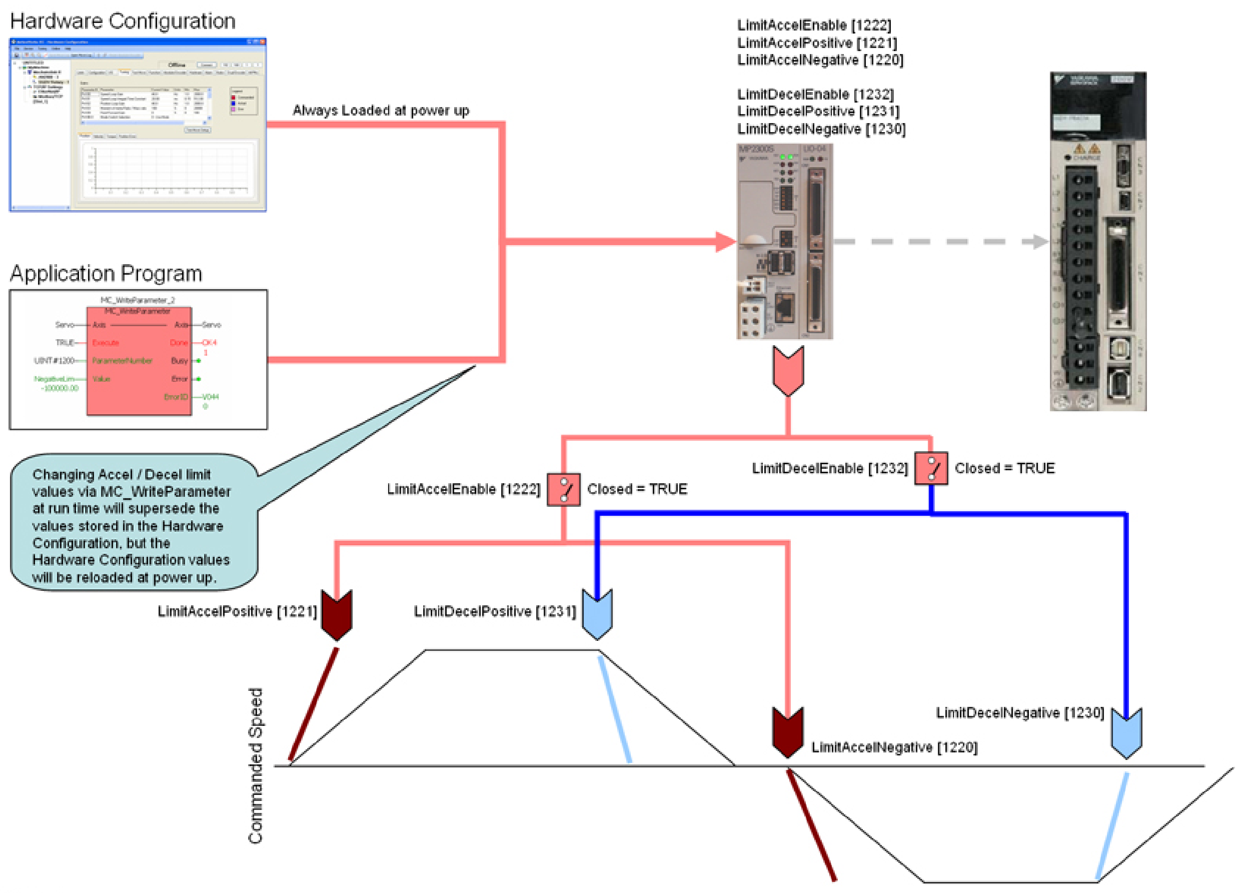
Parameters

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

Notes

The function block uses MC_ReadBoolParameter, MC_WriteBoolParameter, MC_ReadParameter, and MC_WriteParameter.


Hardware Configuration
Always Loaded at power up
Application Program
MC_WriteParameter_2
MC_WriteParameter
Servo
Axis
TRUE
Execute
Done
UINT#1200
ParameterNumber
Busy
NegativeLim
-100000.00
Value
Error
ErrorID
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.
LimitAccelEnable [1222]
Closed = TRUE
LimitDecelEnable [1232]
Closed = TRUE
LimitAccelPositive [1221]
LimitDecelPositive [1231]
LimitDecelNegative [1230]
LimitAccelNegative [1220]
Commanded Speed

Accel / Decel Limits

• The software acceleration & deceleration limits are managed by the MP2000iec 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 / sec^2.

• To disable the acceleration limit, set LimitAccelEnable, parameter 1222 to zero.

Deceleration Limits

• Deceleration is defined by decreasing velocity towards zero.

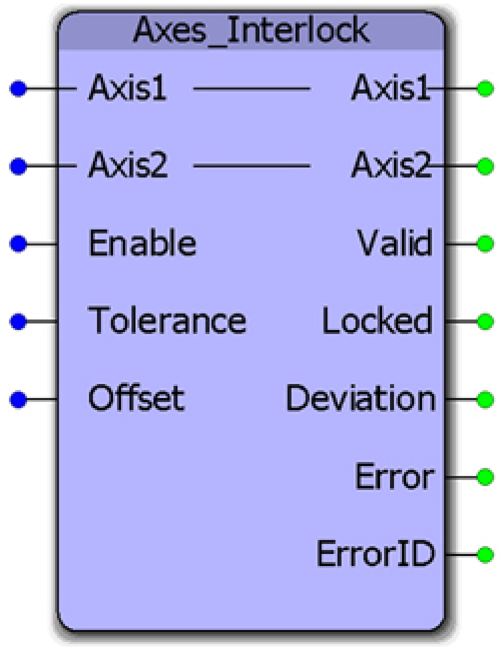
• The parameters are called LimitDecelPositive and LimitDecelNegative, with values of UINT#1231 and UINT#1230 respectively. Use the MC_WriteParameter function block for these and all controller side parameters. Deceleration limit parameters are in user units / sec^2.

• 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 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.
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.

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 output "Locked? 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.

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.	
B	Locked	BOOL	Indicates TRUE if neither axis has an alarm and the position deviation is less than the specified tolerance.	
B	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.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

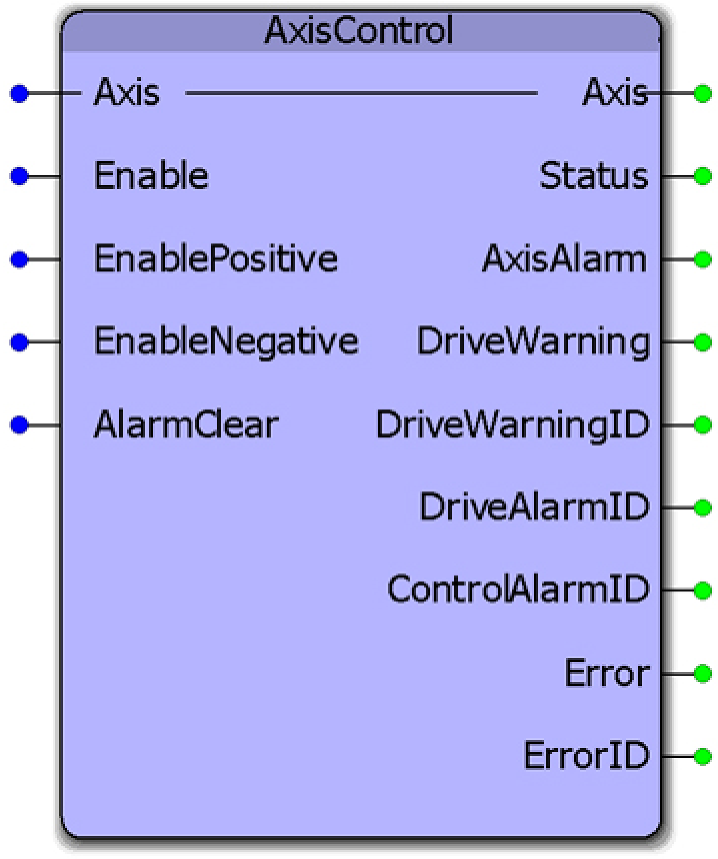
- It is assumed that the axes have the same user units because they are operating the same load.
- See the AxesInterlock eLearning Module on Yaskawa's YouTube channel.

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.
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.

e

AxisControl



This function block combines MC_Power, MC_ReadAxisError, and MC_Reset and provides separate outputs for controller and drive alarms and warnings.

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 while enable is held high.	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 the drive.	
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 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.

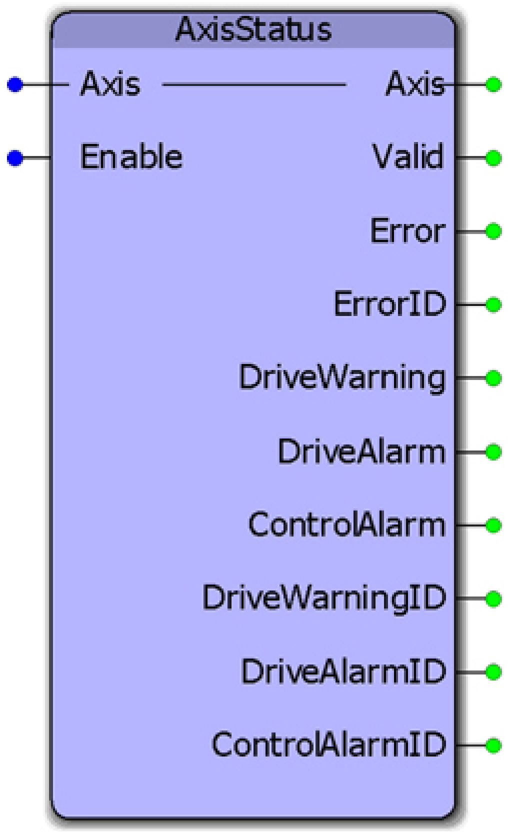
Remember that this function only reports axis specific alarms and warnings. For general system alarms, use the Y_ReadAlarms function block from the PLCopenPlus firmware library.

Error Description

ErrorID	Meaning
0	No error
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4371	The servo drive failed to enable or disable. Check the amplifier wiring for L1 / L2 / L3

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 (HHB) is preventing the drive from enabling.
4414	MECHATROLINK Communications to the drive was disrupted. Execute MC_Reset to restore the connection.
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.
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.
61713	An internal assertion in the motion kernel failed indicating the controller is not in a stable state. Please report this error to Yaskawa America Incorporated.

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.

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 while enable is held high.	FALSE

VAR_OUTPUT			
B	Valid	BOOL	Indicates that the outputs of the function are valid.
V	DriveWarning	BOOL	Indicates a warning on the drive, such as any A.9x display on the drive.
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 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.

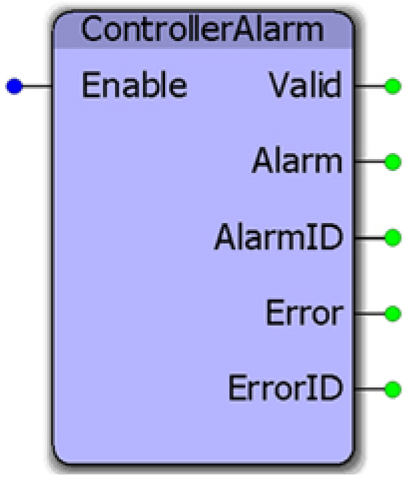
Notes

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. 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 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.
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.

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.

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 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 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.	

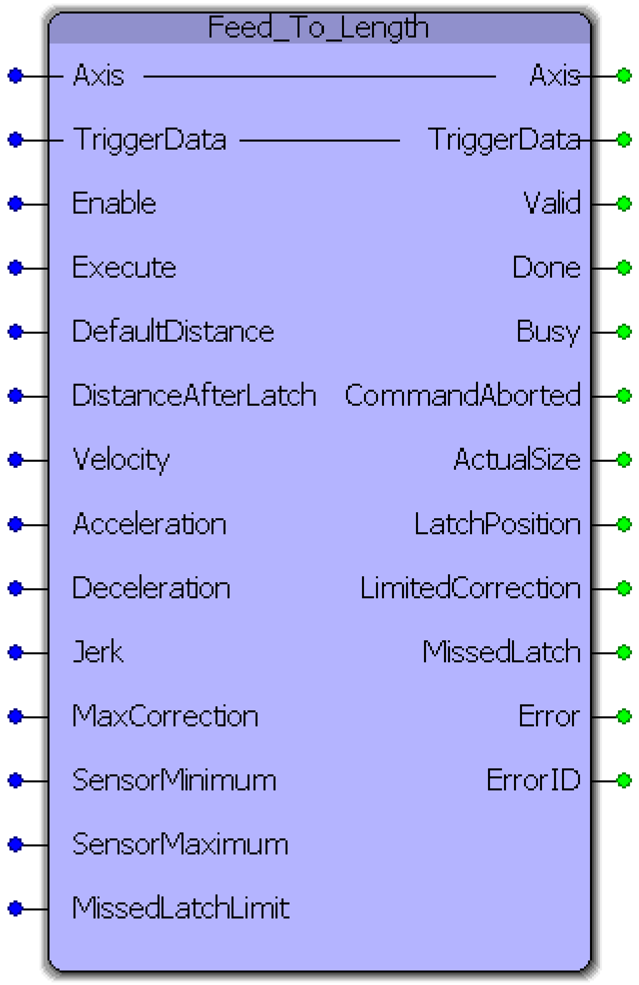
Notes

It is best to view the AlarmID Hex, because the Controller AlarmID list in the PLCopen manual displays all alarm codes in hex. This simplifies alarm identification.

Error Description

ErrorID	Meaning
0	No error
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.

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.

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 while enable is held high.	FALSE
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	FALSE
V	DefaultDistance	LREAL	The default product length. This is the distance the axis will travel if a registration mark is not detected.	LREAL#0.0
V	DistanceAfterLatch	LREAL	The desired additional travel distance after the registration mark is detected	LREAL#0.0
B	Velocity	LREAL	Absolute value of the velocity in user units/second	LREAL#0.0
B	Acceleration	LREAL	Value of the acceleration in user units/second^2 (acceleration is applicable with same sign of torque and velocity)	LREAL#0.0
B	Deceleration	LREAL	Value of the deceleration in user units/second^2 (deceleration is applicable with opposite signs of torque and velocity)	LREAL#0.0
E	*Jerk*	*LREAL*	*Not supported; reserved for future use. Value of the jerk in [user units / second^3].*	
V	MaxCorrection	LREAL	Limits the amount of correction that can be applied	LREAL#0.0
V	SensorMinimum	LREAL	The earliest slave position where a sensor position is valid for correction.	LREAL#0.0
V	SensorMaximum	LREAL	The latest slave position where a sensor position is valid for correction.	LREAL#0.0
V	MissedLatchLimit	UINT	The number of consecutive product lengths that can occur without seeing a mark in the window. Valid sensor detections will reset the internal counter. The next valid sensor detection will reset the internal counter.	UINT#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
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	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	Cumulative number of latches missed
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.
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.

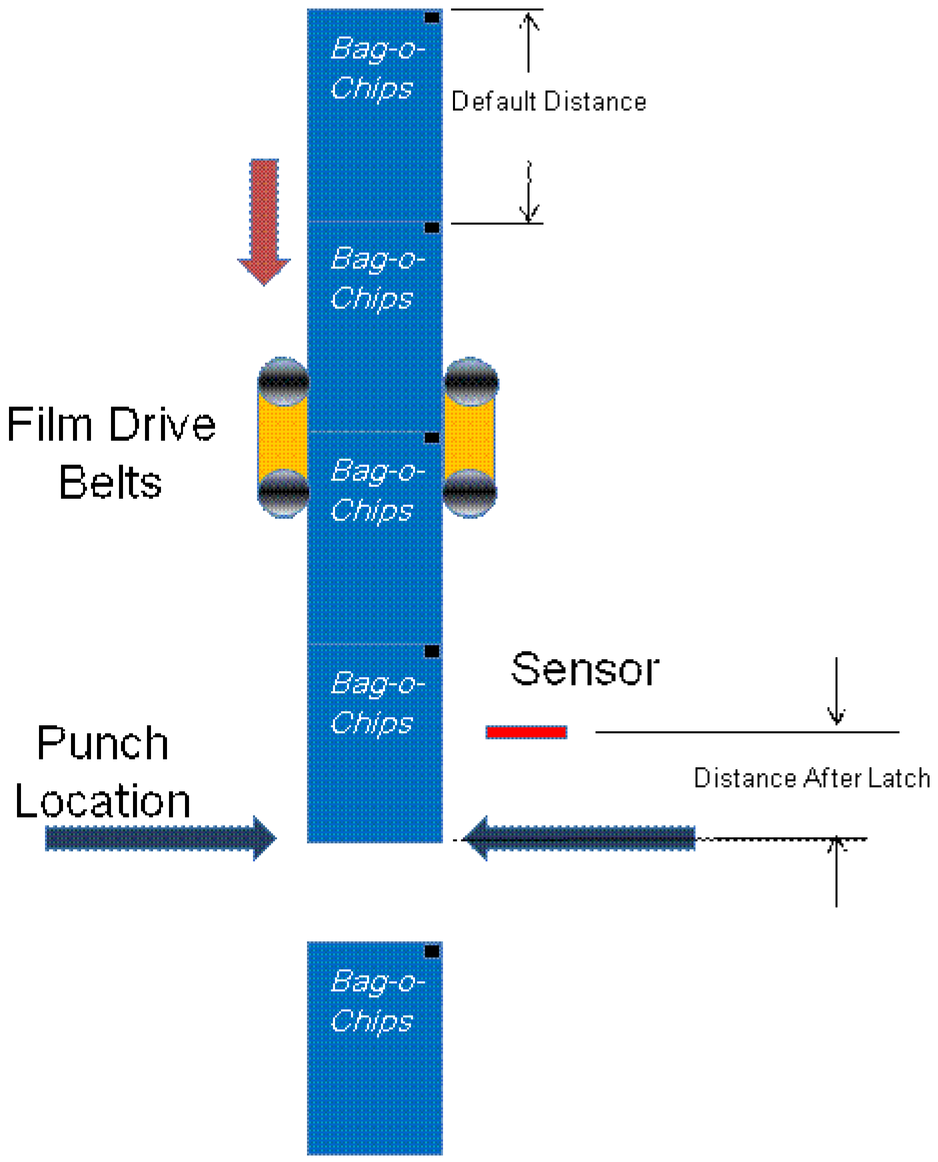
Error Description

ErrorID	Meaning
0	No error
4369	The move could not be buffered because the axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion. If MC_Stop has control of the axis, no other function block can override the "Stopping" state. Other blocks that try to cause motion while MC_Stop has control of the axis will generate this error. Also verify that the limit switches are not active by checking the Global Variables for the axis. Also, a motion block may be attempting to abort an MC_TorqueControl move.
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 can not 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 not supported on external encoders or non-Sigma V servopacks.
4624	
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.
4630	Trigger or pattern 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	Might be crossed or the same non-zero value
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 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.

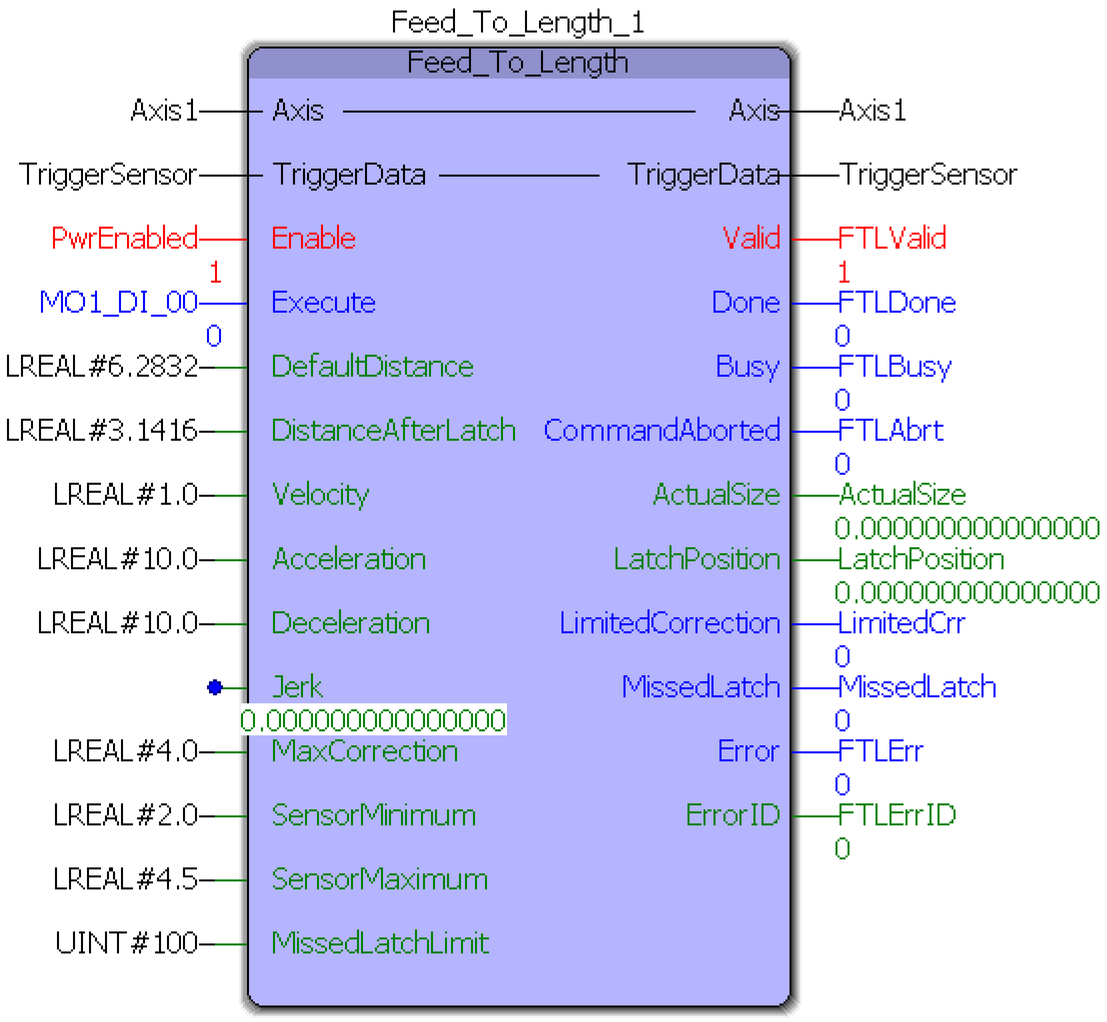
Example

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.

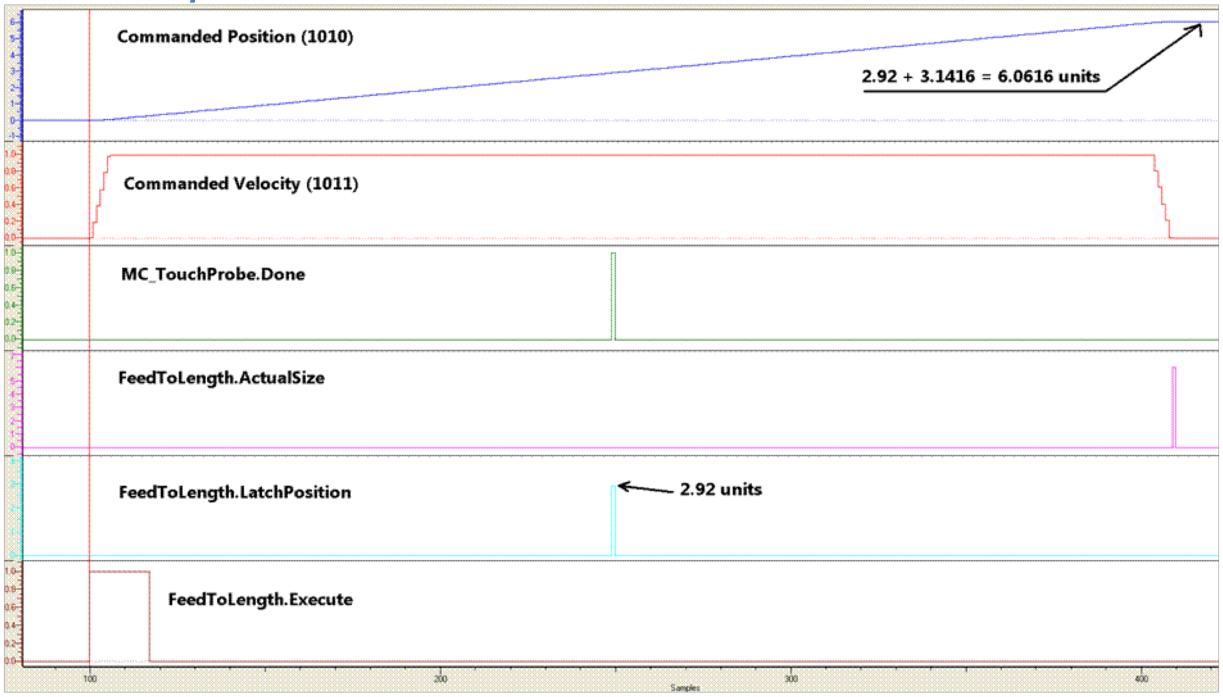


MaxCorrection limits the correction if an erroneous registration mark is captured and the calculation results in a large correction distance.

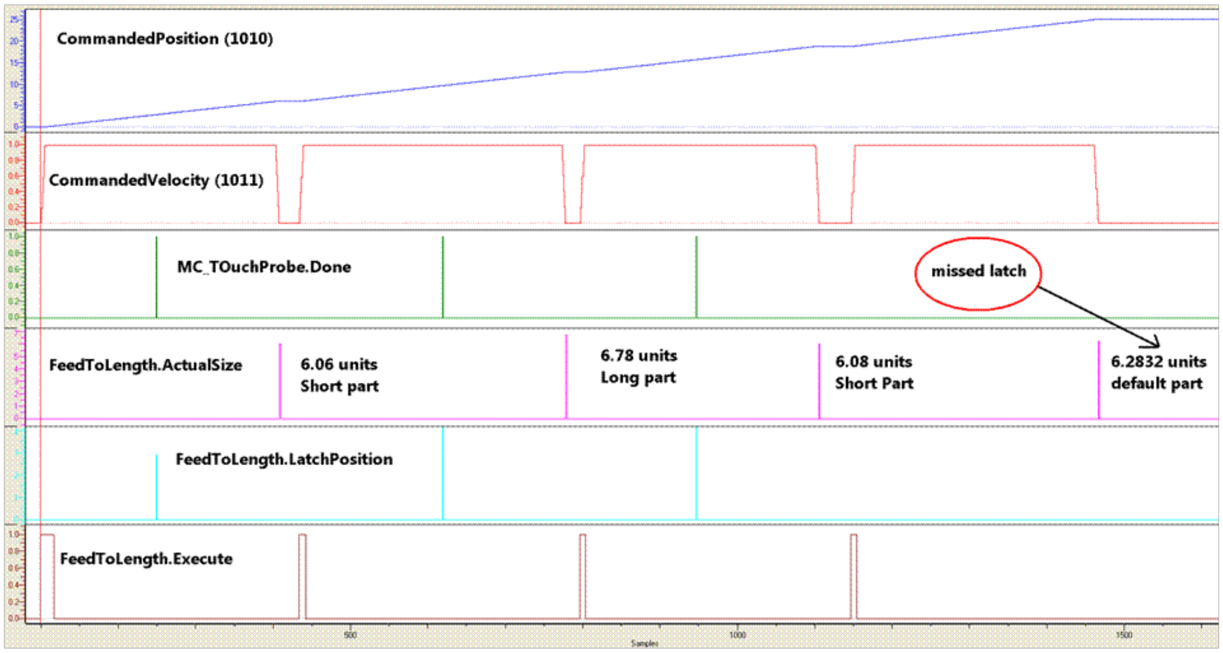
Sensor Minimum and Sensor Maximum provide window in which a registration mark must be seen to be considered a valid registration mark. In this example, the mark is expected around 3.1416 units, therefore a valid window is 2 .0 to 4.5 units. Set the window as small as appropriate for the application.



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

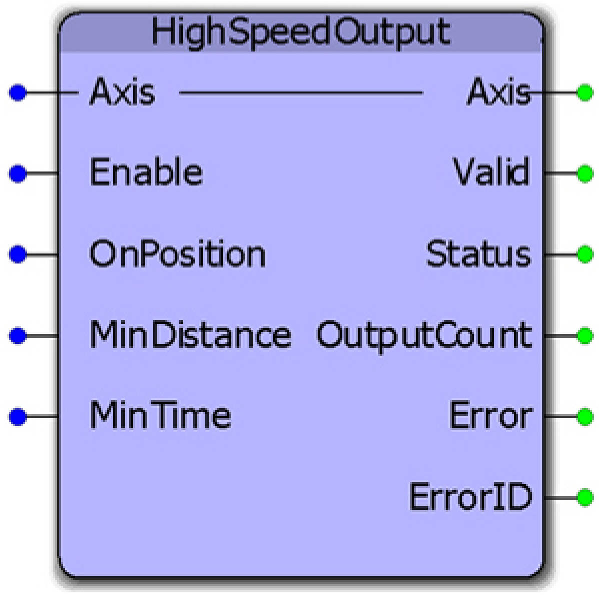


The FeedToLength function block will position the axis exactly 3.1416 units (DistanceAfterLatch) after the registration mark is detected for varying product lengths.



4

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.

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 while enable is held high.	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 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 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

High Speed Output Quick Reference

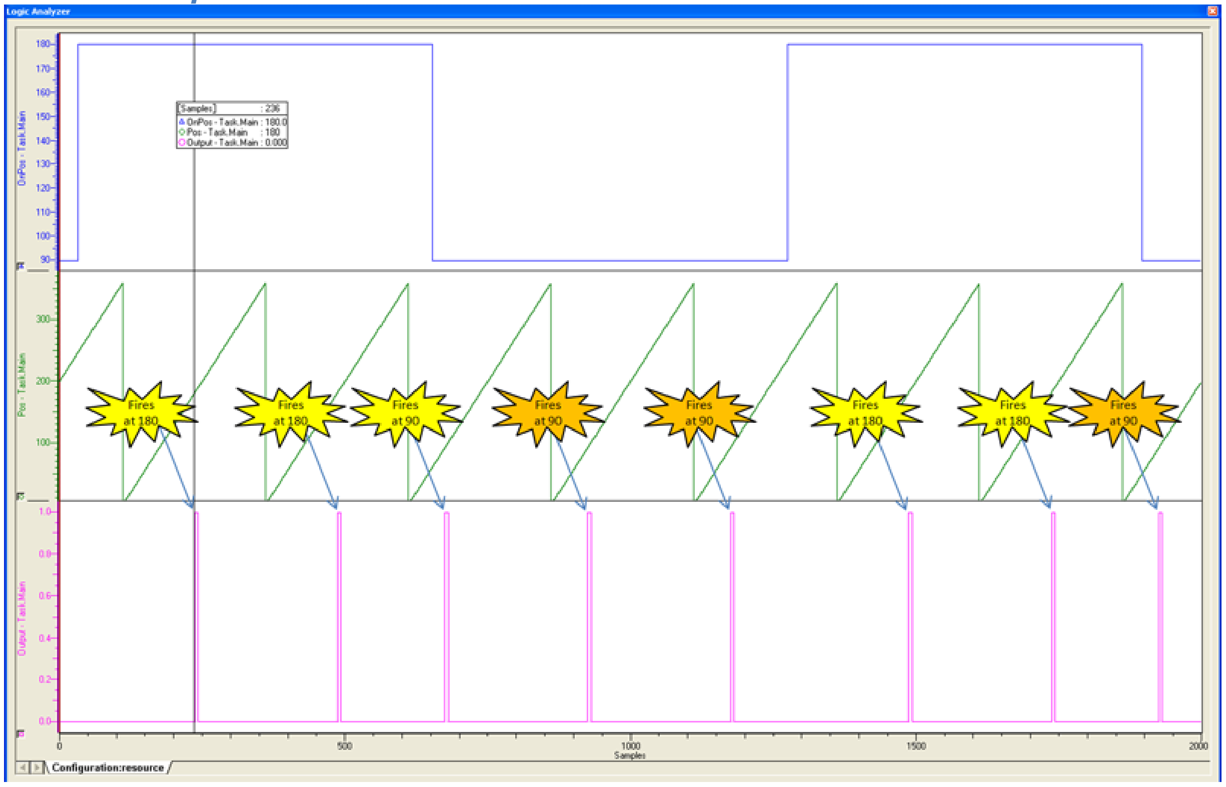
Device	Output Number	Pin Number	Software Default Name
LIO-01	DO-01	A14	M□□_DO_01
LIO-02	DO-01	A14	M□□_DO_01
LIO-06	DO-07	49	M□□_DO_07
MP2600	DO-07	44, 49	MO1_DO_01

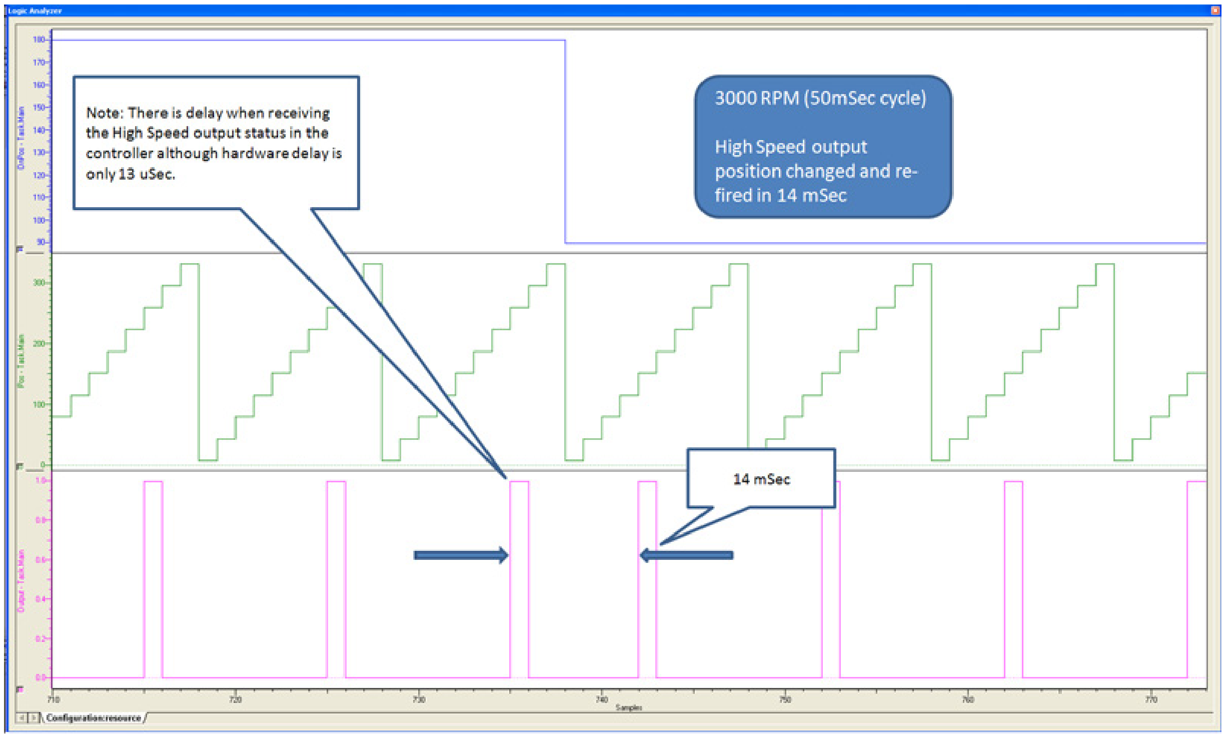
- See the HighSpeedOutput eLearning Module on Yaskawa's YouTube channel.

Error Description

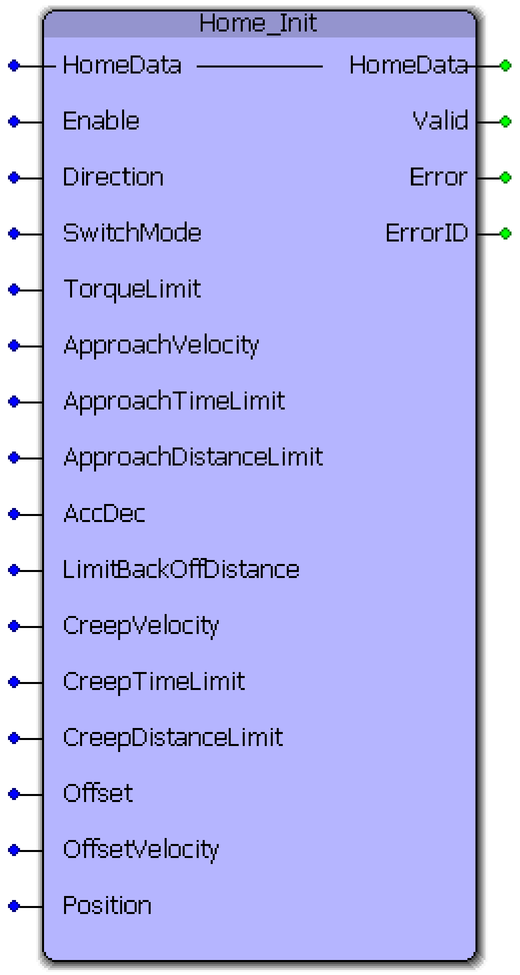
ErrorID	Meaning
0	No error
4401	Axis latch function already in use.
4402	The scan compensation delay parameter 1305 is only valid for external encoders.
4403	The High Speed Output functionality is only available on external encoders.
4625	Axis ID does not correspond to an axis configured on the system. 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.

Timing Diagram


Logic Analyzer
[Samples] : 236
OnPos - Task.Main : 180.0
Pos - Task.Main : 180
Output - Task.Main : 0.000
Fires at 180
Fires at 180
Fires at 90
Fires at 90
Fires at 90
Fires at 180
Fires at 180
Fires at 90
OnPos - Task.Main
Pos - Task.Main
Output - Task.Main
Samples
Configuration:resource


Logic Analyzer
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
Samples
Configuration:resource

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.

Parameters

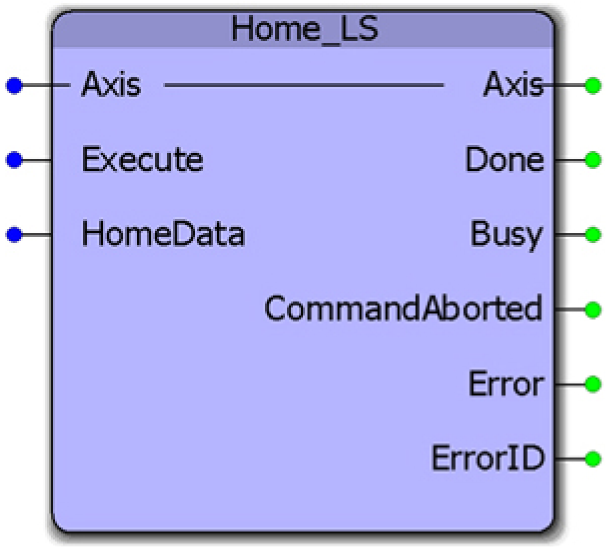
*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	HomeData	HomeStruct	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT				**Default**
B	Enable	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify	FALSE

			an input, change the value and re-trigger the execute input.	
B	Direction	MC_Direction	Direction of travel for homing	
B	SwitchMode	MC_SwitchMode	Edge On is the only mode supported	
B	TorqueLimit	LREAL	Torque limit while attempting homing. In percentage of rated torque of the servo	
B	ApproachVelocity	LREAL	Velocity used to approach limit switch or c channel	
B	ApproachTimeLimit	LREAL	Time limit for the homing attempt. In seconds	
B	ApproachDistanceLimit	LREAL	Distance limit for the homing attempt	
B	AccDec	LREAL	Acceleration/deceleration for offset moves.	
B	LimitBackOffDistance	LREAL	Distance limit for back off after a limit switch is encountered	
B	CreepVelocity	LREAL	Velocity to creep to C channel	
B	CreepTimeLimit	LREAL	Time limit for the creep attempt. In seconds	
B	CreepDistanceLimit	LREAL	Distance limit for the creep attempt	
B	Offset	LREAL	Offset distance to move after limit switch or C channel	
B	OffsetVelocity	LREAL	Velocity of the offset move after limit switch or C channel	
B	Position	HomeStruct	Position to be defined as the home position	All zeros in structure
VAR_OUTPUT				
B	Valid	BOOL	Set high if the function block is active and there are no errors	
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.	

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.

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	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' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.	
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	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 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 axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4378	The function block is not applicable for the external axis specified
4379	A homing sequence is already in progress.
4380	MC_SetPosition can not be executed while the axis is moving.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
4382	When the axis is in 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.
4390	Position cannot be defined while the axis is the cam master of other axes.
4396	Axis latch function already in use.
4397	Over travel limit still ON after attempting to move away from it.
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.
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.
61713	An internal assertion in the motion kernel failed indicating the controller is not in a stable state. Please report this error to Yaskawa America Incorporated.

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 /sec^2 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 *)

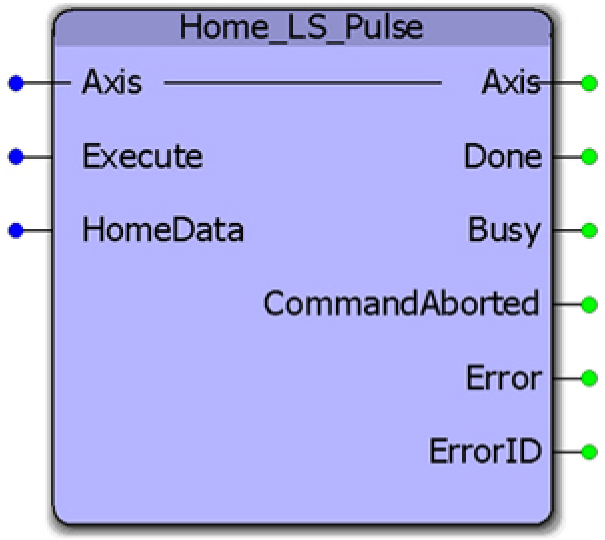
HomeStruct_ReplaceMe.OffsetVelocity:=LREAL#500.0; (* In User units / sec as set in the Hardware Configuration *)

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.

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	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 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	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 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

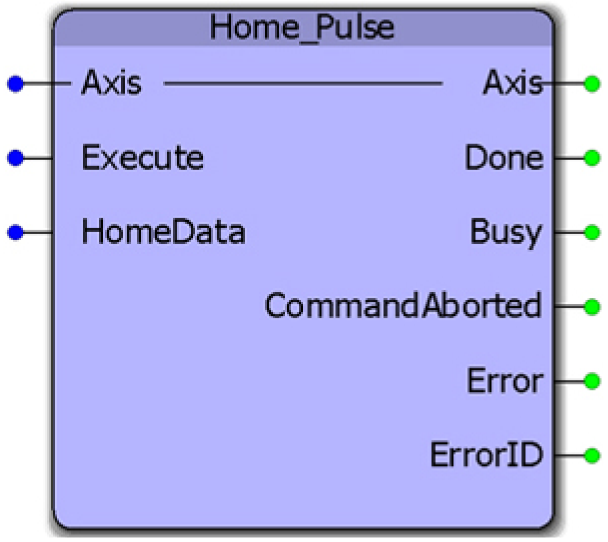
ErrorID	Meaning
0	No error
1	Time limit exceeded
2	Distance limit exceeded
3	Torque limit exceeded
4369	The move could not be buffered because the axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4378	The function block is not applicable for the external axis specified
4379	A homing sequence is already in progress.
4380	MC_SetPosition can not be executed while the axis is moving.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
4382	When the axis is in 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.
4390	Position cannot be defined while the axis is the cam master of other axes.
4396	Axis latch function already in use.
4397	Over travel limit still ON after attempting to move away from it.
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.
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.
61713	An internal assertion in the motion kernel failed indicating the controller is not in a stable state. Please report this error to Yaskawa America Incorporated.

Example

i

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.

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	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' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.	
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	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 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.

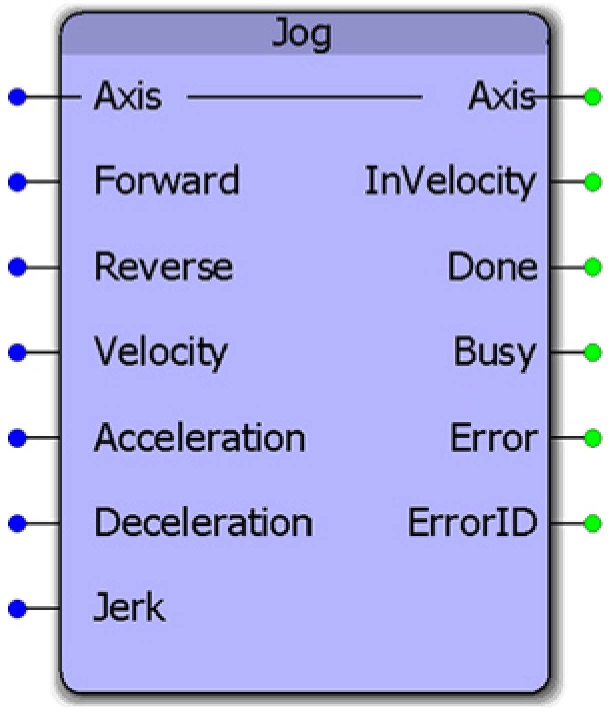
Error Description

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 axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4378	The function block is not applicable for the external axis specified
4379	A homing sequence is already in progress.
4380	MC_SetPosition can not be executed while the axis is moving.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
4382	When the axis is in 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.
4390	Position cannot be defined while the axis is the cam master of other axes.
4396	Axis latch function already in use.
4397	Over travel limit still ON after attempting to move away from it.
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.
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.
61713	An internal assertion in the motion kernel failed indicating the controller is not in a stable state. Please report this error to Yaskawa America Incorporated.

Example

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).

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	LREAL#0.0

			torque and velocity)	
B	*Jerk*	*LREAL*	*Not supported; reserved for future use. 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' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

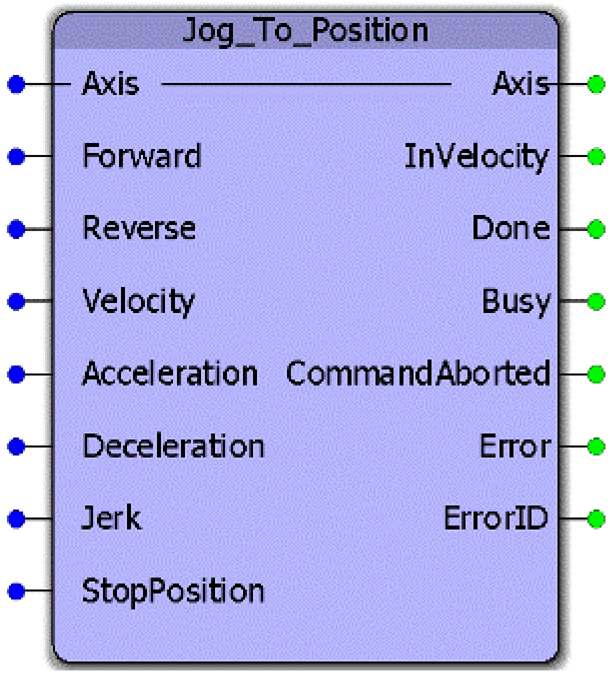
- The velocity can be changed on the fly without toggling the Forward or Reverse input. The code inside this function block will detect if the velocity has changed, and automatically re trigger the MC_MoveVelocity function block inside. Starting in PLCopen Toolbox v202, changes in Acceleration and Deceleration 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 axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

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.

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	LREAL#0.0

			with same sign of torque and velocity)	
B	Deceleration	LREAL	Value of the deceleration in user units/second^2 (deceleration is applicable with opposite signs of torque and velocity)	LREAL#0.0
E	*Jerk*	*LREAL*	*Not supported; reserved for future use. Use S-Curve parameters 1300 and 1301. Value of the jerk in [user units / second^3].*	*LREAL#0.0*
V	StopPosition	LREAL	Once the Forward and Reverse inputs are false, the axis will decelerate to a stop at the specified StopPosition using the specified deceleration rate	LREAL#0.0
VAR_OUTPUT				
B	InVelocity	BOOL	Set high when the axis first reaches the specified velocity (function is complete). 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' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.	
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 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.

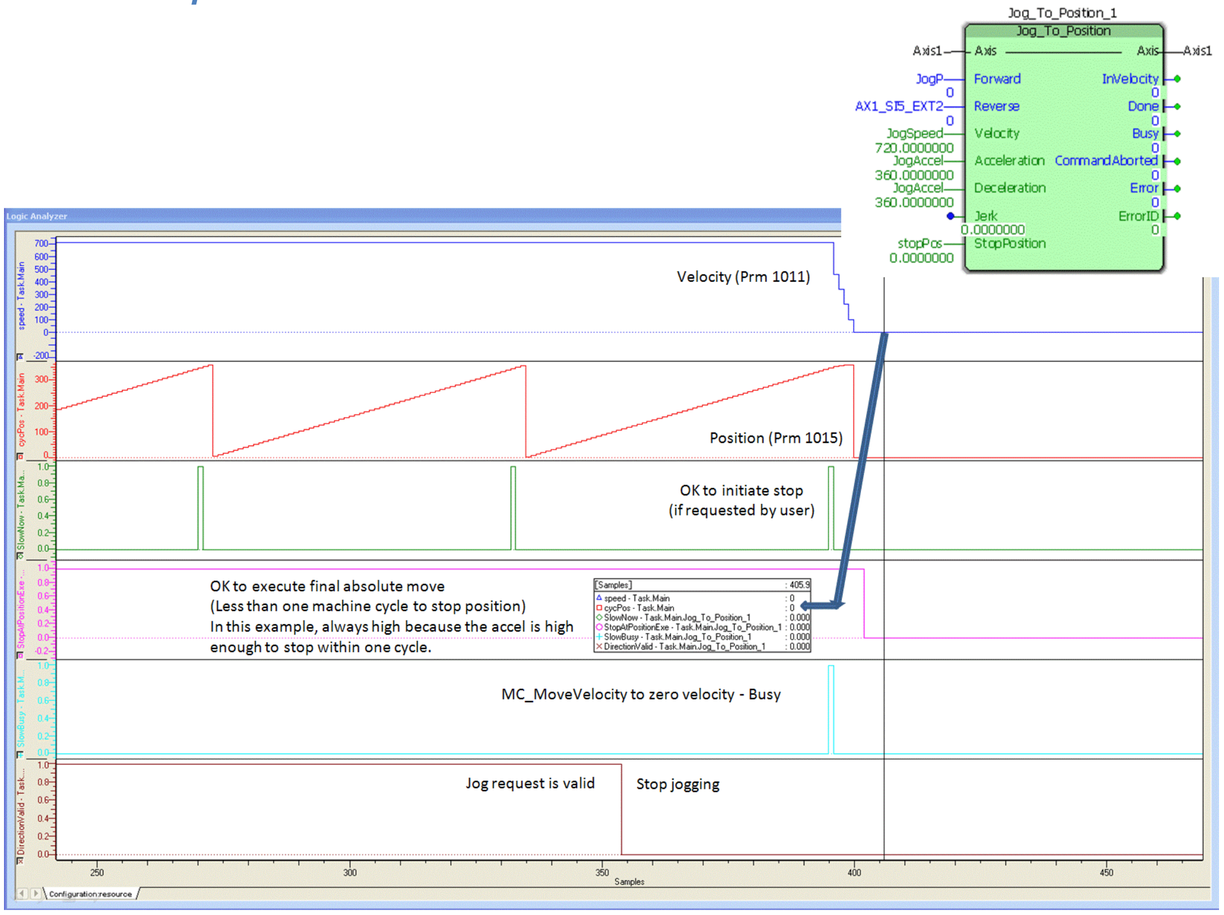
Error Description

ErrorID	Meaning
0	No error
4369	The move could not be buffered because the axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check

	MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

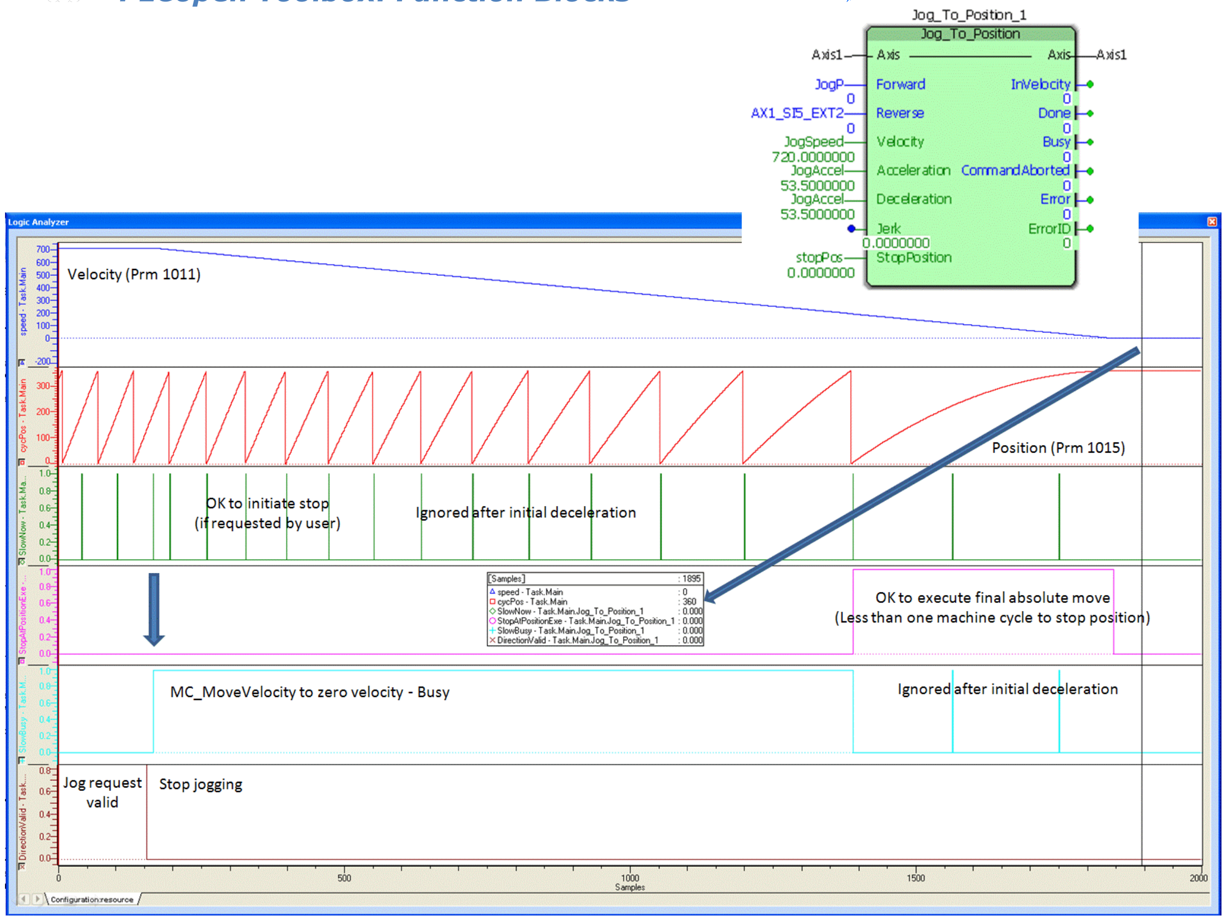
Example 1

In the first example the speed is low enough and the deceleration high enough that the axis can stop within one revolution. This is the easiest condition.



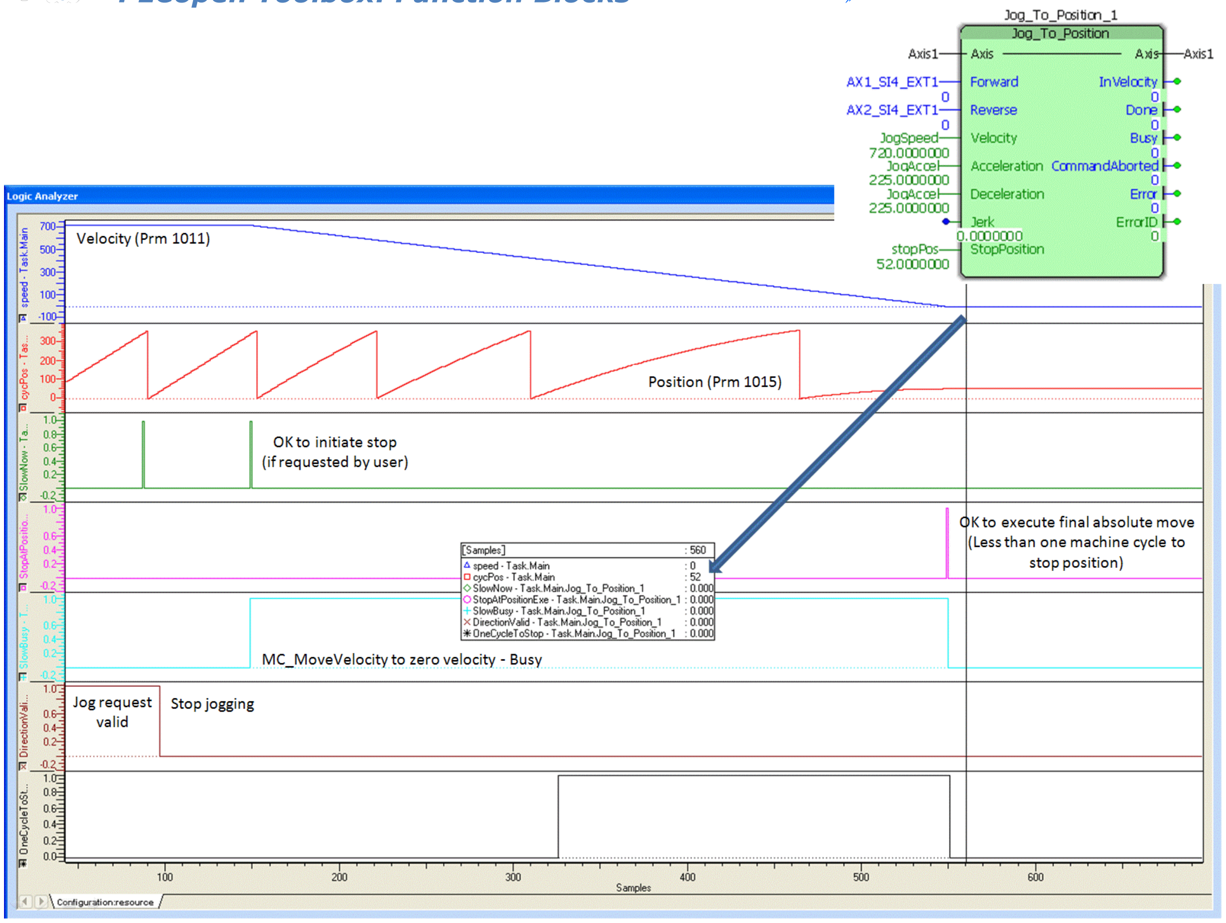
Example 2

In this example, the axis requires about 13 revolutions to come to a stop at the specified velocity and deceleration. The data "SlowNow" in green is an internal monitoring bit which results from a calculation made to determine a position that will allow the motion profile to follow the deceleration rate to the specified StopPosition. Notice there is a very brief delay between the time the Forward jog request is removed and the axis starts decelerating. This allow the axis to decelerate smoothly to the StopPositiion. The pink data indicates when the MC_MoveAbsolute is active.



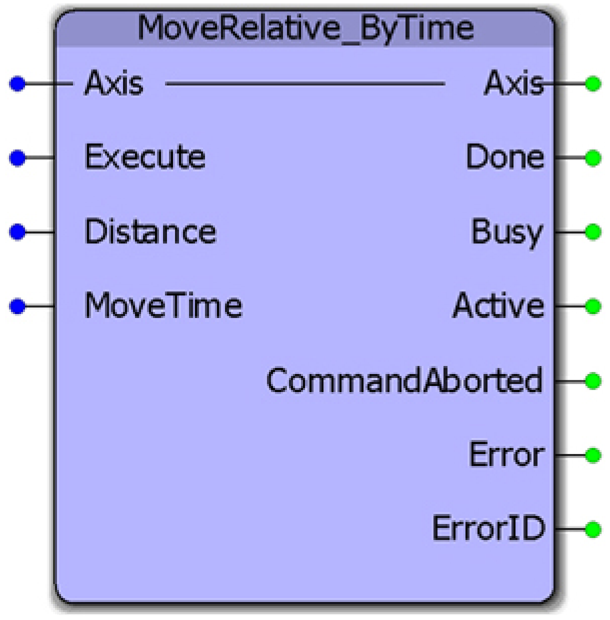
Example 3

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


Jog_To_Position_1
Jog_To_Position
Axis1 — Axis — Axis — Axis1
AX1_SI4_EXT1 — Forward — InVelocity 0
0
AX2_SI4_EXT1 — Reverse — Done 0
0
JogSpeed 720.0000000 — Velocity — Busy 0
JogAccel 225.0000000 — Acceleration — CommandAborted 0
JogAccel 225.0000000 — Deceleration — Error 0
Jerk 0.0000000 — ErrorID 0
stopPos 52.0000000 — StopPosition
Logic Analyzer
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
speed - Task.Main
cycPos - Tas...
SlowNow - Ta...
StopAtPositio...
SlowBusy - T...
DirectionVali...
OneCycleToSt...
Samples
Configuration:resource

```
[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
```

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.

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
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	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 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

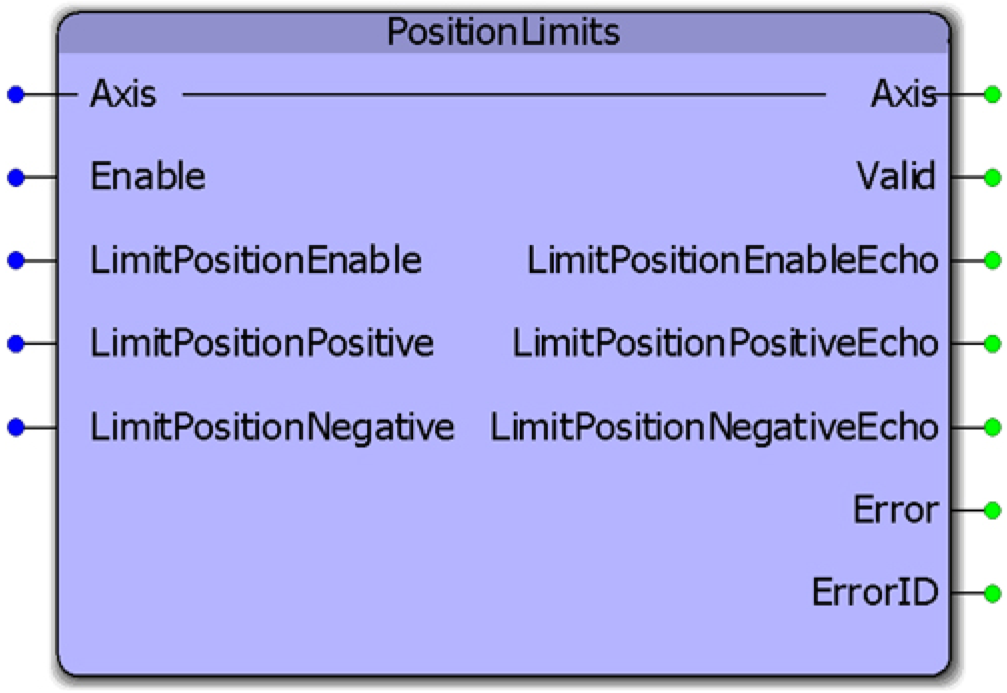
- Because this function creates a 1/3, 1/3, 1/3 trapezoidal move, it may not be appropriate for very long moves, because the calculated commanded speed may be too high.
- See the MoveRelative_ByTime eLearning Module on Yaskawa's YouTube channel.

Error Description

ErrorID	Meaning
0	No error
4369	The move could not be buffered because the axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
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 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.
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.

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.

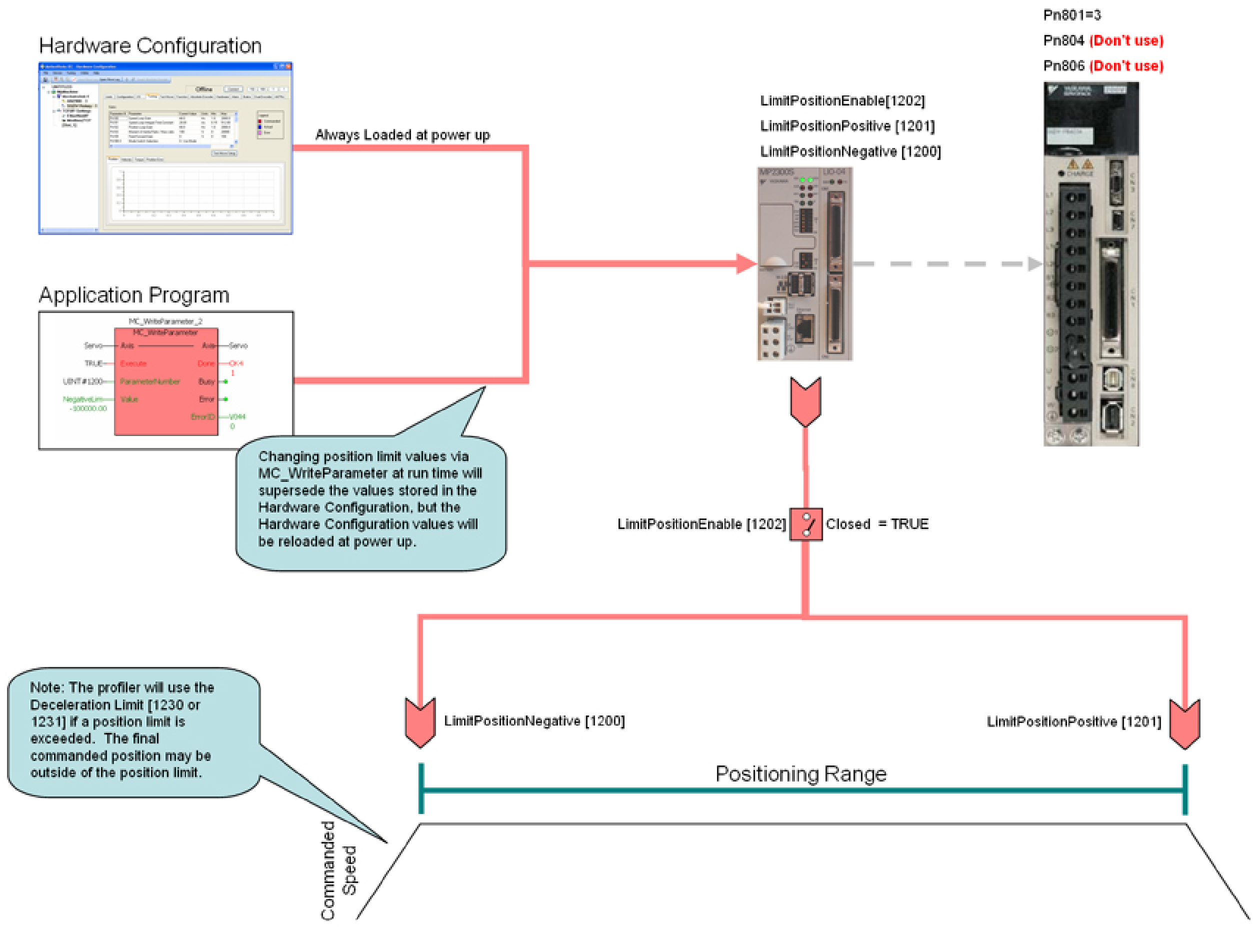
Parameters

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

Notes

The function block uses MC_ReadBoolParameter, MC_WriteBoolParameter, MC_ReadParameter, and MC_WriteParameter.


Hardware Configuration
Always Loaded at power up
Application Program
MC_WriteParameter_2
MC_WriteParameter
Servo
Axis
TRUE
Execute
Done
UINT#1200
ParameterNumber
Busy
NegativeLim
-100000.00
Value
Error
ErrorID
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.
LimitPositionEnable[1202]
LimitPositionPositive [1201]
LimitPositionNegative [1200]
Pn801=3
Pn804 (Don't use)
Pn806 (Don't use)
LimitPositionEnable [1202]
Closed = TRUE
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.
LimitPositionNegative [1200]
LimitPositionPositive [1201]
Positioning Range
Commanded Speed

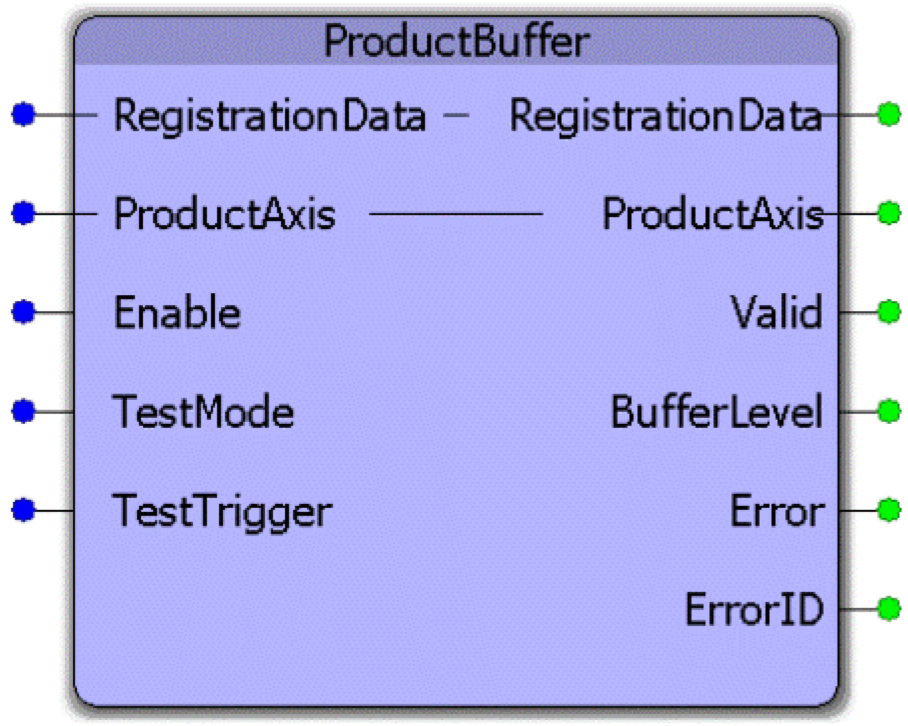
- The software position limits are managed by the MP2000iec controller. The parameters are called LimitPositionPositive and LimitPositionNegative, with values of UINT#1201 and UINT#1200 respectively. Use the MC_WriteParameter function block for these and all controller side parameters. Position limit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

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 especially for axes that transfer incoming products to a process. The accompanying "RegistrationData? structure contains information pertaining to the circular buffer and other machine dimensions related to such operations.

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 while enable is held high.	FALSE
V	TestMode	BOOL	If TRUE, then the internal MC_TouchProbe is aborted, and the function block can be used to "dry	FALSE

			cycle" the machine by simulating products using the TestTrigger input.	
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 outputs of the function are valid.	
V	BufferLevel	BOOL	Indicates the number of products in the buffer by subtracting UsePointer from StorePointer.	
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.	
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 ProductBuffer function block manages only the "storing? activity and only updates the StorePointer. Another part of your application must update the UsePointer and PrevUsePointer as the products leave the machine. If these pointers are not updated, the function block will Error with code 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, they can be inserted into the chain outside of the ProductBufferStruct.
- Both a cyclic (modularized) and unmodularized circular latch buffer are stored simultaneously.
- TestMode can be switched 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

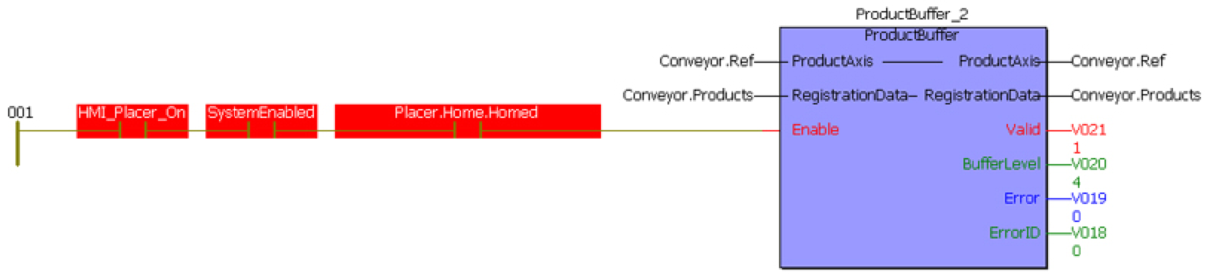
ErrorID	Meaning
0	No error
4396	Axis latch function already in use.
4624	Axis latch function already in use.
4625	Axis ID does not correspond to an axis configured on the system. Verify the value of AxisNum matches a logical axis number in the configuration. Tip: Make sure AXIS_REF is properly declared as a VAR or VAR_GLOBAL in all relevant POUs.
4630	Trigger reference is not valid
4894	The specified virtual axis may not be used with this function block.

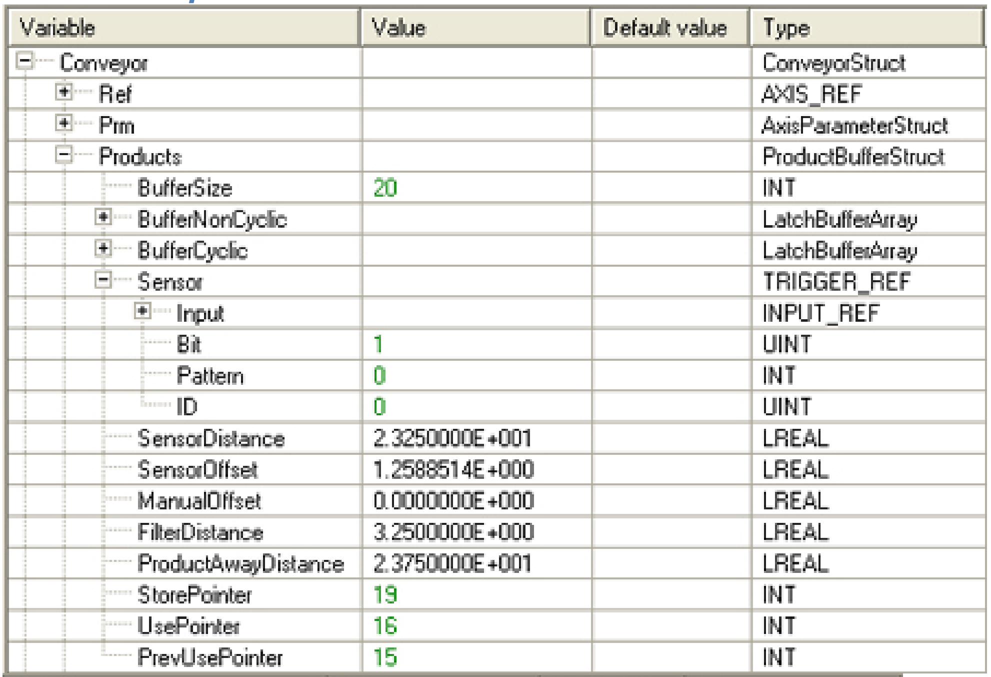
10022	Product or circular buffer overrun / full
10023	Buffer size too small / cannot be zero
57620	The DataType connected to a function block parameter specified as ANY type does not match the required data size. Right click on the function block and select "Object Properties" to determine which parameters are ANY type.

Example

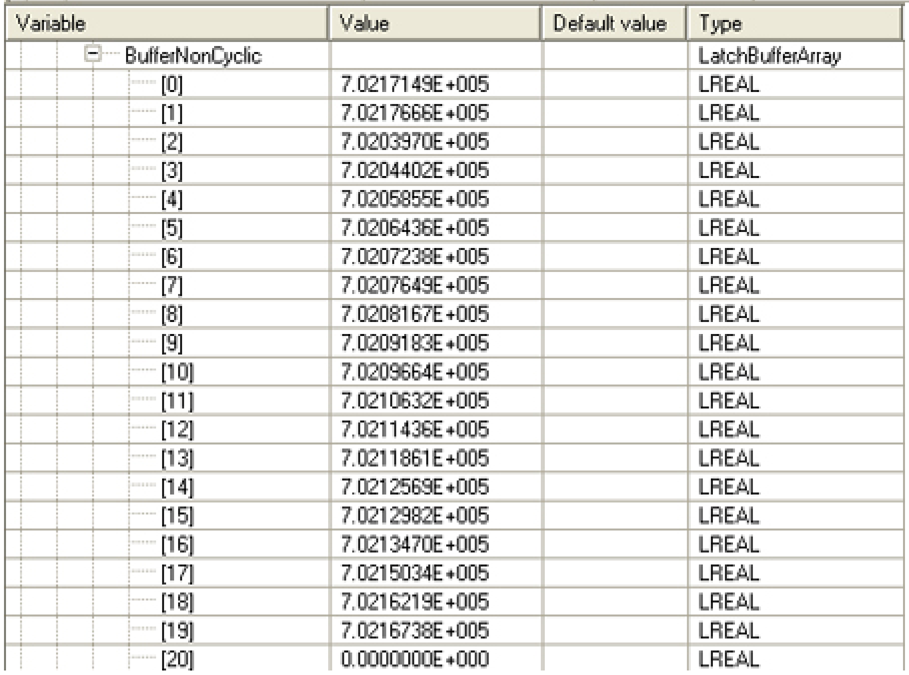
Initialization of the ProductBufferStruct in an initialize program:

```
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);
```

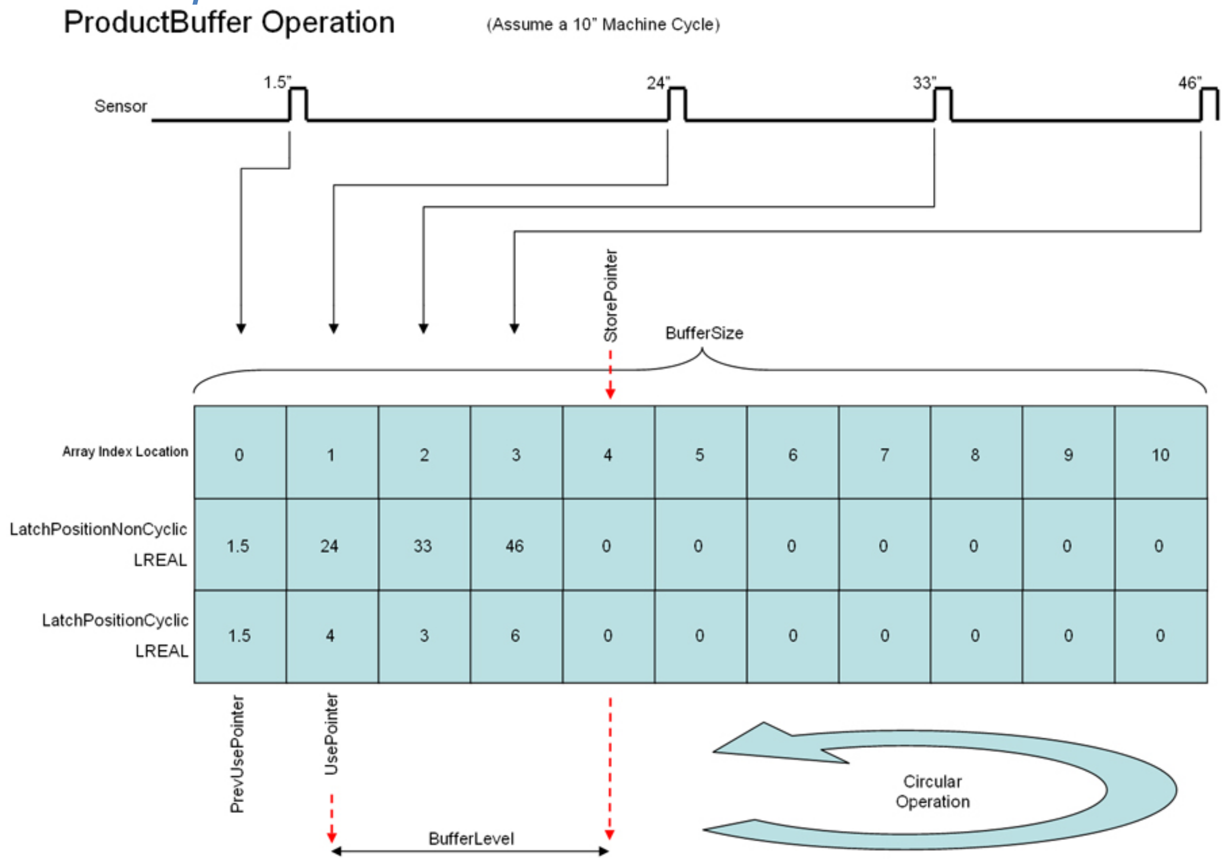




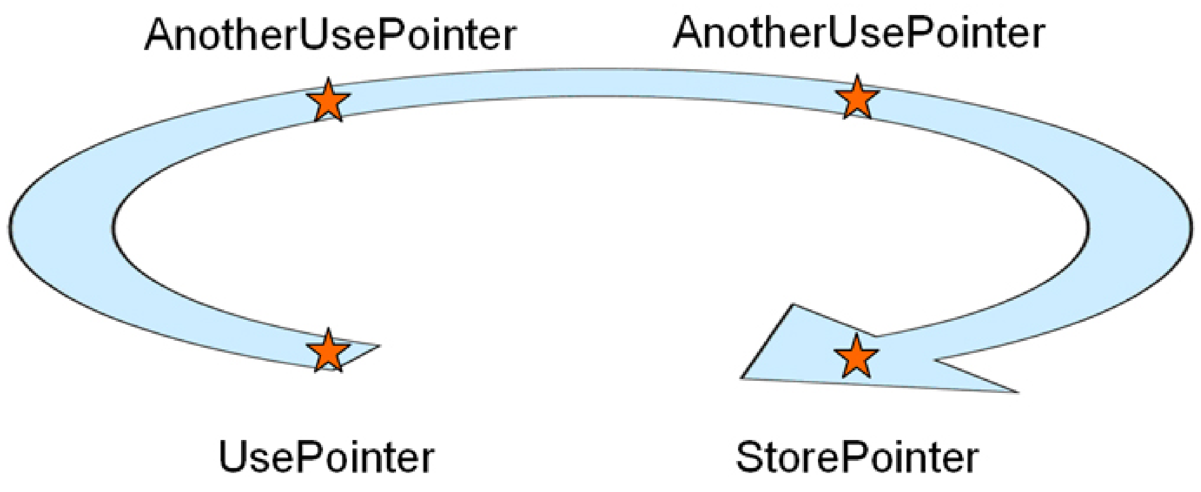
Variable	Value	Default value	Type
⊟ Conveyor			ConveyorStruct
⊞ Ref			AXIS_REF
⊞ Prm			AxisParameterStruct
⊟ Products			ProductBufferStruct
BufferSize	20		INT
⊞ BufferNonCyclic			LatchBufferArray
⊞ BufferCyclic			LatchBufferArray
⊟ Sensor			TRIGGER_REF
⊞ Input			INPUT_REF
Bit	1		UINT
Pattern	0		INT
ID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



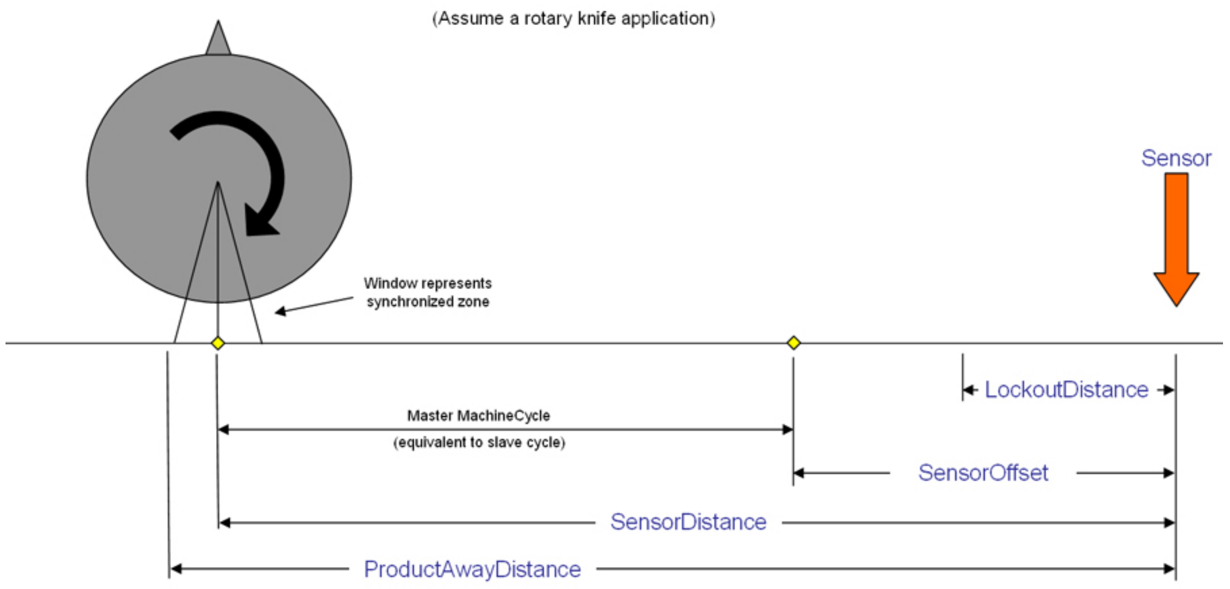
Variable	Value	Default value	Type
⊟ BufferNonCyclic			LatchBufferArray
[0]	7.0217149E+005		LREAL
[1]	7.0217666E+005		LREAL
[2]	7.0203970E+005		LREAL
[3]	7.0204402E+005		LREAL
[4]	7.0205855E+005		LREAL
[5]	7.0206436E+005		LREAL
[6]	7.0207238E+005		LREAL
[7]	7.0207649E+005		LREAL
[8]	7.0208167E+005		LREAL
[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

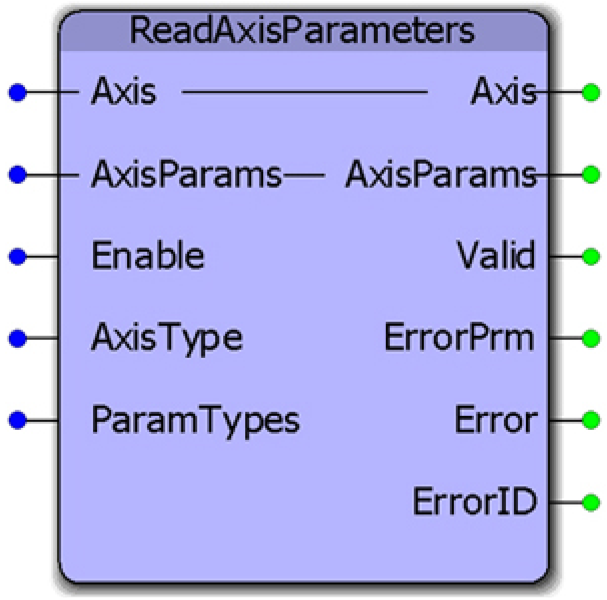


ProductBufferStruct Definitions


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

o

ReadAxisParameters



This function block reads all the commonly updated axis parameters that may be used within an application and copies them to an AxisParameterStruct. Firmware library Y_Motion should be inserted in the project that uses ReadAxisParameters.

Parameters

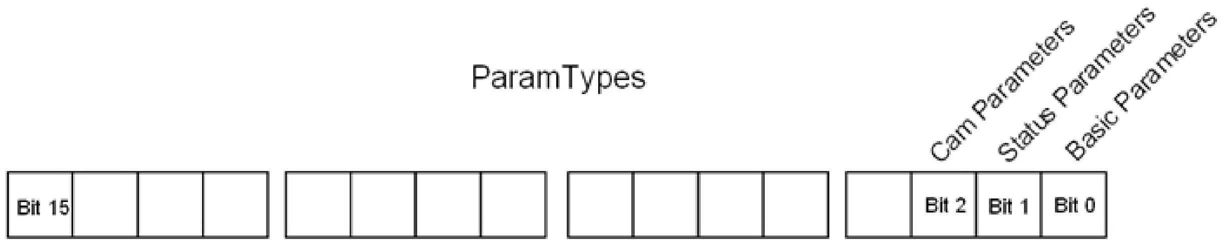
*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
V	AxisParams	AxisParameterStruct	User Defined DataType declared in the PLCopen Toolbox.	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	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	WORD#0

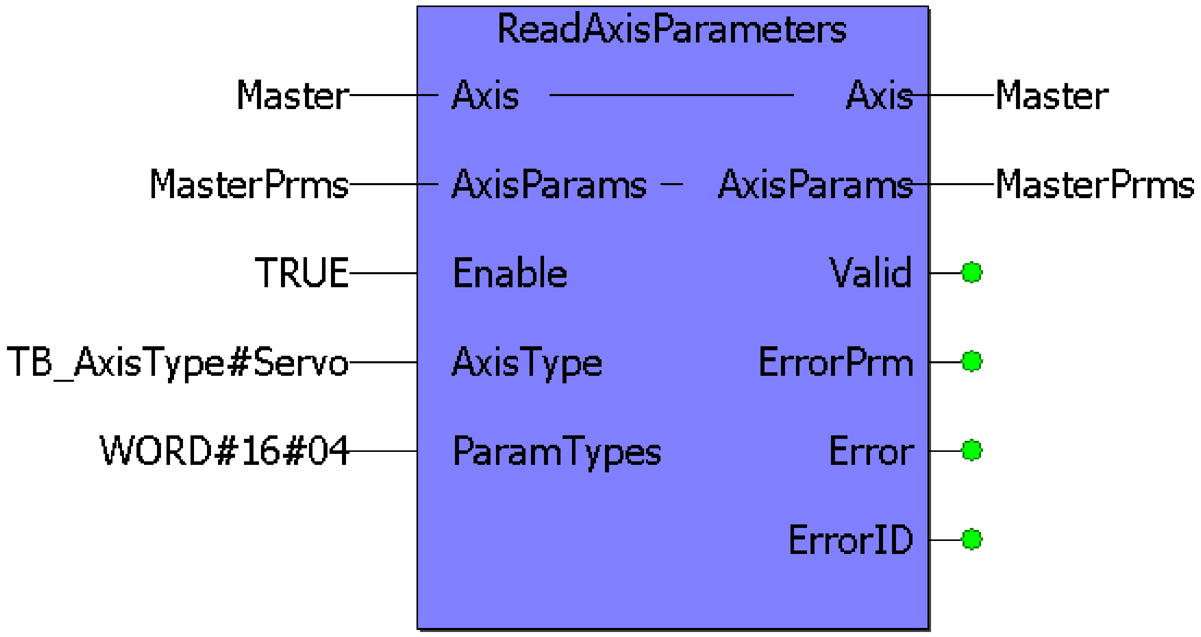
			parameter sets, such as camming.	
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
V	ErrorPrm	UINT	If there was an error while attempting to read one of the parameters listed in the ParamStruct, this output will contain the offending parameter number.	
B	Error	BOOL	Set high if 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

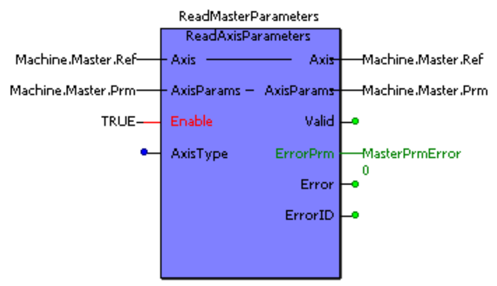
- See the ReadAxisParameters eLearning Module on Yaskawa's YouTube channel.

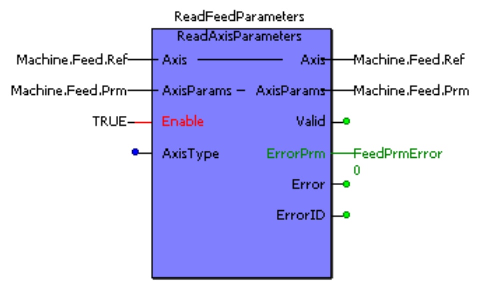
Error Description

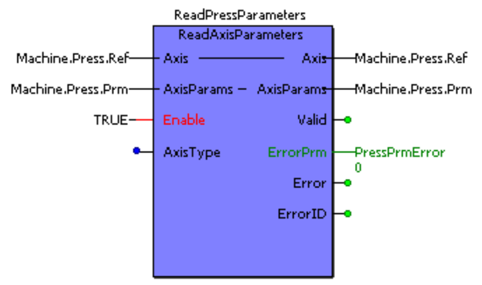
ErrorID	Meaning
0	No error
4378	The function block is not applicable for the external axis specified
4625	Axis ID does not correspond to an axis configured on the system. Verify the value of AxisNum matches a logical axis number in the configuration. Tip: Make sure AXIS_REF is properly declared as a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

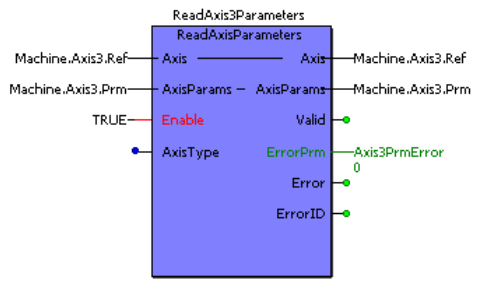
4648	The parameter number does not exist for the specified axis
57620	The DataType connected to a function block parameter specified as ANY type does not match the required data size. Right click on the function block and select "Object Properties" to determine which parameters are ANY type.

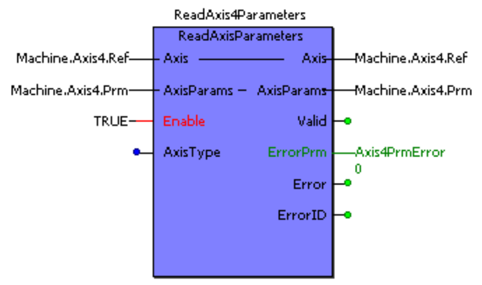
Example

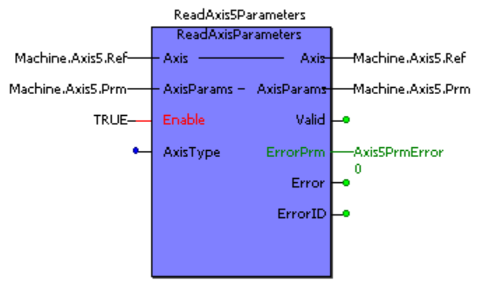


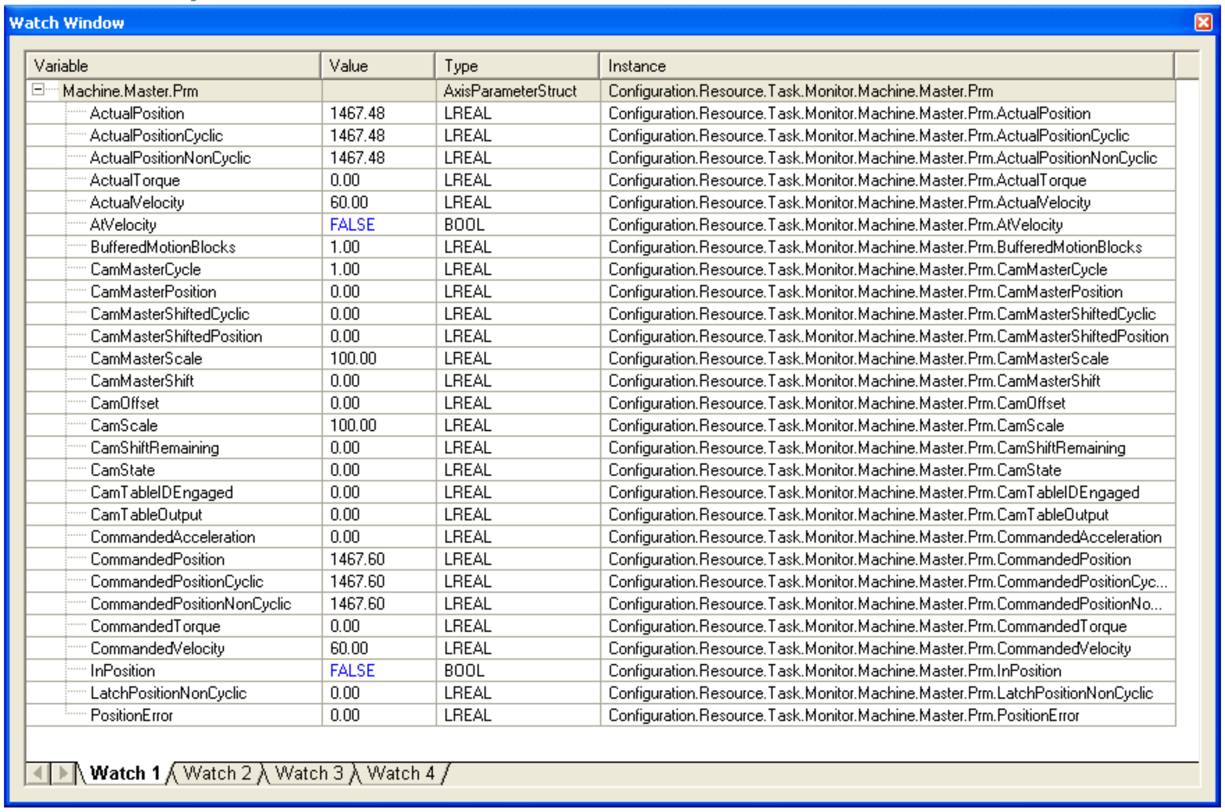








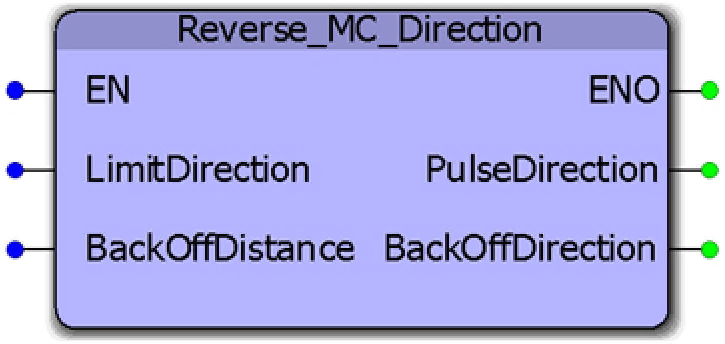




Variable	Value	Type	Instance
Machine.Master.Prm		AxisParameterStruct	Configuration.Resource.Task.Monitor.Machine.Master.Prm
ActualPosition	1467.48	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualPosition
ActualPositionCyclic	1467.48	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualPositionCyclic
ActualPositionNonCyclic	1467.48	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualPositionNonCyclic
ActualTorque	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualTorque
ActualVelocity	60.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.ActualVelocity
AtVelocity	FALSE	BOOL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.AtVelocity
BufferedMotionBlocks	1.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.BufferedMotionBlocks
CamMasterCycle	1.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterCycle
CamMasterPosition	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterPosition
CamMasterShiftedCyclic	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterShiftedCyclic
CamMasterShiftedPosition	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterShiftedPosition
CamMasterScale	100.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterScale
CamMasterShift	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamMasterShift
CamOffset	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamOffset
CamScale	100.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamScale
CamShiftRemaining	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamShiftRemaining
CamState	0.00	LREAL	Configuration.Resource.Task.Monitor.Machine.Master.Prm.CamState
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

r

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.

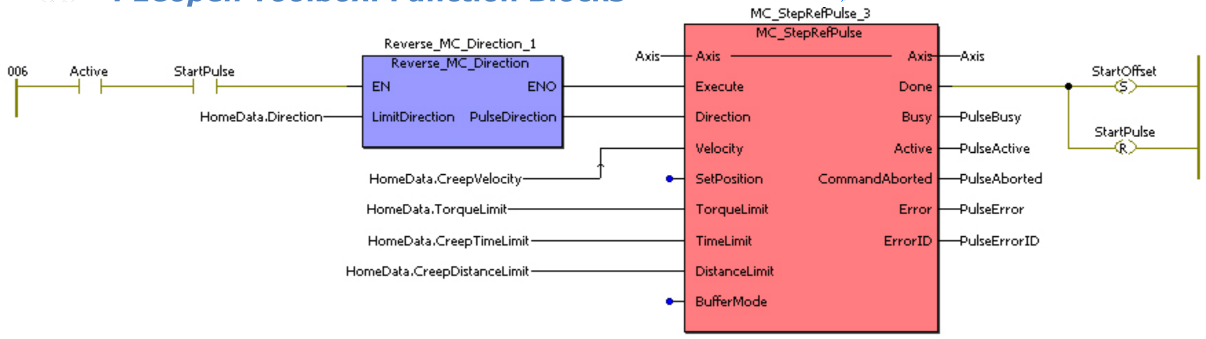
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	EN	BOOL	Enables the function.	FALSE
V	LimitDirection	INT	ENum	
V	BackOffDistance	LREAL		INT#0
VAR_OUTPUT				
B	ENO	BOOL	High if the function executed normally	
V	PulseDirection	INT	MC_Direction#positive_direction or MC_Direction#negative_direction	
V	BackOffDirection	LREAL		

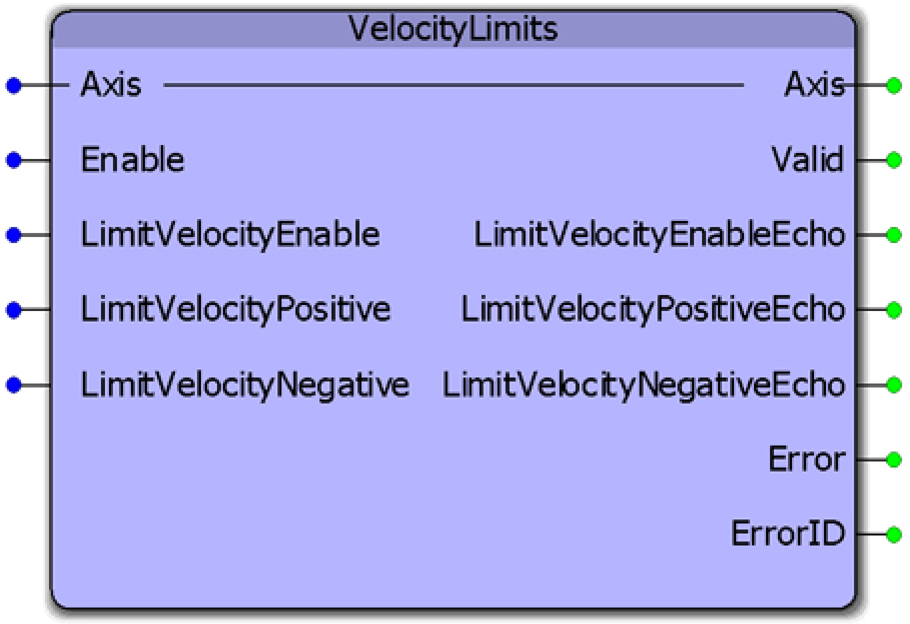
Error Description

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
LimitDirection
PulseDirection
HomeData.Direction
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.

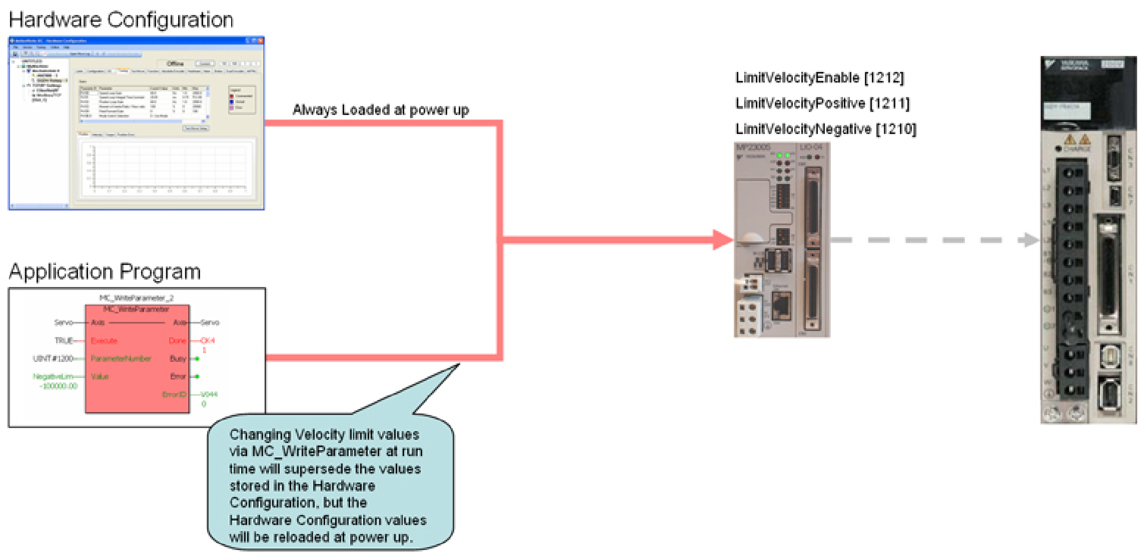
Parameters

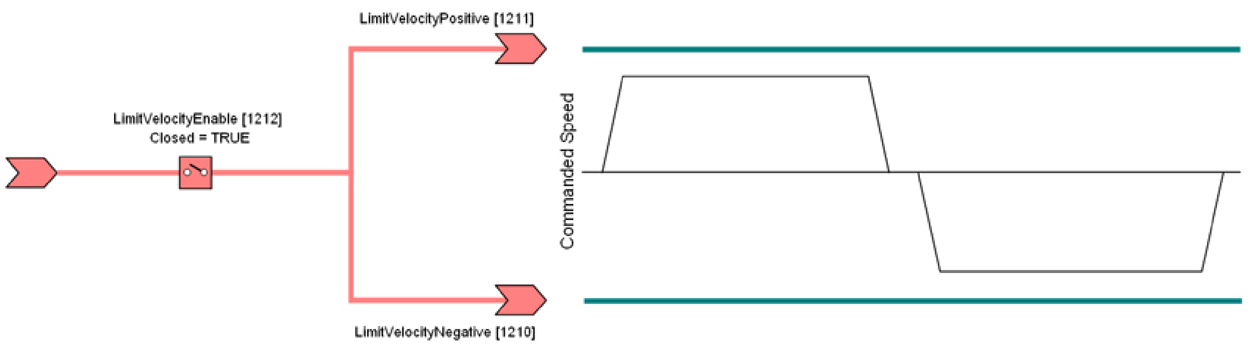
*	Parameter	Data Type	Description	
VAR_IN_OUT				
B	Axis	AXIS_REF	Logical axis reference. This value can be located on the Configuration tab in the Hardware Configuration (logical axis number).	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
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 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 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
Always Loaded at power up
LimitVelocityEnable [1212]
LimitVelocityPositive [1211]
LimitVelocityNegative [1210]
MC_WriteParameter
Servo
TRUE
UINT#1200
Axis
Execute
ParameterNumber
Value
Done
Busy
Error
ErrorID
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.


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.

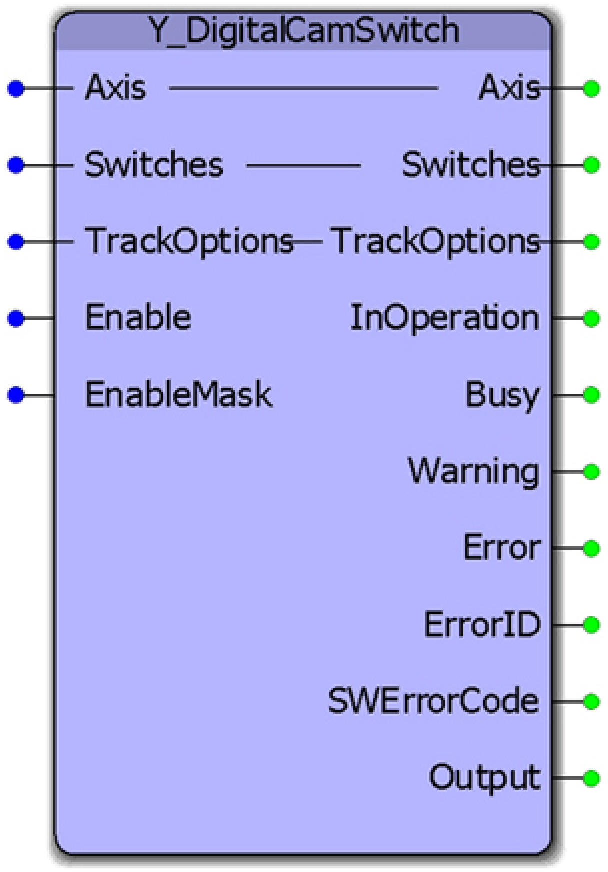
• LimitVelocityPositive must be zero or greater.

• 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.

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.	
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.	FALSE
VAR_OUTPUT				
B	InOperation	BOOL	Function Block Enable is ON and at least 1 track is enabled (EnableMask is <> 0).	
E	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 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

Data Validation

The data passed into the function block will be validated at run time.

All TrackNumbers must be in the range of 1 ~ 32 (corresponds to bit locations in EnableMask).

AxisDirection must be 0, any other number will default to 0. (values 1 and 2 not supported at this time)

CamSwitchMode must be 0 or 1, any other number will default to 0.

The total number of switches must be less than or equal to 256.

Error Description

ErrorID	Meaning
0	No error
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.
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

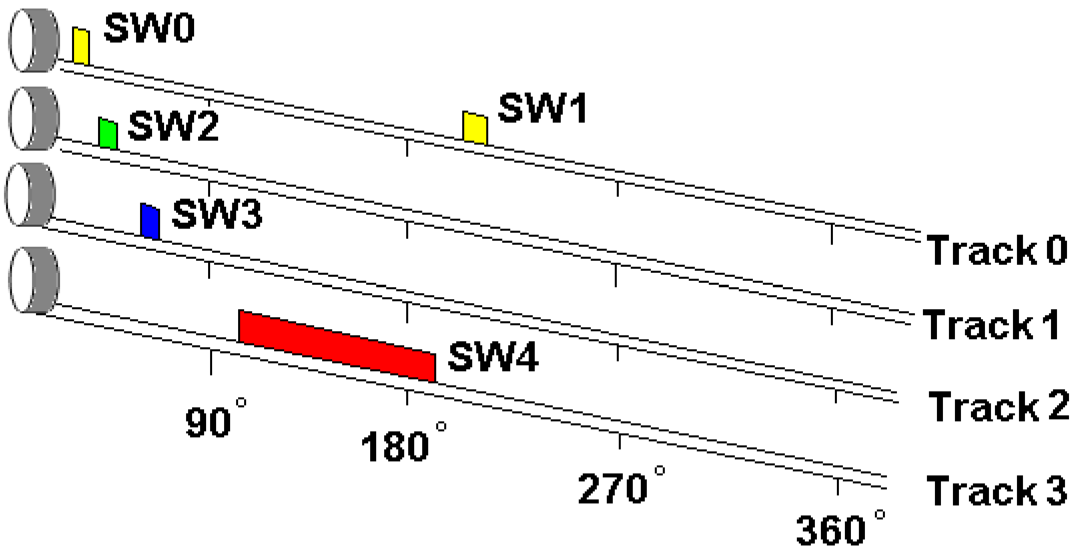
Switch 3: On Position : 50 degrees

Off Position : 60 degrees

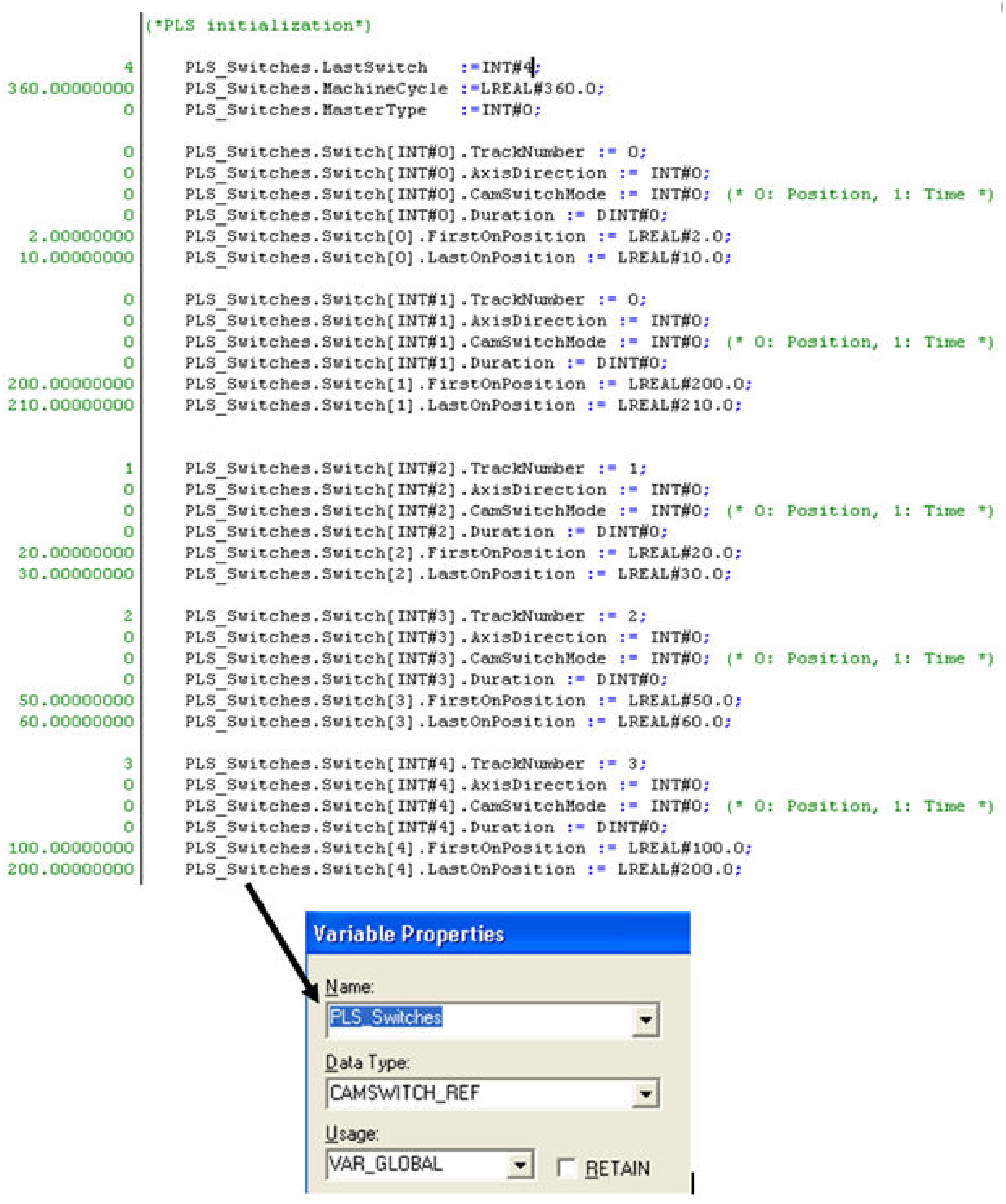
Track 3 has 1 switch associated with it

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;
```

Variable Properties

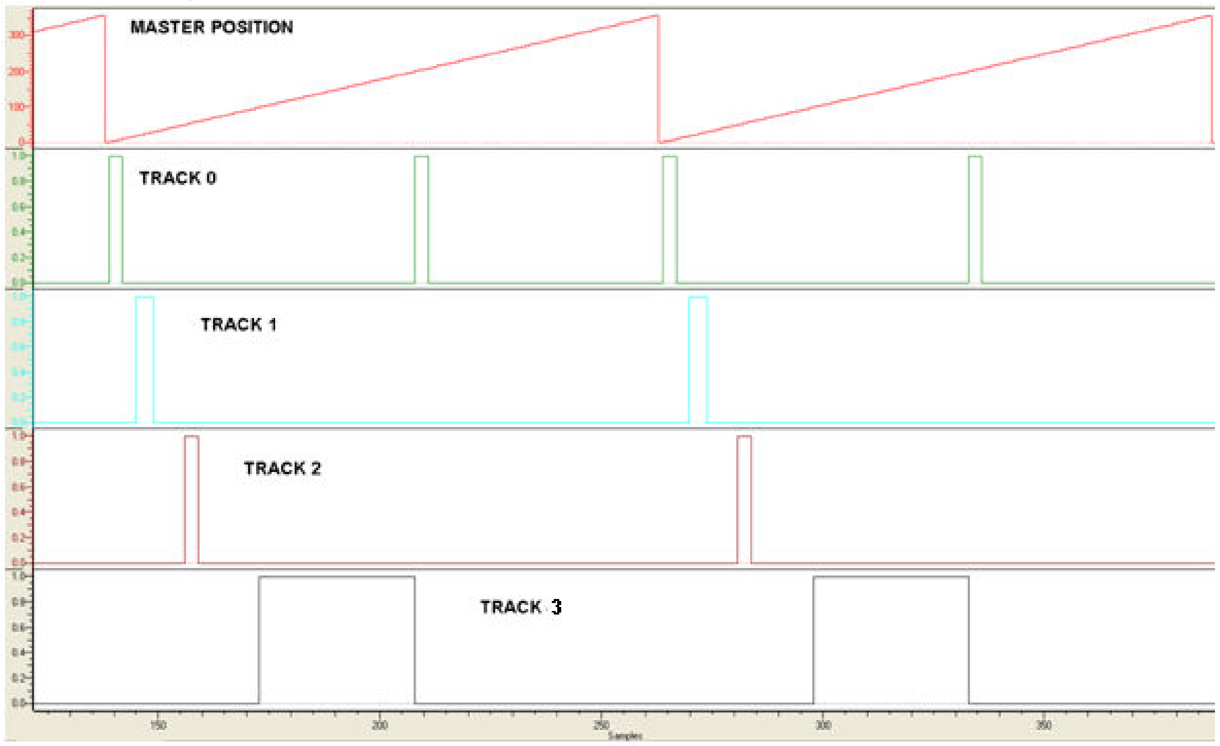
Name: PLS_Switches

Data Type: CAMSWITCH_REF

Usage: VAR_GLOBAL

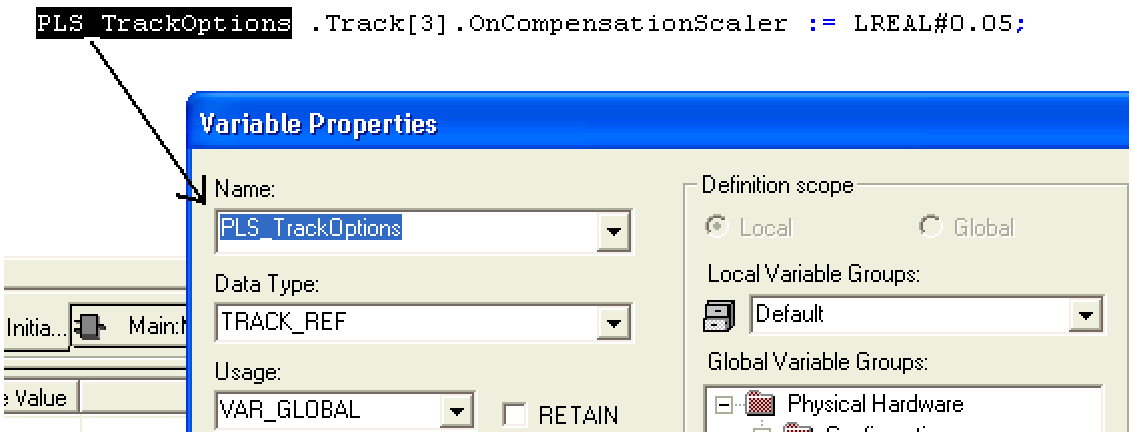
☐ RETAIN

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.



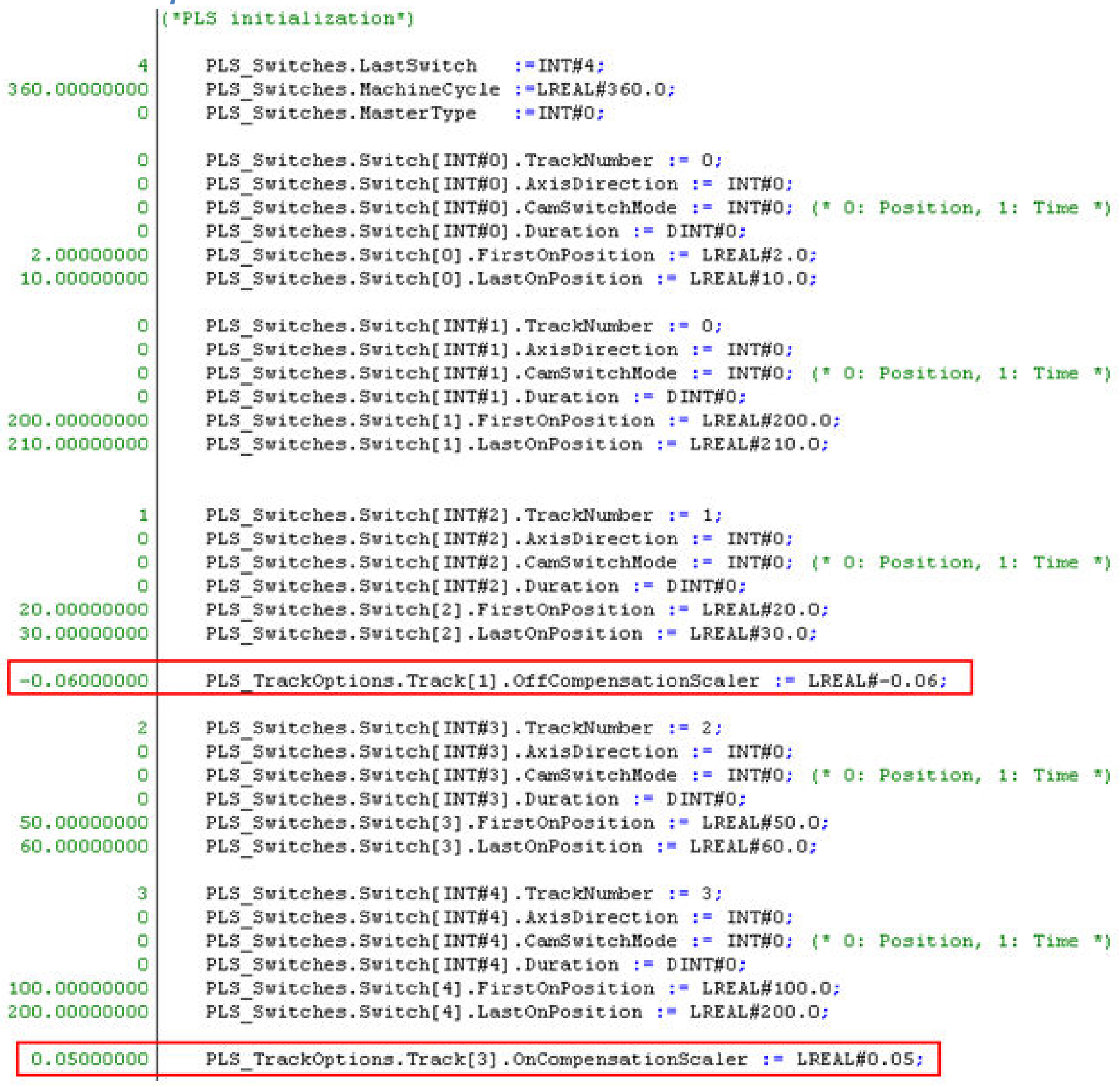
```
PLS_TrackOptions .Track[3].OnCompensationScaler := LREAL#0.05;
```

Variable Properties

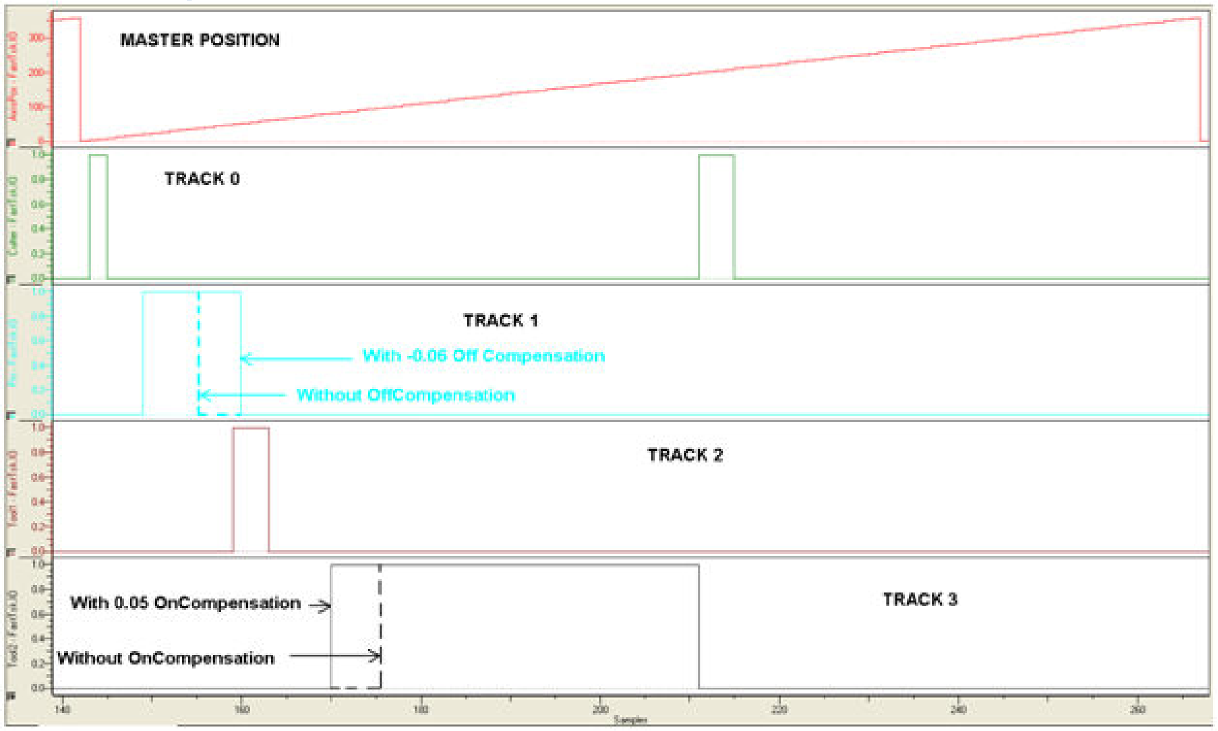
Name: PLS_TrackOptions

Data Type: TRACK_REF

Usage: VAR_GLOBAL



```
             (*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
TRACK 0
TRACK 1
With -0.06 Off Compensation
Without OffCompensation
TRACK 2
TRACK 3
With 0.05 OnCompensation
Without OnCompensation
Samples

Yaskawa Toolbox

Yaskawa Toolbox

The Yaskawa Toolbox consists of the following:

Data Types:

Enumerated Type	Description
MovingAverageArray	For use with the MovingAverage function block.
PIDStruct	Used with the PIDControl function block.
RTCStruct	Used with the RealTimeClock, DateCompare, and the Y_SetRTC function blocks.
XYArray	Supporting structure for XYDataStruct. For use with the XYLookup function block
XYData	Supporting structure for XYArray. For use with the XYLookup function block
XYDataStruct	For use with the XYLookup function block

Function Blocks:

Function Block	Description
Action	
Blink	Toggles the Output at the frequency specified at the input.
CommWatchDog	Allows the application program to monitor data being transmitted from a master device.
DataRecord	Records data into the array.
DataSort	Sorts data from the lowest to highest value of X data.
DataCompare	Calculates the difference between two real time clock values and provides the difference as a real time clock value.
Enable_FB_Template	Template which can be used when developing functions which adhere to the PLCopen output behavior.
Enable_ST_Template	
Execute_FB_Template	
Execute_ST_Template	
MovingAverage	
PackByte	
PackWord	
PIDControl	Can be used as a generic control loop feedback mechanism.
RangeCheck	
RealTimeClock	

Scaler	
UnpackByte	
UnpackWord	
WindowCheck	
XYLookup	

Getting Started: Yaskawa

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:

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

Previous Versions:

(*****************   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: v010beta 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 outout to PID function.

(***********************************   2009-04-03: v006 released
    ***********************************)

Added CommHeartbeat Funtion

(***********************************   2009-03-27: v005 released
    ***********************************)

Added MovingAverage Funtion

(***********************************   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 varaibles 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 varaibles match the FB version.

Action:

Dummy FB to show simulation of the template function blocks.

Explicit Data Types

ExplicitSendDataStruct

For use with the Explicit_Message function block

Data Type Declaration

```
ExplicitSendDataStruct : STRUCT (*Refer to 2-5.7.2 in Vol 2, Chapter 2 EtherNet/IP Adaptation of CIP*)

ED_Command1   : BYTE;  (*ED:Encapsulation Data*)

ED_Command2   : BYTE;

ED_Length1 : BYTE;

ED_Length2 : BYTE;

ED_SessionHandle1 : BYTE;

ED_SessionHandle2 : BYTE;

ED_SessionHandle3 : BYTE;

ED_SessionHandle4 : BYTE;

ED_Status1 : BYTE;

ED_Status2 : BYTE;

ED_Status3 : BYTE;

ED_Status4 : BYTE;

ED_SenderContext : SenderContext;

ED_Options1 : BYTE;

ED_Options2 : BYTE;

ED_Options3 : BYTE;

ED_Options4 : BYTE;

ED_InterfaceHandle1 : BYTE;

ED_InterfaceHandle2 : BYTE;

ED_InterfaceHandle3 : BYTE;
```

ED_InterfaceHandle4 : BYTE;

ED_TimeOut1 : BYTE;

ED_TimeOut2 : BYTE;

ED_ItemCount1 : BYTE;

ED_ItemCount2 : BYTE;

ED_AddressItemID1 : BYTE;

ED_AddressItemID2 : BYTE;

ED_AddressItemLength1 : BYTE;

ED_AddressItemLength2 : BYTE;

ED_DataItemID1 : BYTE;

ED_DataItemID2 : BYTE;

ED_DataItemLength1 : BYTE;

ED_DataItemLength2 : BYTE;

ED_DataService : Service;

ED_Data : ExplicitData;

```
END_STRUCT;
```

ExplicitReceiveDataStruct

For use with the Explicit_Message function block

Data Type Declaration

```
ExplicitReceiveDataStruct : STRUCT (*Refer to 2-5.7.2 in Vol 2, Chapter 2 EtherNet/IP Adaptation of CIP*)

ED_Command1   : BYTE;  (*ED:Encapsulation Data*)

ED_Command2   : BYTE;

ED_Length1 : BYTE;

ED_Length2 : BYTE;

ED_SessionHandle1 : BYTE;

ED_SessionHandle2 : BYTE;

ED_SessionHandle3 : BYTE;

ED_SessionHandle4 : BYTE;

ED_Status1 : BYTE;

ED_Status2 : BYTE;

ED_Status3 : BYTE;

ED_Status4 : BYTE;

ED_SenderContext : SenderContext;

ED_Options1 : BYTE;

ED_Options2 : BYTE;

ED_Options3 : BYTE;

ED_Options4 : BYTE;

ED_InterfaceHandle1 : BYTE;

ED_InterfaceHandle2 : BYTE;

ED_InterfaceHandle3 : BYTE;

ED_InterfaceHandle4 : BYTE;
```

```
ED_TimeOut1 : BYTE;

ED_TimeOut2 : BYTE;

ED_ItemCount1 : BYTE;

ED_ItemCount2 : BYTE;

ED_AddressItemID1 : BYTE;

ED_AddressItemID2 : BYTE;

ED_AddressItemLength1 : BYTE;

ED_AddressItemLength2 : BYTE;

ED_DataItemID1 : BYTE;

ED_DataItemID2 : BYTE;

ED_DataItemLength1 : BYTE;

ED_DataItemLength2 : BYTE;

ED_Response1 : BYTE;

ED_Response2 : BYTE;

ED_ResponseStatus1 : BYTE;

ED_ResponseStatus2 : BYTE;

ED_Data : ExplicitData;

END_STRUCT;
```

RegSessionRequestStruct

For use with the Explicit_Message function block.

Data Type Declaration

RegSessionRequestStruct : STRUCT (*Refer to 2-5.4.2 in Vol 2, Chapter 2 EtherNet/IP Adaptation of CIP*)

RSR_Command1   : BYTE; (*RSR: Register Session Request*)

RSR_Command2 : BYTE;

RSR_Length1 : BYTE;

RSR_Length2 : BYTE;

RSR_SessionHandle1 : BYTE;

RSR_SessionHandle2 : BYTE;

RSR_SessionHandle3 : BYTE;

RSR_SessionHandle4 : BYTE;

RSR_Status1 : BYTE;

RSR_Status2 : BYTE;

RSR_Status3 : BYTE;

RSR_Status4 : BYTE;

RSR_SenderContext : SenderContext;

RSR_Options1 : BYTE;

RSR_Options2 : BYTE;

RSR_Options3 : BYTE;

RSR_Options4 : BYTE;

RSR_ProtocolVersion1 : BYTE;

RSR_ProtocolVersion2 : BYTE;

RSR_OptionFlags1 : BYTE;

RSR_OptionFlags2 : BYTE;

```
END_STRUCT;
```

UnRegSessionRequestStruct

For use with the Explicit_Message function block.

Data Type Declaration

```
UnRegSessionRequestStruct  : STRUCT (*Refer to 2-5.4.3 in Vol 2, Chapter 2 EtherNet/IP Adaptation of CIP*)

USR_Command1   : BYTE;

USR_Command2   : BYTE;  (*USR: Unregister Session Request *)

USR_Length1  : BYTE;

USR_Length2  : BYTE;

USR_SessionHandle1 : BYTE;

USR_SessionHandle2 : BYTE;

USR_SessionHandle3 : BYTE;

USR_SessionHandle4 : BYTE;

USR_Status1 : BYTE;

USR_Status2 : BYTE;

USR_Status3 : BYTE;

USR_Status4 : BYTE;

USR_SenderContext : SenderContext;

USR_Options1 : BYTE;

USR_Options2 : BYTE;

USR_Options3 : BYTE;

USR_Options4 : BYTE;

END_STRUCT;
```

SenderContext

For use with the Explicit_Message function block.

Data Type Declaration

```
TYPE

SenderContext : ARRAY[0..7] OF BYTE;

END_TYPE
```

Service

For use with the Explicit_Message function block.

Data Type Declaration

TYPE

Service : ARRAY[0..7] OF BYTE;

END_TYPE

ExplicitData

For use with the Explicit_Message function block.

Data Type Declaration

```
TYPE

ExplicitData : ARRAY[0..503] OF BYTE;

END_TYPE
```

Data Types

Data Types for Yaskawa Toolbox

The following is a complete list of all DataTypes included in the Yaskawa Toolbox. The list is arranged to separate those that are used internally, and not useful outside of their particular function, and those that an application program must incorporate when the programmer wishes to use the associated Function Block.

Data Type	Usage
DataTypes for external use with Yaskawa Toolbox function blocks	
MovingAverageArray	For use with the MovingAverage function block.
PIDStruct	Used with the PIDControl function block.
RTCStruct	Used with the RealTimeClock, DateCompare, and the Y_SetRTC function blocks.
XYDataStruct	For use with the XYLookup function block
DataTypes that support other DataTypes (no need for direct use by the application programmer)	
XYArray	Supporting structure for XYDataStruct. For use with the XYLookup function block
XYData	Supporting structure for XYArray. For use with the XYLookup function block

Data Type: MovingAverageArray

For use with the MovingAverage function block.

Data Type Declaration

```
TYPE

MovingAverageArray: ARRAY[0..30000] OF LREAL;  (*   Adjust the array size if more data elements are desired.
 *)

END_TYPE
```

Data Type: PIDStruct

Used with the PIDControl function block.

Data Type Declaration

```
TYPE

PIDStruct: STRUCT

Ts:LREAL;          (*   Sample time  *)

Kp:LREAL;          (*   Proportional Gain   *)

Ki:LREAL;          (*   Integral Gain   *)

Kd:LREAL;          (*   Derivative Gain   *)

Ti:LREAL;          (*   Integral Time (in Sec.)   *)

Td1:LREAL;         (*   Derivative Time for Divergent Inputs   *)

Td2:LREAL;         (*   Derivative Time for Convergent Inputs   *)

ILL:LREAL;         (*   Integral Lower Limit  *)

IUL:LREAL;         (*   Integral Upper Limit  *)

LowerLimit:LREAL; (*   Lower Limit for ControlOutput   *)

UpperLimit:LREAL; (*   Upper Limit for ControlOutput  *)

DeadBand:LREAL;   (*   Dead band limit  *)

END_STRUCT;

END_TYPE
```

Data Type: RTCStruct

Used with the RealTimeClock, DateCompare, and the Y_SetRTC function blocks.

Data Type Declaration

```
TYPE

RTC_Struct: STRUCT

Year:INT;

Month:INT;

Day:INT;

Hour:INT;

Minute:INT;

Second:INT;

mSec:INT;

END_STRUCT;

END_TYPE
```

Data Type: XYArray

Supporting structure for XYDataStruct. For use with the XYLookup function block.

Data Type Declaration

```
TYPE

XYArray: ARRAY[0..4000] OF XYData;  (*   NOTE!  Had strange error message after

                                download when set to 5000   *)

END_TYPE
```

Data Type: XYData

Supporting structure for XYArray. For use with the XYLookup function block

Data Type Declaration

```
TYPE

XYData: STRUCT

X:LREAL;  (*   Any data that will be used with the XY lookup function

               as input   *)

Y:LREAL;  (*   Any data that will be used with the XY lookup function

               as output   *)

END_STRUCT;

END_TYPE
```

Data Type: XYDataStruct

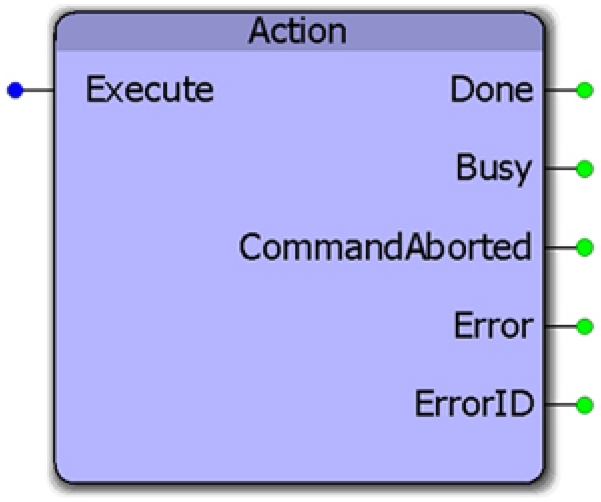
For use with the XYLookup function block

Data Type Declaration

```
TYPE

XYDataStruct: STRUCT

Pair: XYArray;    (*   Adjust the XYArray size if more data elements are desired.  *)

LastPair:INT;     (*   Set this value to indicate the last ACTUAL array element

                       that contains user data   *)

END_STRUCT;

END_TYPE;
```

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.

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 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	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 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.

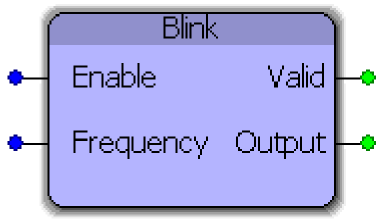
Error Description

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.

Parameters

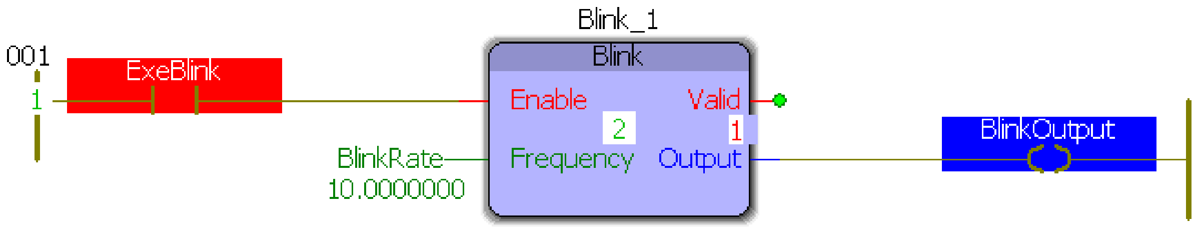
Parameter		Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
V	Frequency	LREAL	The cycle rate in Hertz.	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates if the function is operating	
V	Output	BOOL	Toggled at the specified frequency when the function is enabled.	

Error Description

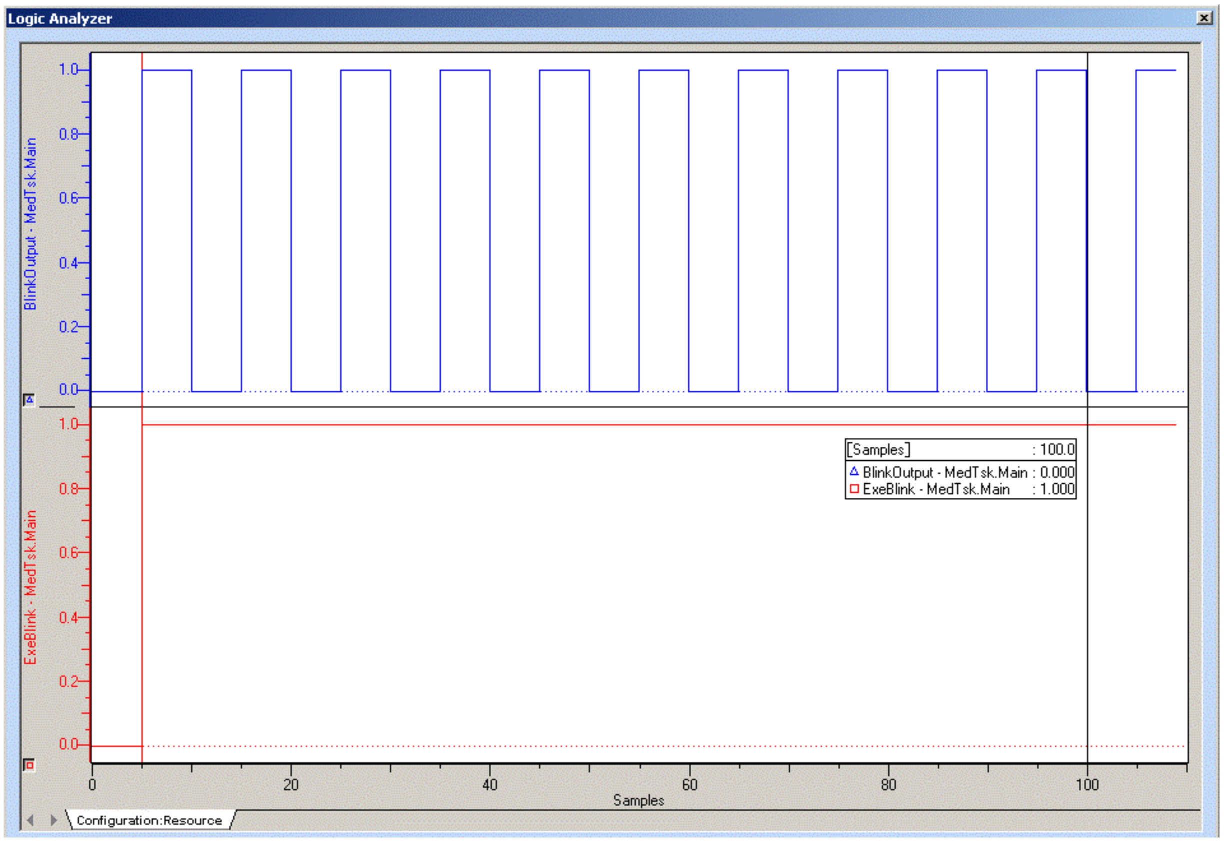
The valid output will be high if the function is operating. If Enable is held high and the Frequency is not greater than zero, the valid output will be low.

Example

Blink_1 was placed in a 10ms task so the expected output is 50ms on and 50ms off which corresponds to 5 cycles on, 5 cycles off.

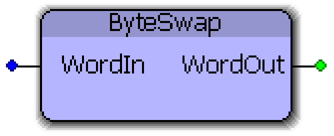


Logic Analyzer output:



```
[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.

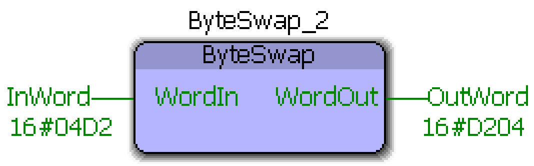
Parameters

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

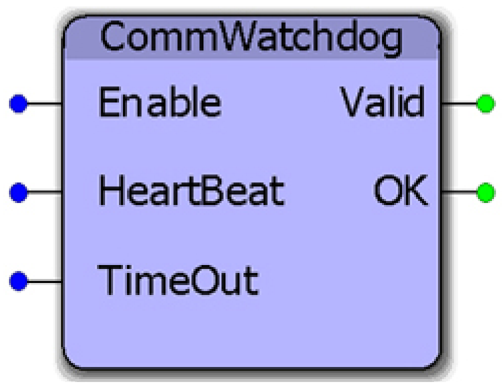
Error Description

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.

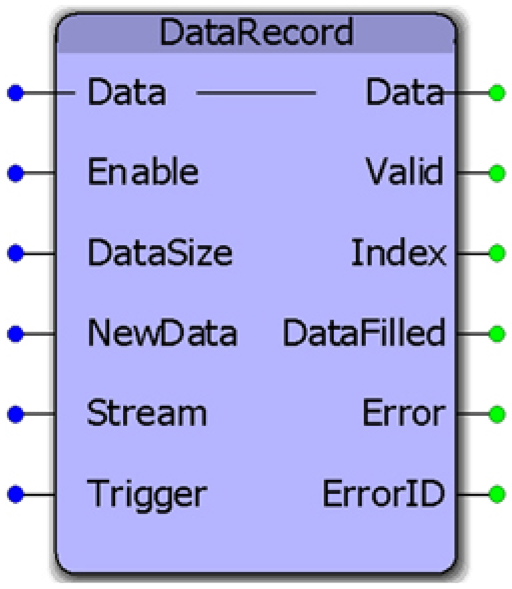
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
V	HeartBeat	DINT	Value that the master changes and sends to the MP2000iec controller.	DINT#0
V	WatchDog	DINT	The HeartBeat input must change value within the TimeOut period for the communications to be considered OK.	DINT#0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that 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.

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.

Parameters

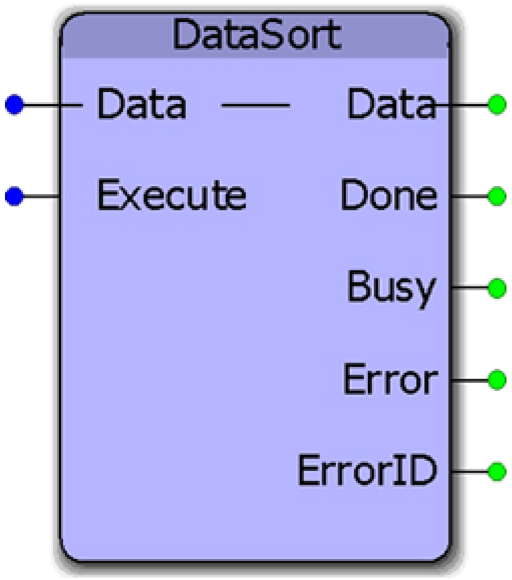
Parameter		Data Type	Description	
VAR_IN_OUT				
V	Data	XYDataStruct	Structure where recorded data is stored	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
V	DataSize	INT	The maximum amount of data to be stored, which must be less than or 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 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 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
10024	DataSize must be greater than zero

DataSort



This function block will sort data from the lowest to highest value of X data. It was designed to work with data that may be used with a cam function where the X or master data must continually increase, but this generic function can be customized for other sorting needs.

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

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

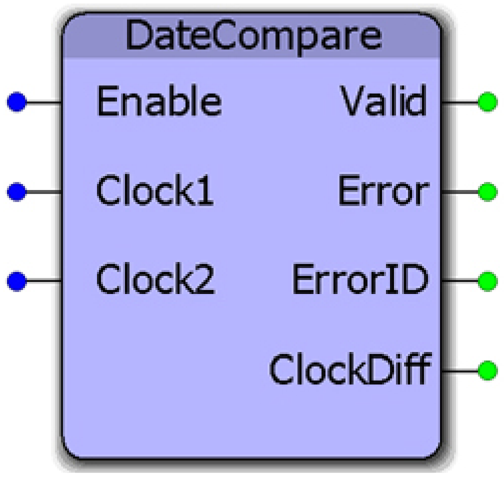
Notes

This function 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.

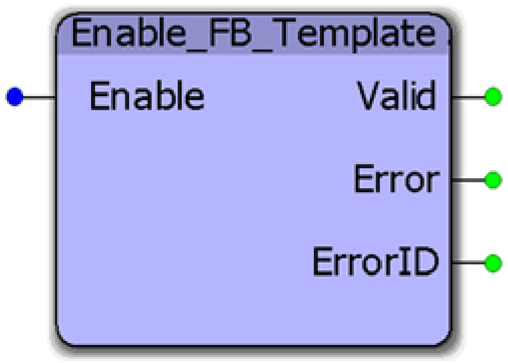
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
V	Clock1	RTCStruct	The first (older) real time clock value	N/A
V	Clock2	RTCStruct	The second (newer) real time clock value	N/A
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
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.	
V	ClockDiff	RTCStruct	Outputs the time difference between Clock1 and Clock2	

Error Description

There will be no Errors reported.

Enable_FB_Template



This function block is a template which can be used when developing functions which adhere to the PLCopen output behavior.

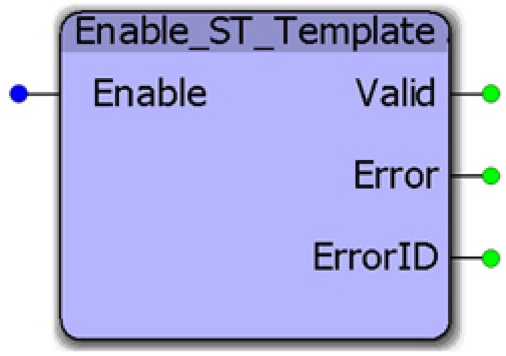
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
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.	
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.

Enable_ST_Template



This function block is a template which can be used when developing functions which adhere to the PLCopen output behavior.

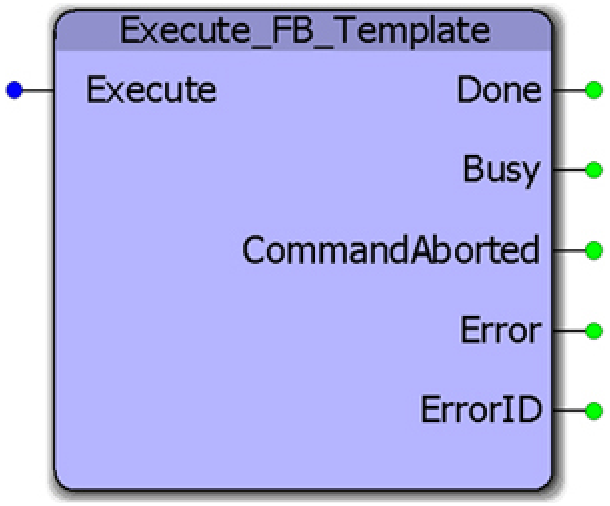
Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
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.	
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.

Execute_FB_Template



This function block is a template which can be used when developing functions which adhere to the PLCopen output behavior.

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 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	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 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.	

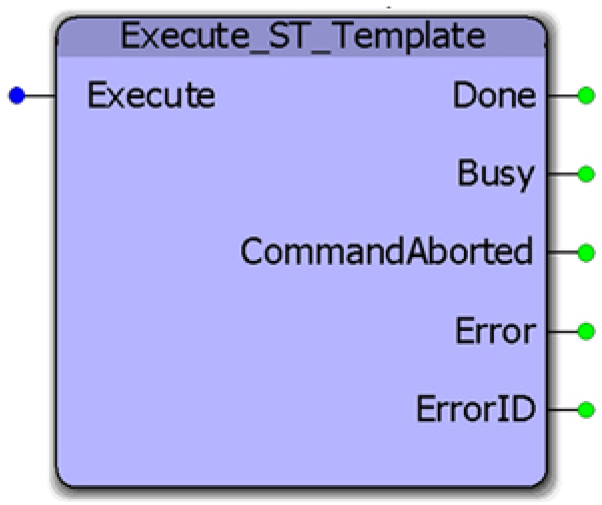
Notes

Depending on the exact usage, there may be outputs in the template that will not apply, such as CommandAborted. Please 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.

Execute_ST_Template



This function block is a template which can be used when developing functions which adhere to the PLCopen output behavior.

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 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	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 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:

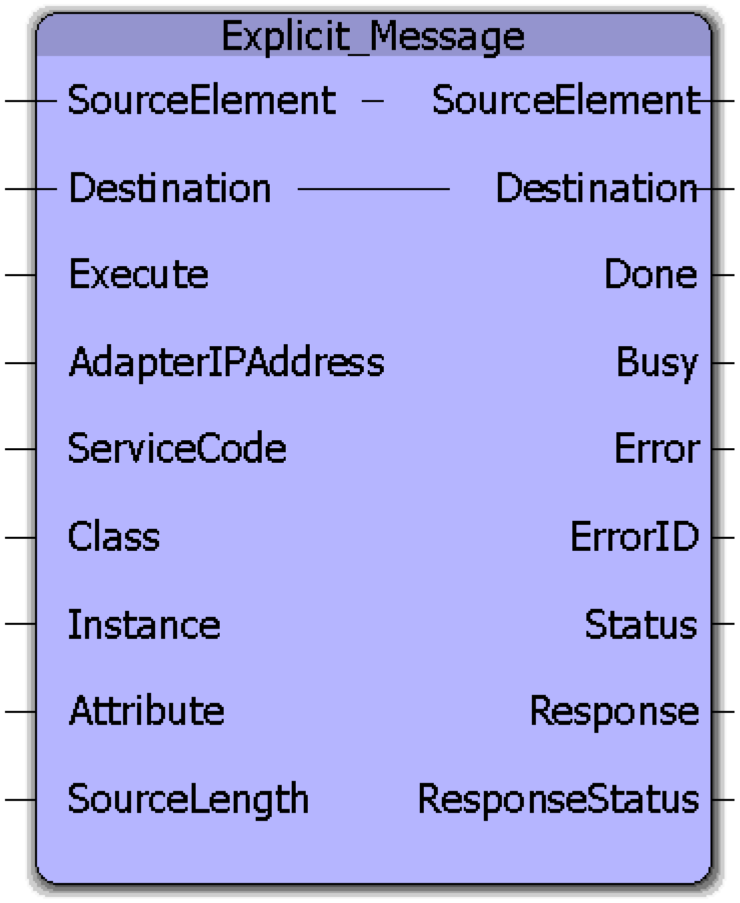
- Initialization
- Main code body
- Output status updates

Depending on the exact usage, there may be outputs in the template that will not apply, such as CommandAborted. Please 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.

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.

Parameters

*	Parameter	Data Type	Description
VAR_IN_OUT			
B	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.
B	Destination	ExplicitData	When reading a message from the Target (Adapter), the

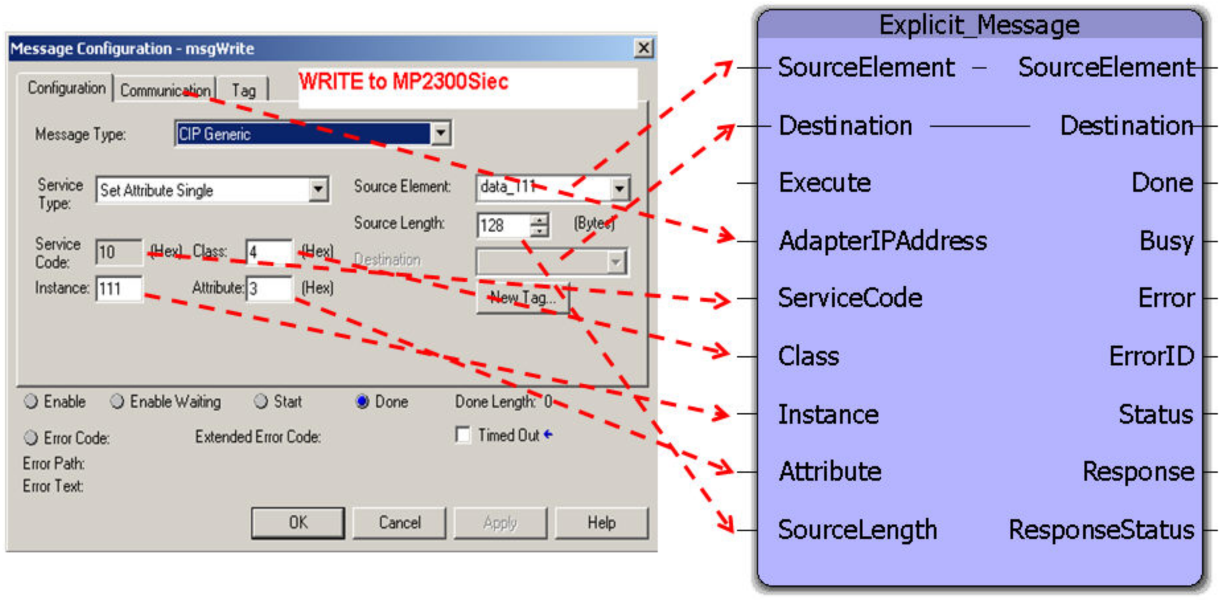
			Destination Element is the data (as an array of bytes) where the Scanner (MPiec Controller) will copy the data from the Target.	
VAR_INPUT				**Default**
B	Execute	BOOL	Upon the rising edge, all other function block inputs are read and the function is initiated. To modify an input, change the value and re-trigger the execute input.	0
B	AdapterIPAddress	STRING	IP Address of the Target device.	' '
B	ServiceCode	BYTE	Code for the particular service type as defined for a CIP message. The value can be obtained from the Target's (Adapter's) documentation.	0
B	Class	BYTE	Class parameter of a CIP Generic message. The value can be obtained from the Target's (Adapter's) documentation.	0
B	Instance	BYTE	Instance parameter of a CIP Generic message. The value can be obtained from the Target's (Adapter's) documentation.	0
B	Attribute	BYTE	Attribute parameter of a CIP Generic message. The value can be obtained from the Target's (Adapter's) documentation.	0
B	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.	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' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	
B	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.	
B	Response	WORD	Response from the Target.	
B	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 Configuration - msgWrite

WRITE to MP2300Siec

Message Type: CIP Generic

Service Type: Set Attribute Single

Service Code: 10 (Hex)

Class: 4 (Hex)

Instance: 111

Attribute: 3 (Hex)

Source Element: data_111

Source Length: 128 (Bytes)

Destination

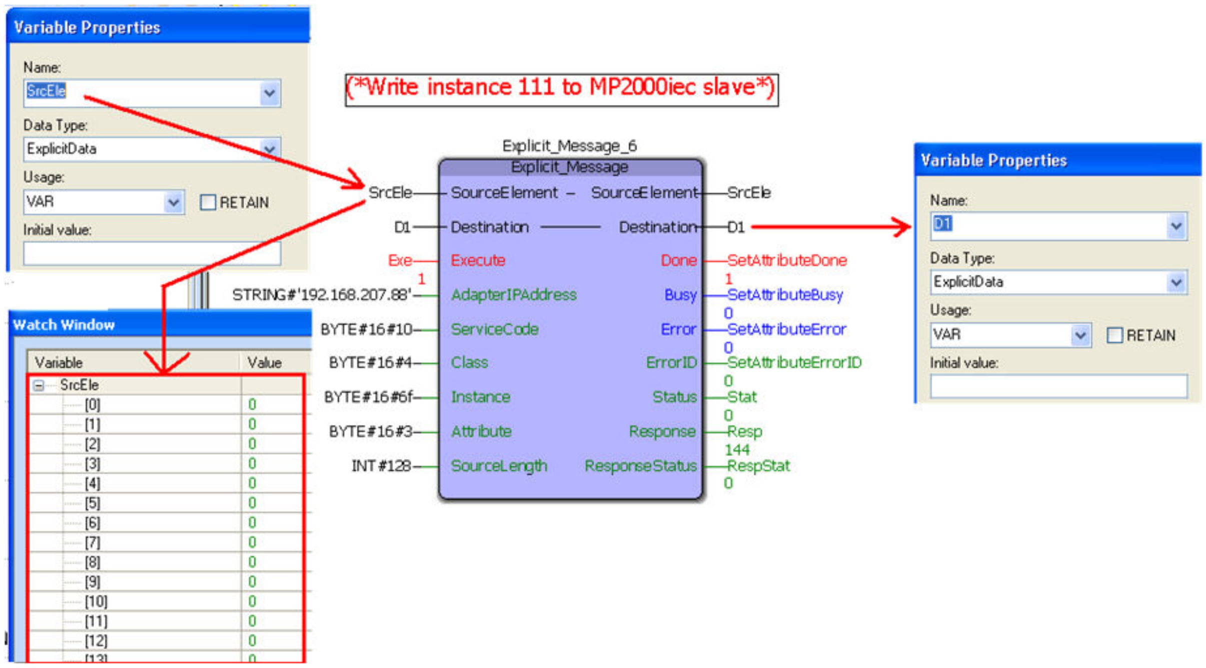
Error Description

ErrorID	Meaning
0	No error
8705	The maximum number of concurrently open user IO devices (sockets/files) has been reached.
8706	The socket handle was invalid.
8707	The IP address string was not in a valid format.
8708	The socket 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.
8711	Unable to accept new socket connection.
8712	Unable to bind to the specified address.
8713	The socket type argument was invalid.
8714	The local address or port was not valid.
8715	The socket could not be connected.
8716	There is no network routing path to the specified address.

8717	The socket is already connected to another endpoint.
8718	The socket connection attempt was actively refused by the remote peer.
8719	The socket 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	The Buffer argument to WriteDevice and ReadDevice is required.
8724	The device option ID was invalid.
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 could not be opened.

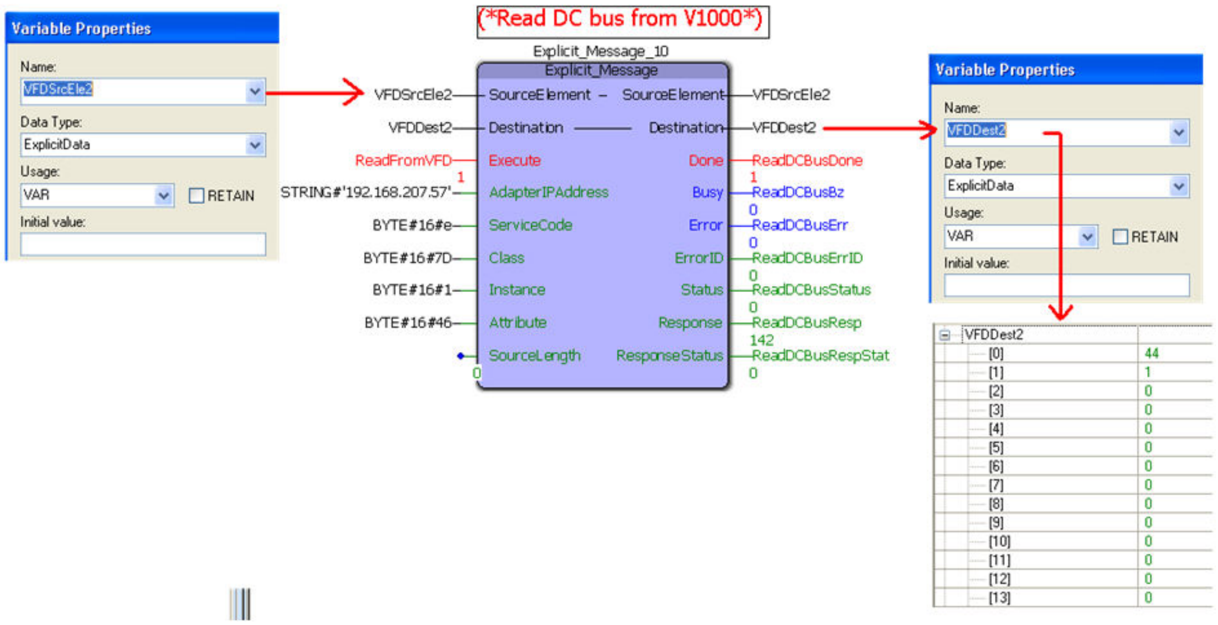
Example 1

Set Single Attribute



Example 2

Get Attribute single



(*Read DC bus from V1000*)

Variable Properties
Name: VFDSrcEle2
Data Type: ExplicitData
Usage: VAR
RETAIN
Initial value:

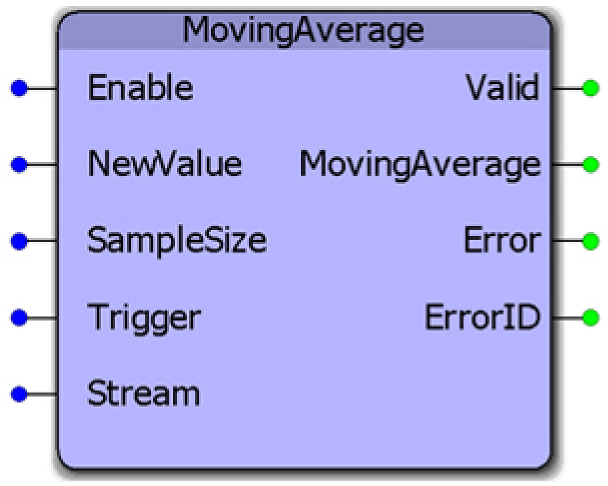
Explicit_Message_10
Explicit_Message

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

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.

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	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 outputs of the function are valid.	
V	MovingAverage	LREAL	The moving average of all the samples.	
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.	
B	ErrorID	UINT	If Error is true, this output provides the Error ID. This output is reset when 'Execute' or 'Enable' goes low.	

Notes

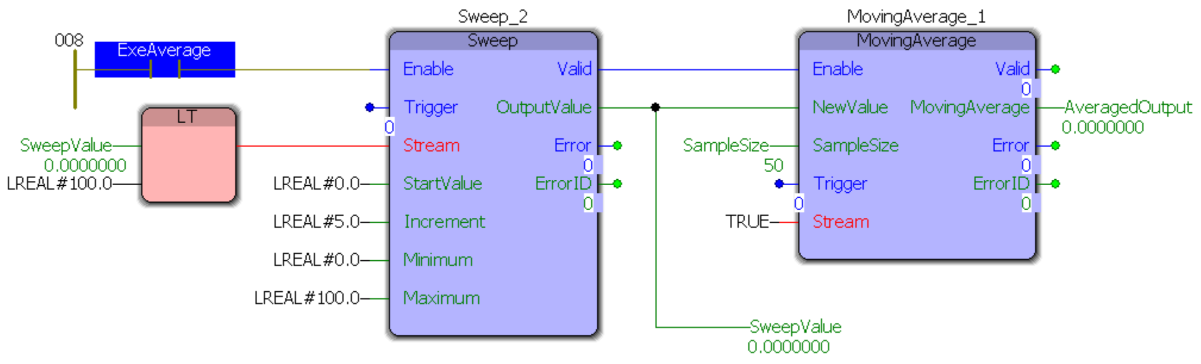
- See Yaskawa's Youtube webinar - MPiec Web Tension Control Applications for more info on using this function block.

Error Description

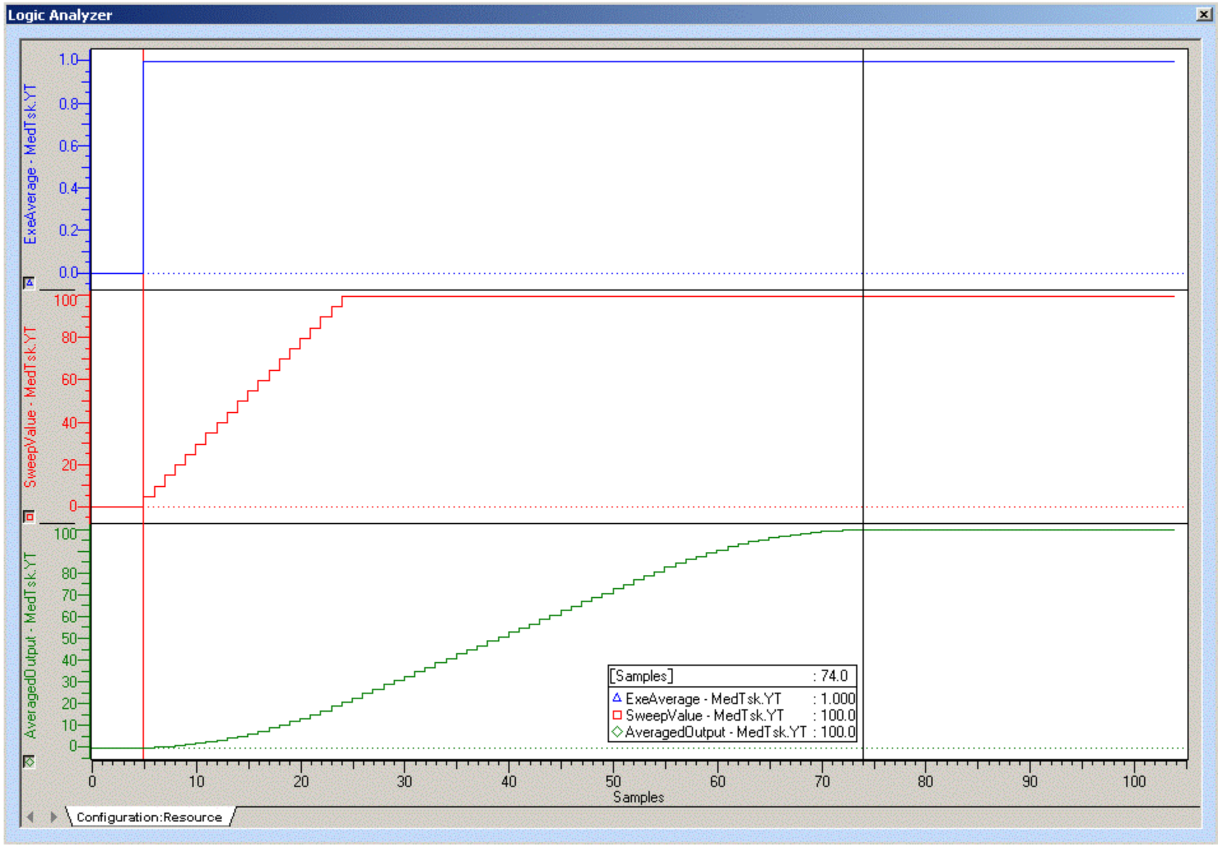
ErrorID	Meaning
0	No error
10024	DataSize must be greater than zero

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.

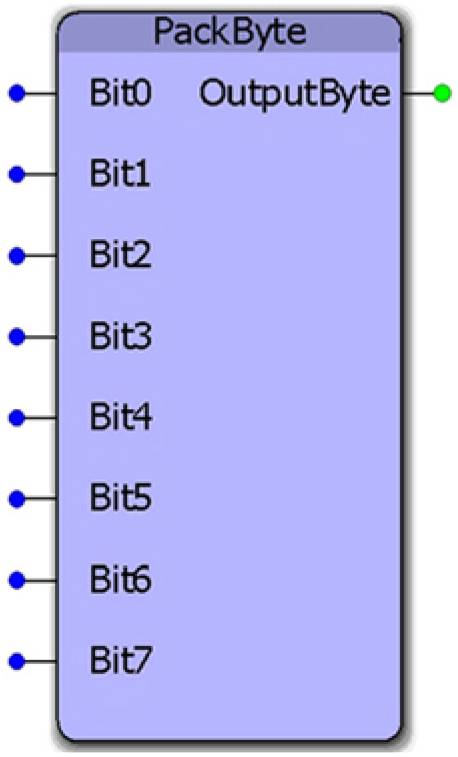


The Moving average function has a sample size of 50 which means that if Sweep reaches its maximum value after 19 cycles, MovingAverage will output the maximum value after 69 cycles. By looking at the Logic Analyzer plot below, notice there is a 5 cycle pre-record that must be taken in to account: 74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

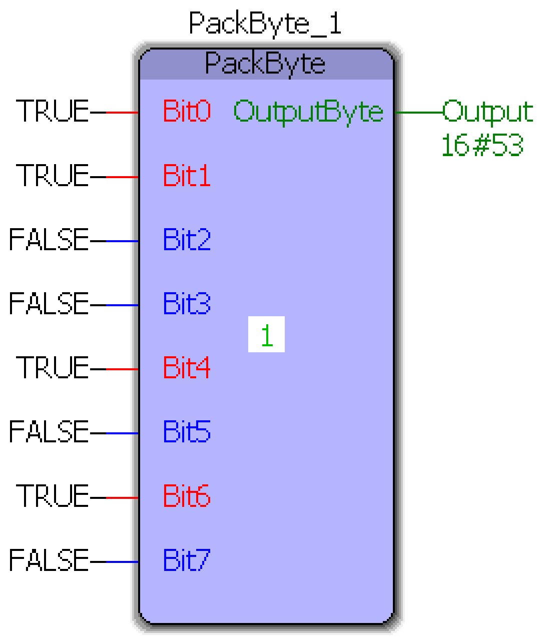
Parameters

*	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	

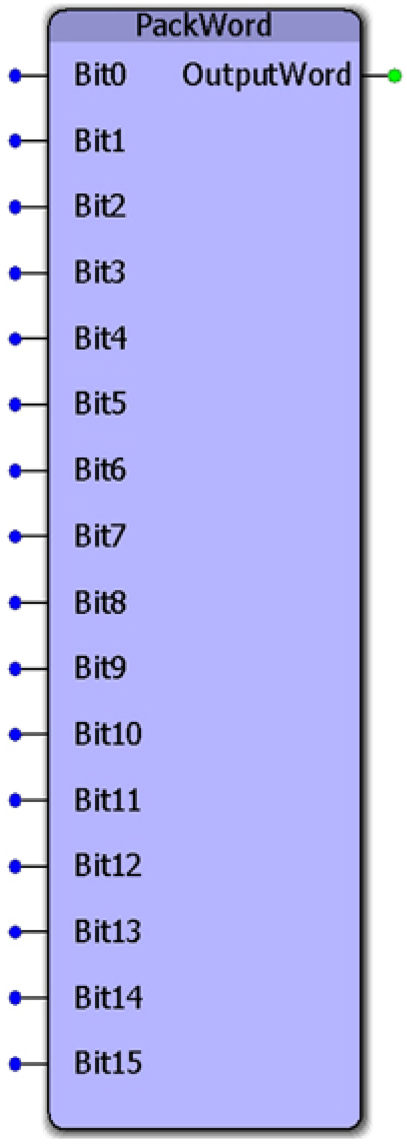
Error Description

No errors will be generated

Example



PackWord



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

Parameters

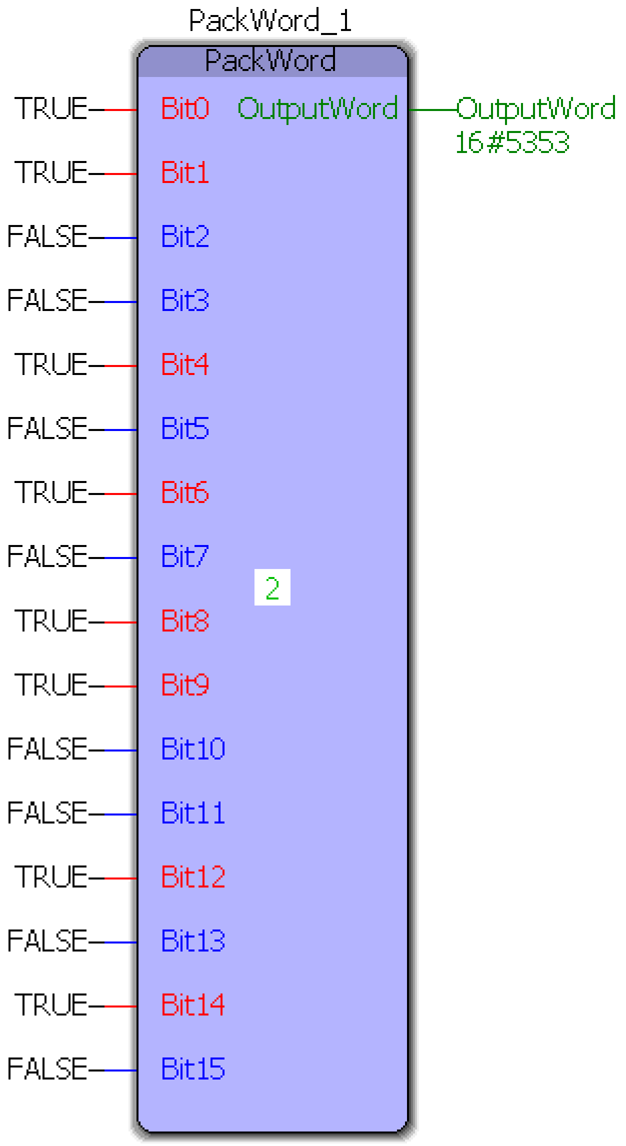
*	Parameter	Data Type	Description	
VAR_INPUT				Default
V	Bit0	BOOL	Bit 0 of the WORD to be output	
V	Bit1	BOOL	Bit 1 of the WORD to be output	
V	Bit2	BOOL	Bit 2 of the WORD to be output	
V	Bit3	BOOL	Bit 3 of the WORD to be output	

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	

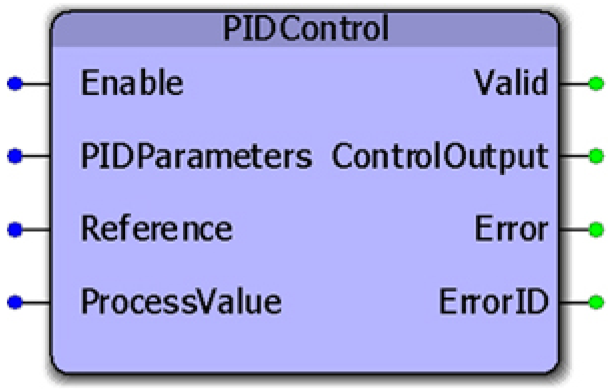
Error Description

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.

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	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 outputs of the function are valid.	
V	ControlOutput	BOOL	Output value from the PID control block. The range of values will be governed by the PIDParameters, especially the upper and lower limit.	
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.	
B	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:

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

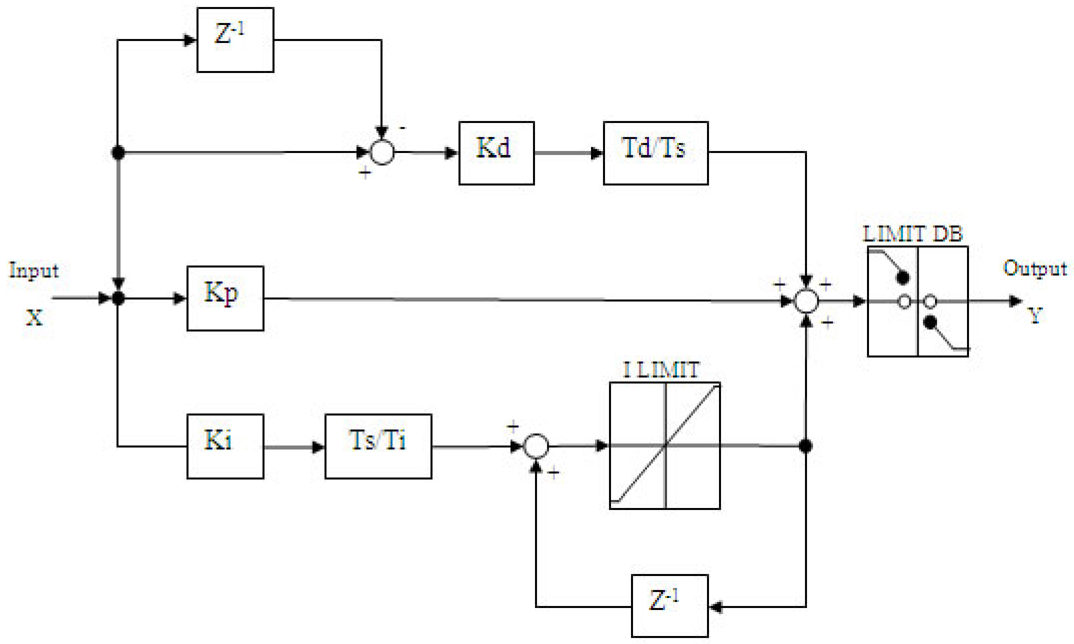
$$\frac{Y}{X} = Kp + Kd.Td.S$$

X: deviation input value; Y: output value

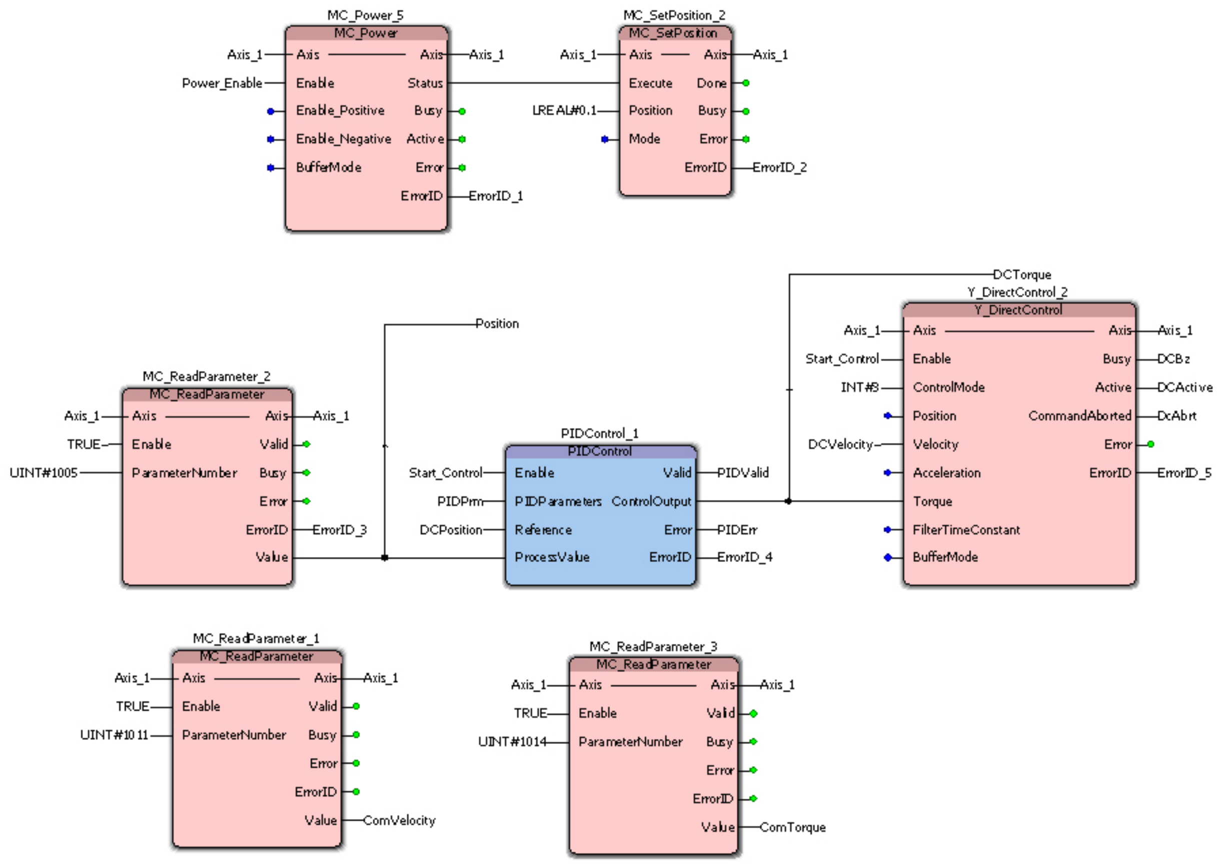
The following operation is performed within the PID instruction:

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

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

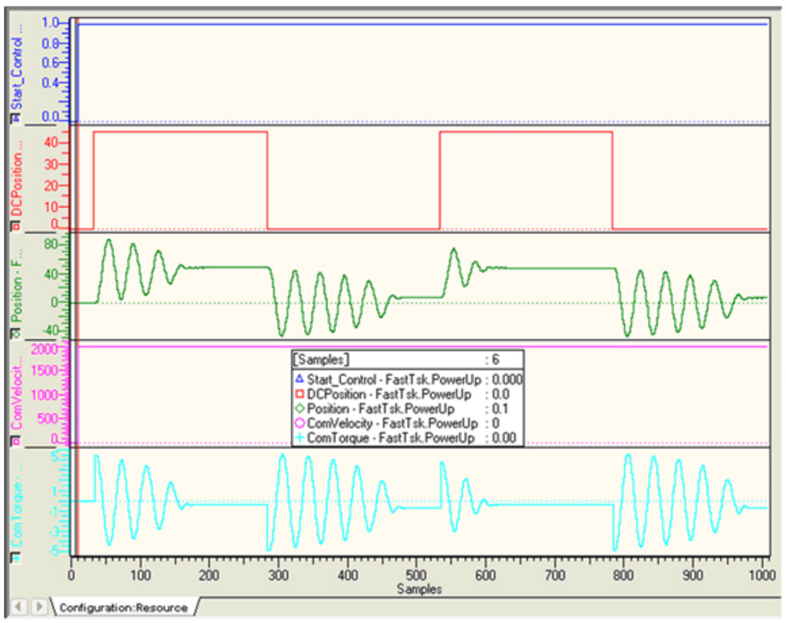


1. An example controlling a servo in torque mode:

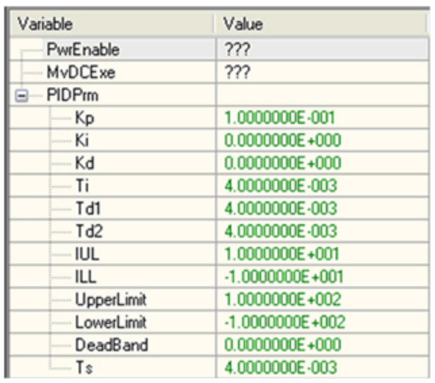


The following series of graphs show changes made to the PID gains to minimize error:

a. Proportional Control Only. Severe oscillation:

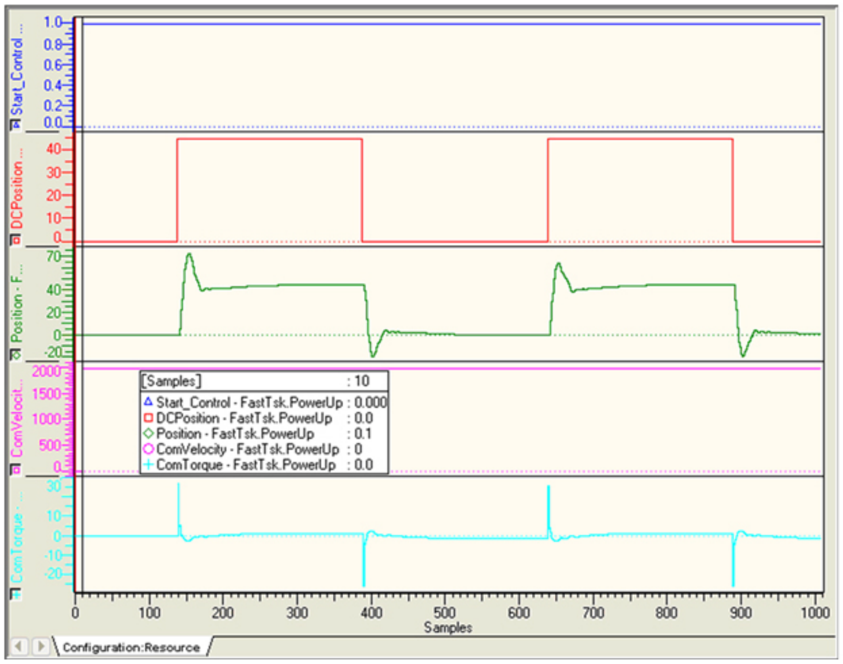


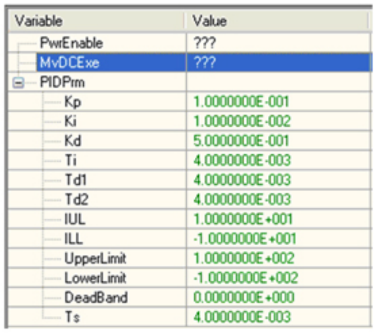
```
[Samples]                          : 6
Start_Control - FastTsk.PowerUp    : 0.000
DCPosition - FastTsk.PowerUp       : 0.0
Position - FastTsk.PowerUp         : 0.1
ComVelocity - FastTsk.PowerUp      : 0
ComTorque - FastTsk.PowerUp        : 0.00
```



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

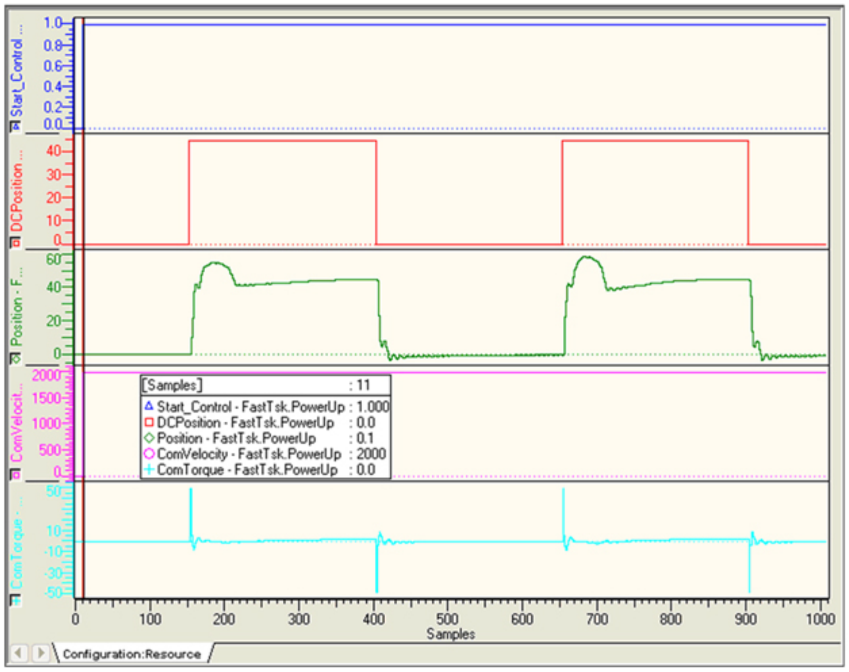
b. PID Control. Derivative helps to control oscillation:

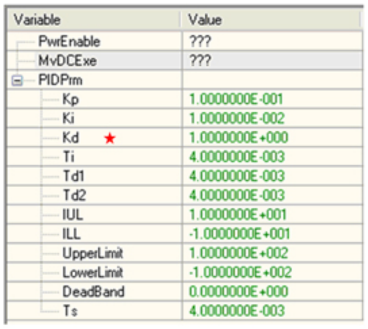




Variable	Value
PwrEnable	???
MvDCExe	???
PIDPrm	
Kp	1.0000000E-001
Ki	1.0000000E-002
Kd	5.0000000E-001
Ti	4.0000000E-003
Td1	4.0000000E-003
Td2	4.0000000E-003
IUL	1.0000000E+001
ILL	-1.0000000E+001
UpperLimit	1.0000000E+002
LowerLimit	-1.0000000E+002
DeadBand	0.0000000E+000
Ts	4.0000000E-003

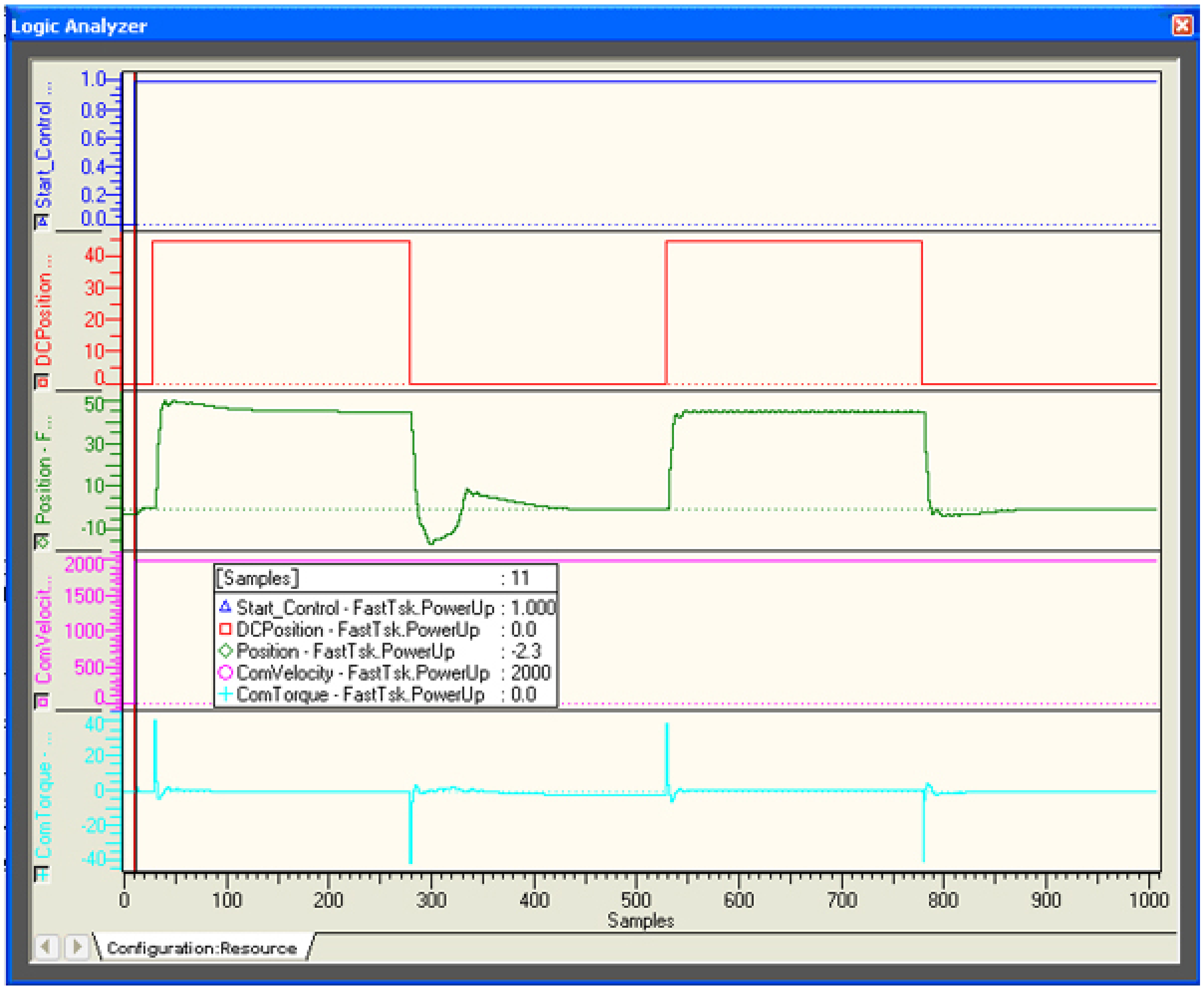
c. PID Control – Increasing the derivative gain:



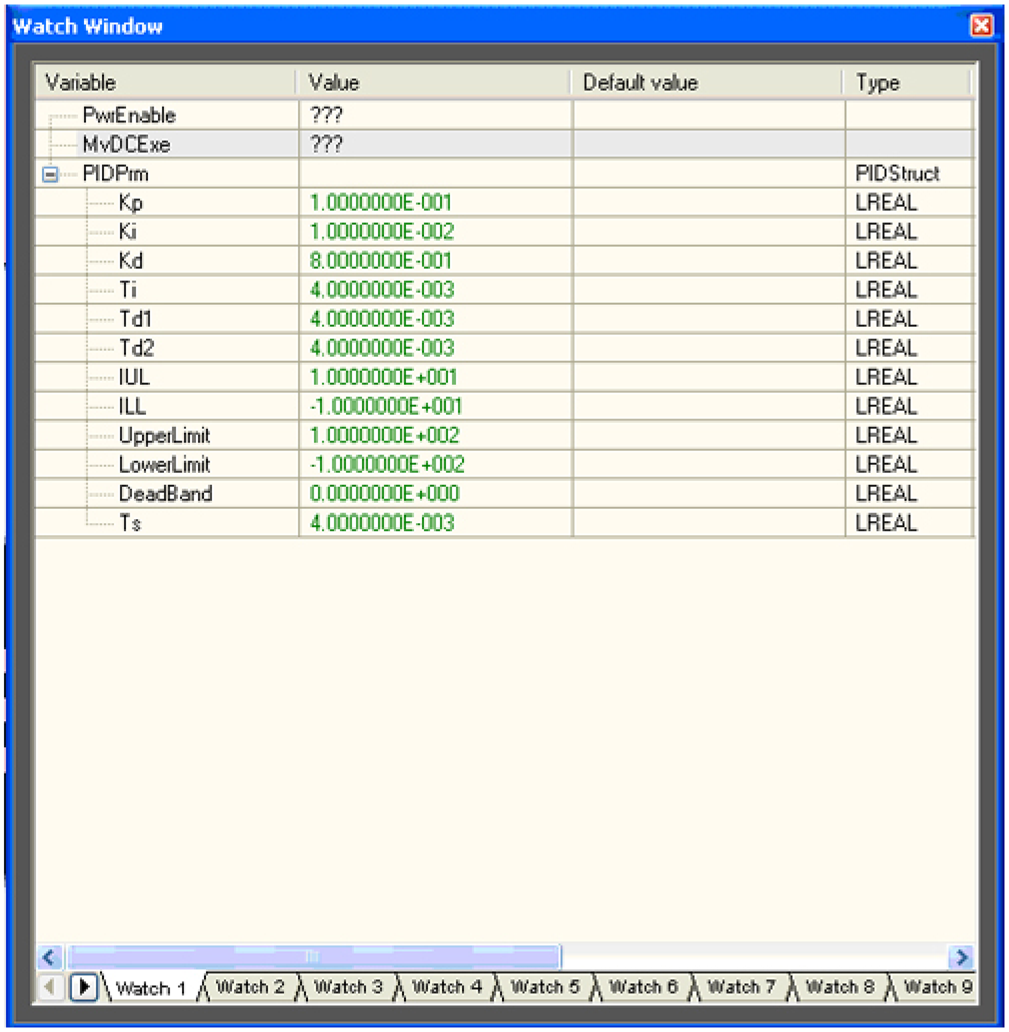


Variable	Value
PwrEnable	???
MvDCExe	???
PIDPrm	
Kd ★	1.0000000E+000
Kp	1.0000000E-001
Ki	1.0000000E-002
Ti	4.0000000E-003
Td1	4.0000000E-003
Td2	4.0000000E-003
IUL	1.0000000E+001
ILL	-1.0000000E+001
UpperLimit	1.0000000E+002
LowerLimit	-1.0000000E+002
DeadBand	0.0000000E+000
Ts	4.0000000E-003

d. Further increase in the derivative gain:

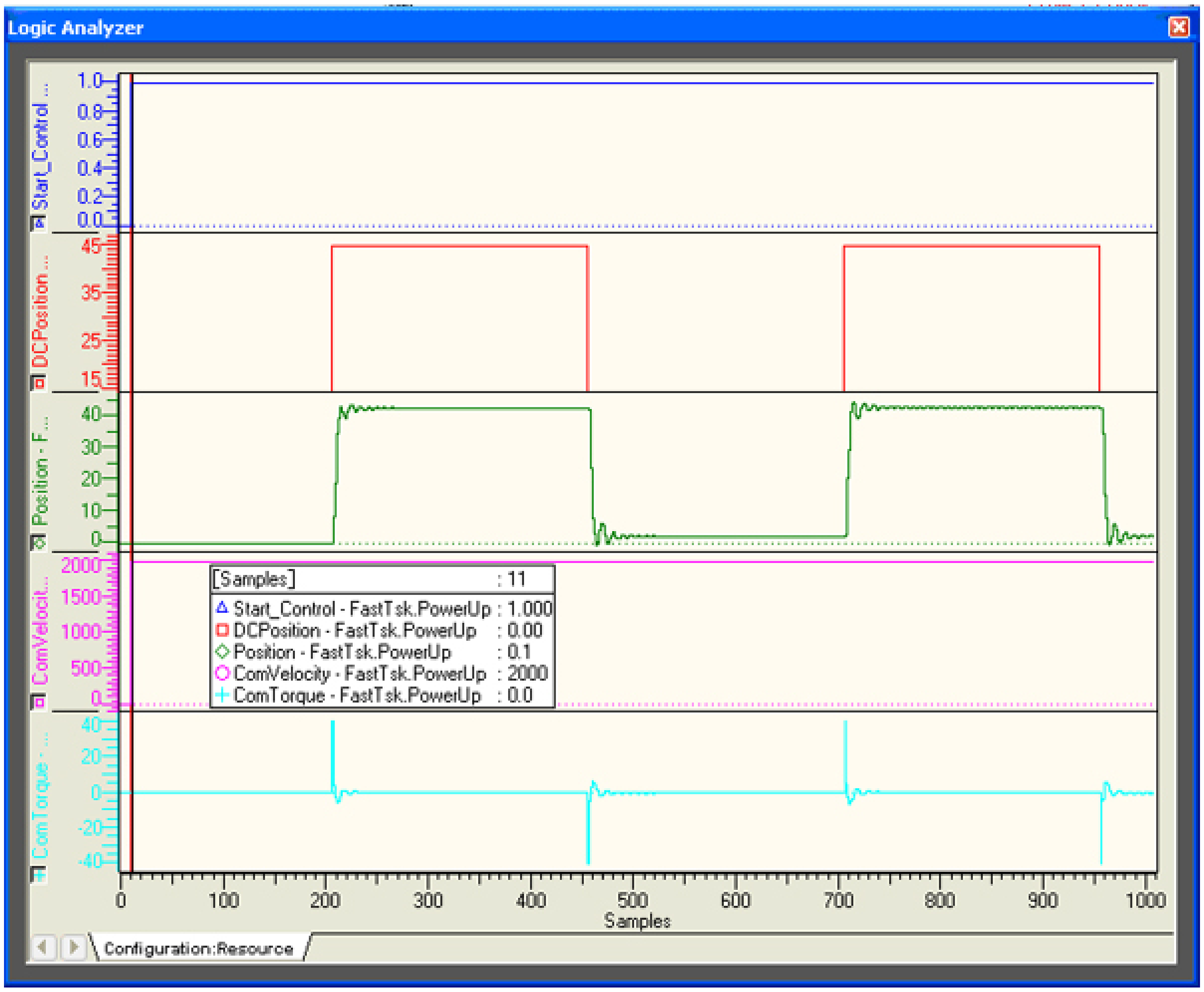


```
[Samples]                          : 11
Start_Control - FastTsk.PowerUp : 1.000
DCPosition - FastTsk.PowerUp    : 0.0
Position - FastTsk.PowerUp      : -2.3
ComVelocity - FastTsk.PowerUp   : 2000
ComTorque - FastTsk.PowerUp     : 0.0
```

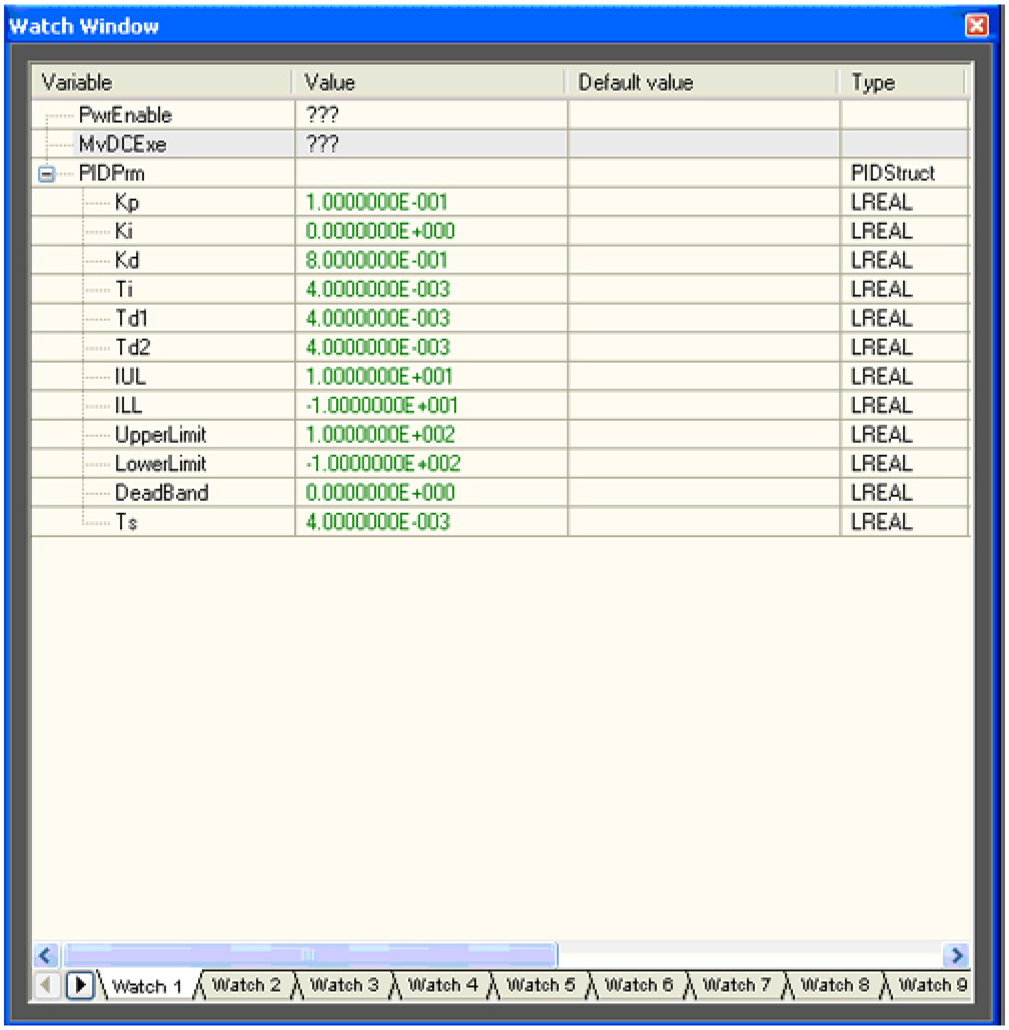


Variable	Value	Default value	Type
PwrEnable	???		
MvDCExe	???		
PIDPrm			PIDStruct
Kp	1.0000000E-001		LREAL
Ki	1.0000000E-002		LREAL
Kd	8.0000000E-001		LREAL
Ti	4.0000000E-003		LREAL
Td1	4.0000000E-003		LREAL
Td2	4.0000000E-003		LREAL
IUL	1.0000000E+001		LREAL
ILL	-1.0000000E+001		LREAL
UpperLimit	1.0000000E+002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

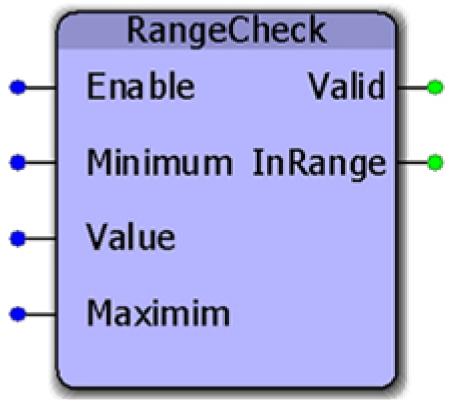


```
[Samples]                           : 11
Start_Control - FastTsk.PowerUp : 1.000
DCPosition - FastTsk.PowerUp    : 0.00
Position - FastTsk.PowerUp      : 0.1
ComVelocity - FastTsk.PowerUp   : 2000
ComTorque - FastTsk.PowerUp     : 0.0
```



Variable	Value	Default value	Type
PwrEnable	???		
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.

Parameters

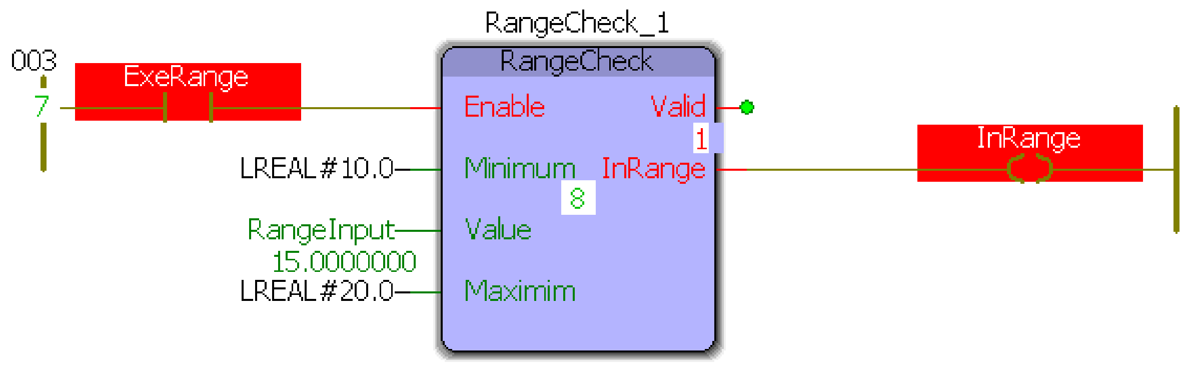
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
V	Minimum	LREAL	The smallest 'Value' that will set the InRange output high.	LREAL#0.0
V	Value	LREAL	The data to be tested against the Minimum and Maximum.	LREAL#0.0
V	Maximum	LREAL	The largest 'Value' that will set the InRange output high.	LREAL#0.0
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
V	InRange	BOOL	Indicates if the Value is between the Minimum and Maximum. (Inclusive)	

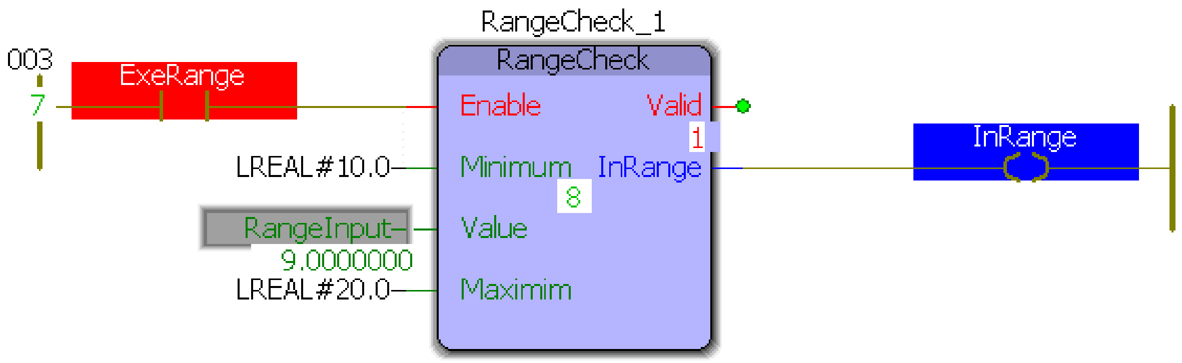
Error Description

No errors will be generated.

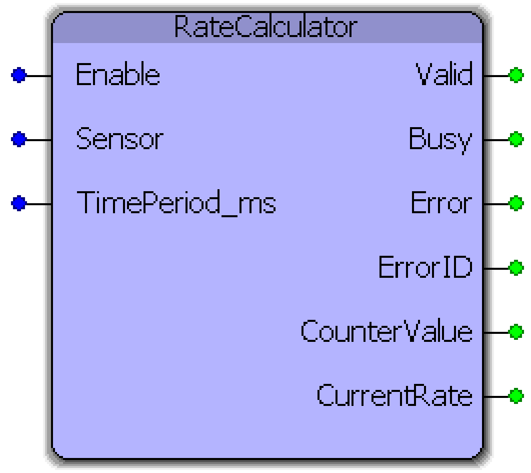
Example

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





RateCalculator



This function block determines the frequency and number of occurrences of an event, such as determining the part output rate of a machine. RateCalculator counts the number of times an input 'Sensor' signal produces a rising edge and determines the frequency of that signal with respect to a chosen time period. It can account for real-time changes to the time period.

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	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 outputs of the function are valid.	
B	Busy	BOOL	Set high upon the rising edge of the 'Execute' or 'Enable' input, and reset if Done, CommandAborted, or Error is true.	
B	Error	BOOL	Set high if error has occurred during the execution of the function block. This output is cleared when 'Execute' or 'Enable' goes low.	
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

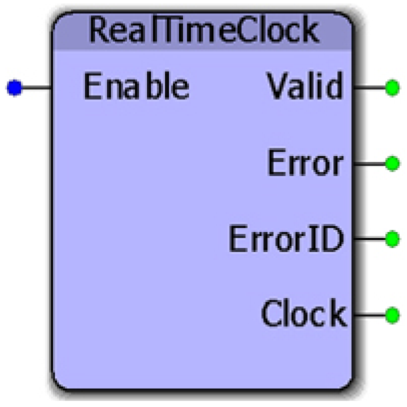
Notes

- 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.

Parameters

*	Parameter	Data Type	Description	
VAR_INPUT				Default
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
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.	
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

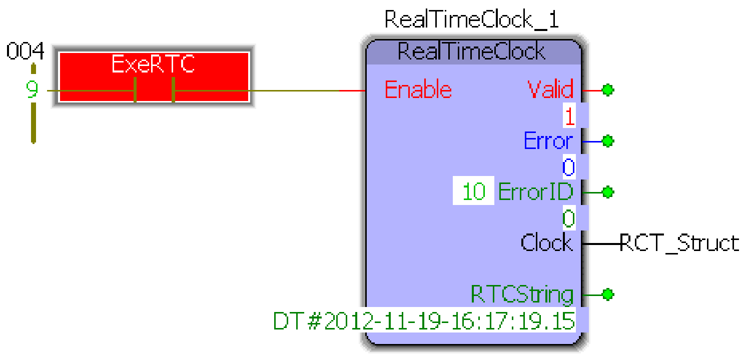
The controllers clock can be set from the web server, or by using the Y_SetRTC function block, which requires firmware version 2.0.0 or greater.

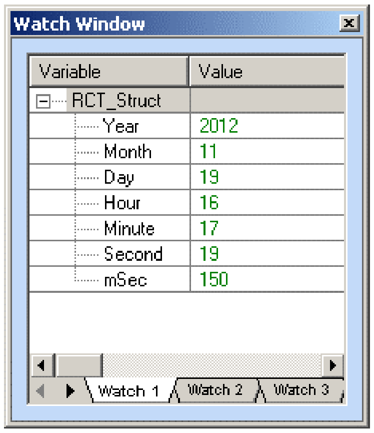
Error Description

No errors will be generated.

Example

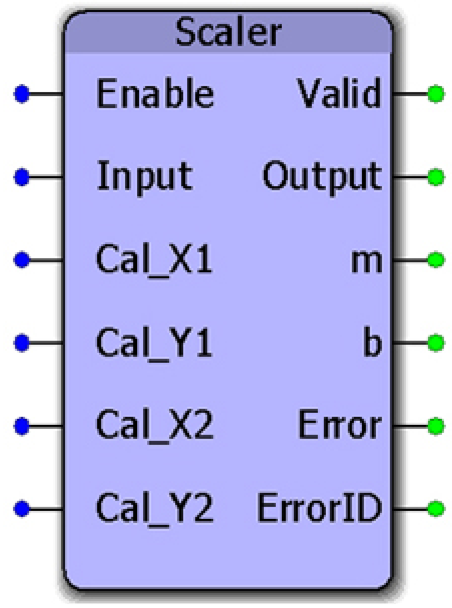
The output of this block is continually updated as long as Enable is TRUE.





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.

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.

This function can be used with temperature sensors or any analog value that must be adjusted before further processing takes place.

Parameters

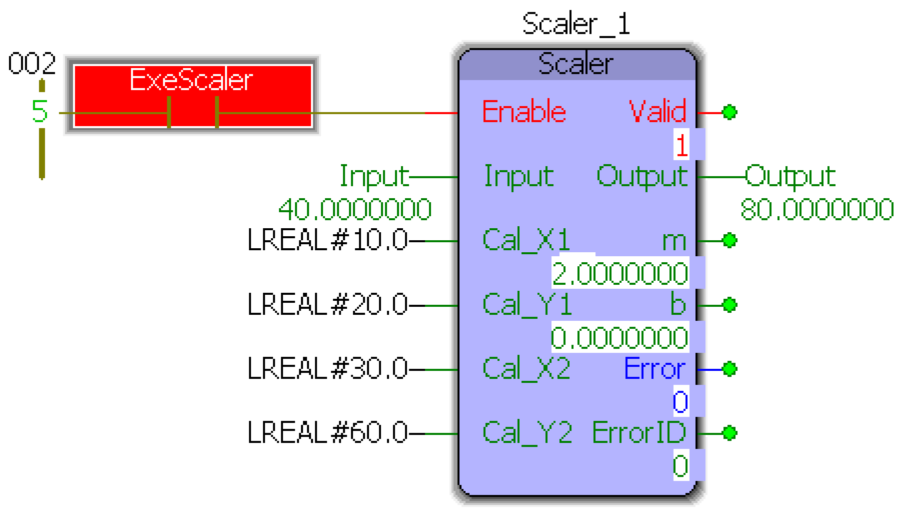
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
V	Input	LREAL	The x variable of y:=mx + b.	LREAL#0.0
V	CalX1	LREAL	Datapoint specifying a line along which data is to be scaled.	LREAL#0.0
V	CalY1	LREAL	Datapoint 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	LREAL#0.0

			be scaled.	
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 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 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.	

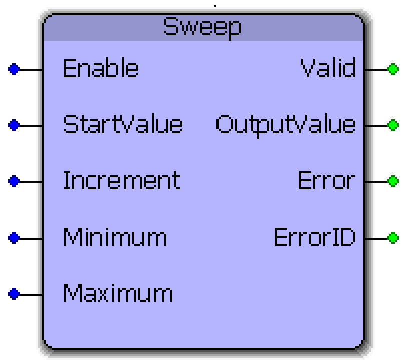
Error Description

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

Example



Sweep



This function block generates an output that rises and falls between the minimum and maximum outputs specified by the inputs. The OutputValue is the changed by the Increment input. This function block is useful for testing purposes by forcing other portions of application code to be tested with a full range of expected values.

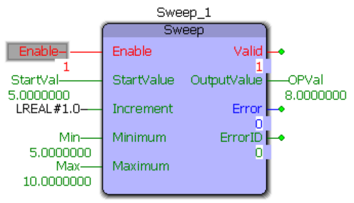
Parameters

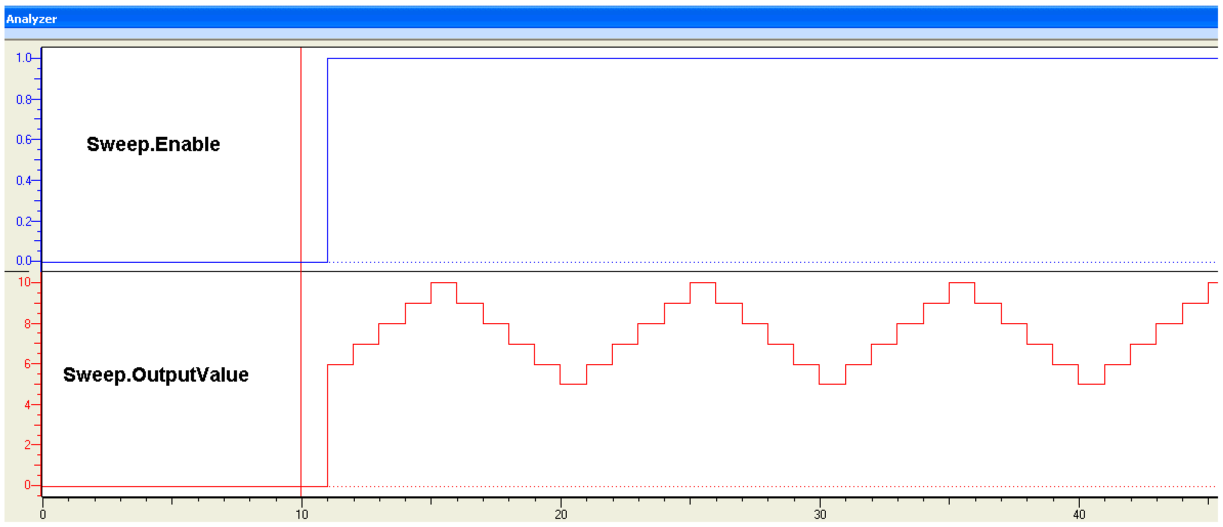
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
B	StartValue	LREAL	The OutputValue will start from this value	LREAL#0.0
B	Increment	LREAL	The amount by which the Outputvalue is changed each scan	LREAL#0.0
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 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

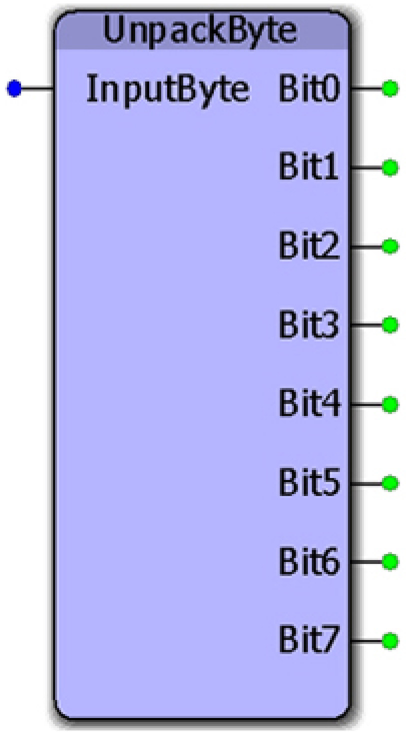
No errors will be generated.

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
Error
0
ErrorID
0
OPVal
8.0000000


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

UnpackByte



This function block converts a byte into discrete bits.

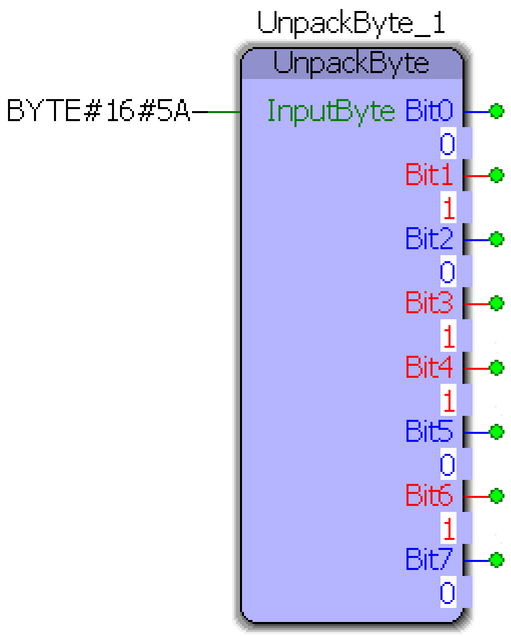
Parameters

*	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	

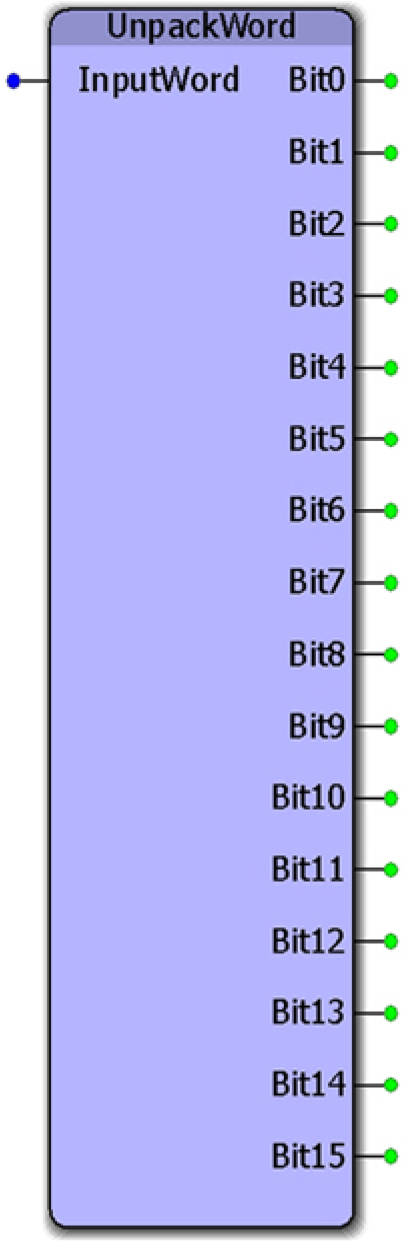
Error Description

No errors will be generated.

Example



UnpackWord



This function block separates a word into individual bits.

Parameters

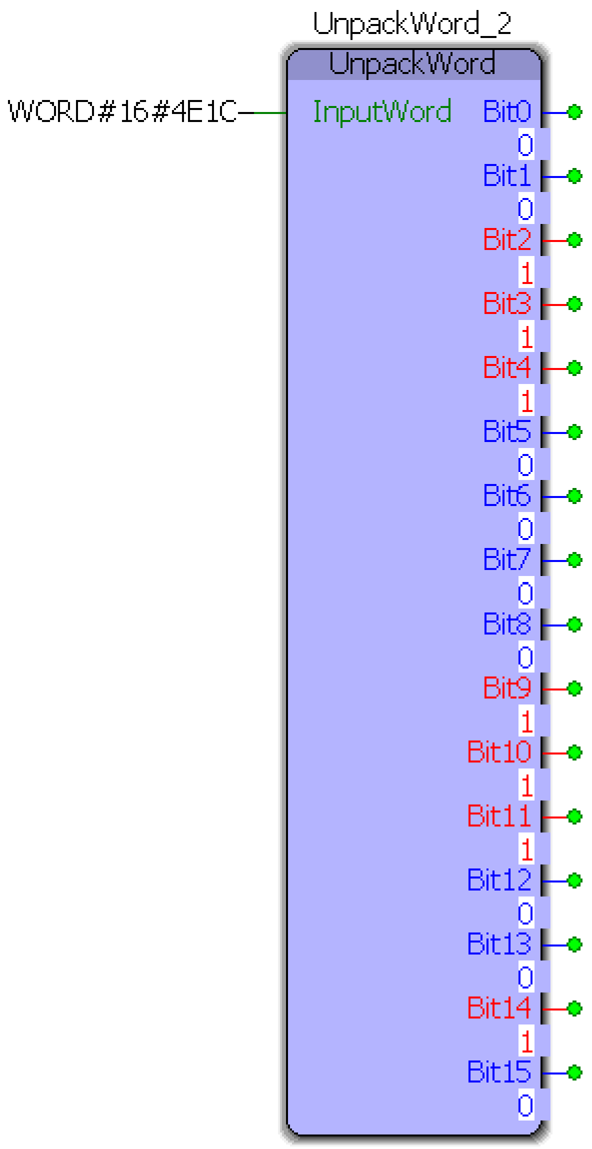
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	InputWord	WORD	The input data to be separated into bits.	WORD#0

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

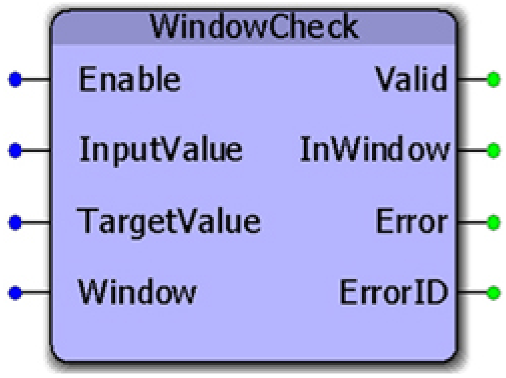
Error Description

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.

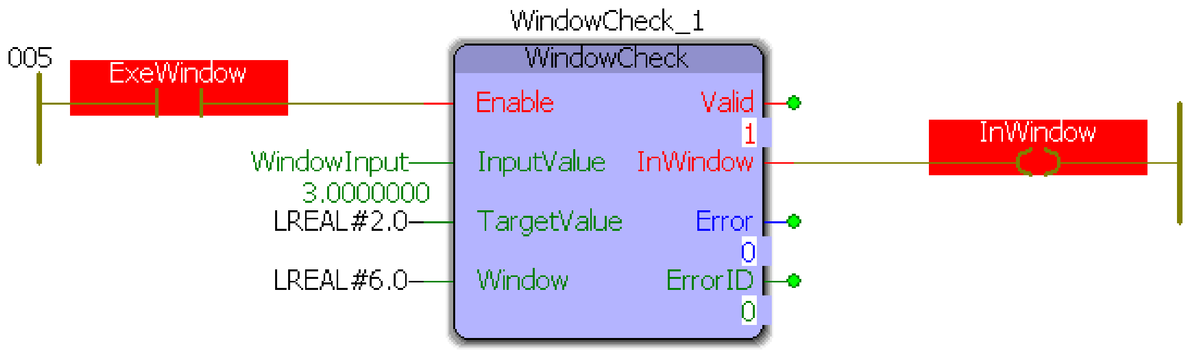
Parameters

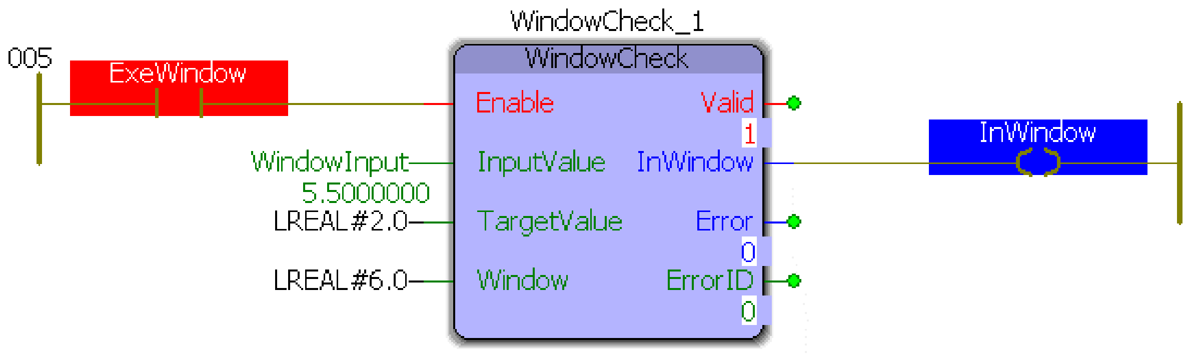
*	Parameter	Data Type	Description	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
V	InputValue	LREAL	The data to be tested against the TargetValue	LREAL#0.0
V	TargetValue	LREAL	The desired data to be compared against.	LREAL#0.0
V	Window	LREAL	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 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 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

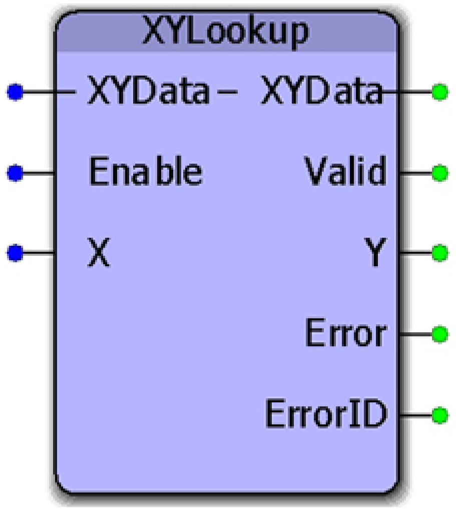
ErrorID	Meaning
0	No error
10076	WindowSize must be greater than zero.

Example





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.

Parameters

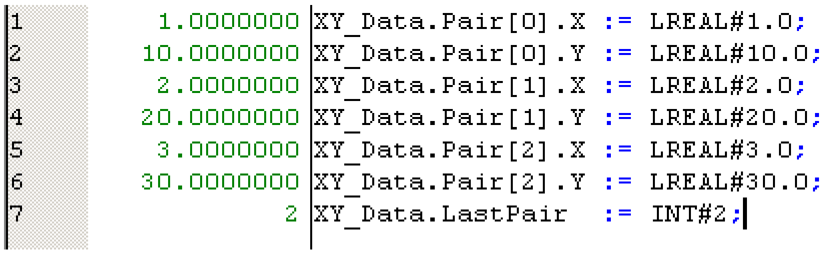
*	Parameter	Data Type	Description	
VAR_IN_OUT				
V	XYData	XYDataStruct	An array of X & Y data pairs	
VAR_INPUT				**Default**
B	Enable	BOOL	The function will continue to execute while enable is held high.	FALSE
V	X	LREAL	The input reference	
VAR_OUTPUT				
B	Valid	BOOL	Indicates that the outputs of the function are valid.	
V	Y	LREAL	The resulting output that relates the input.	
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.	
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

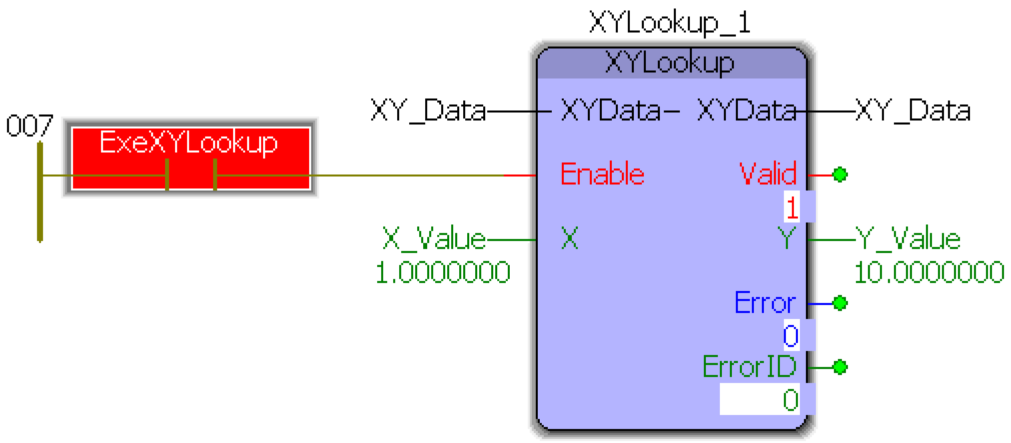
ErrorID	Meaning
0	No error
10038	CamData.LastSegment must be greater than 0 and less than 400, or whatever value has been declared as the ARRAY size in the CTB_Types file.

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;
```



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 axis motion queue is full. 16 moves is the maximum which can be buffered.
4370	The move could not be started because motion is prohibited. The drive may not be enabled. MC_Power.Enable_Positive or MC_Power.Enable_Negative might be low. Check MC_Power.Status output. MC_Stop.Execute might be held high, preventing motion.
4371	The servo drive failed to enable or disable. Check the amplifier wiring for L1 / L2 / L3
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 can not be executed while the axis is moving.
4381	Motion aborted due to axis alarm. It is also possible that a software limit has been exceeded.
4382	When the axis is in 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.
4390	Position cannot be defined while the axis is the cam master of other axes.
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 cams Machine Cycle. (0 to Prm1502, the last master position in the active cam table.)
4396	Axis latch function already in use.
4397	Over travel limit still ON after attempting to move away from it.
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 (HHB) is preventing the drive from enabling.
4401	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.
4404	Can not execute MC_GearOut because axis is not in gear.

4405	Y_CamOut was aborted.
4406	Continuous Latch Mode 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 in progress.
4410	Parameter is read-only.
4411	Parameter read in progress.
4412	Parameter not supported for this axis.
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.
Invalid Structure Value	
4624	Axis latch function already in use.
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.
4626	The master slave relationship is defined. A slave cannot be a master to another axis.
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
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.
4646	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.
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.
4663	Specified time was less than zero.
4665	Velocity parameter is negative.

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.
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.
4677	Array size too large.
4678	Buffer array index out of range.
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. Cam tables must have a header indicating the number of rows, number of columns and a feed forward velocity flag.
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.
4894	The specified virtual axis may not be used with this function block.
4895	File extension is not recognized or missing.
4896	Cound not find the axis parameter file.
4897	The drive's model number or type does not match the parameter file.
4898	No filter configured for axis.
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 IO devices (sockets/files) has been reached.
8706	The socket handle was invalid.
8707	The IP address string was not in a valid format.
8708	The socket 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.
8711	Unable to accept new socket connection.
8712	Unable to bind to the specified address.
8713	The socket type argument was invalid.
8714	The local address or port was not valid.

8715	The socket could not be connected.
8716	There is no network routing path to the specified address.
8717	The socket is already connected to another endpoint.
8718	The socket connection attempt was actively refused by the remote peer.
8719	The socket 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	The Buffer argument to WriteDevice and ReadDevice is required.
8724	The device option ID was invalid.
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 could not be opened.
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	Might be crossed or the same non-zero value
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. Path.Data[Segment].SegmentType must be coded as either being a line (INT#1) or an arc (INT#2).
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
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.
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).
10114	Incorrect cam table size (check the CamTable.Header.Datasize)
10116	Problem converting string data to the output buffer
10117	String Conversion Error already exists on the controller. Clear the alarm 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.
10121	CSV file contains an unsupported version.
10123	Column Start Error. The data is corrupted.
10124	Unsupported Case condition.
10125	Conversion Error. Check the ErrorRow and ErrorCol outputs for details
10126	NoDataError - The End Of File was reached, but the record count is zero
10127	TooManyRecords - DataType is not large enough
10128	MaxNotDefined - User must set the maximum number of records that can be added to 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
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 unrecognised
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	Transcation 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.
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.
Kernel Error	
61713	An internal assertion in the motion kernel failed indicating the controller is not in a stable state. Please report this error to Yaskawa America Incorporated.

Please refer to the following manuals for details regarding servo amplifier errors:

- Sigma II with NS115: SIEPC71080001, see section 9.3
- Sigma III: YEA-SIA-S800-11, see section 10.1.4
- Sigma-5 with rotary motor: SIEPS8000043, see Section 6.1
- Sigma-5 with linear motor: SIEPS8000044, see Section 6.1