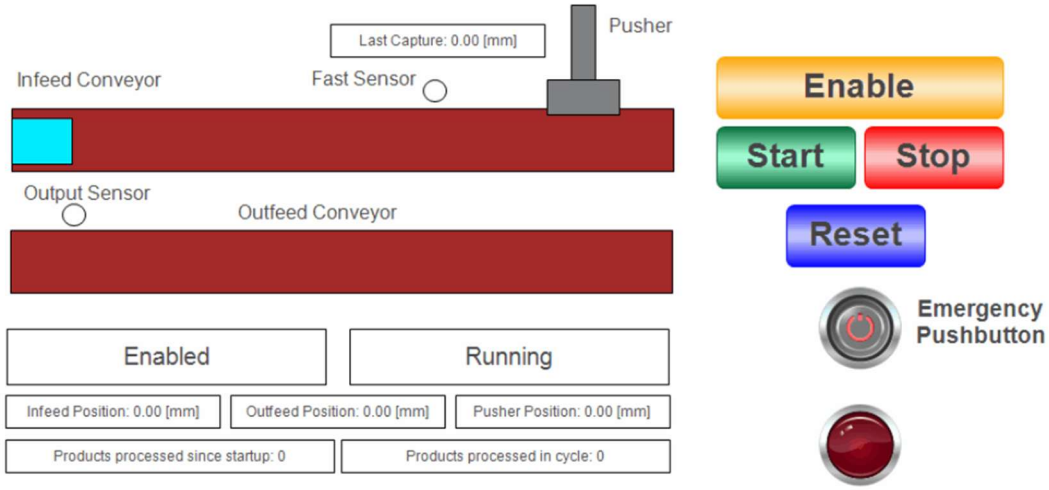
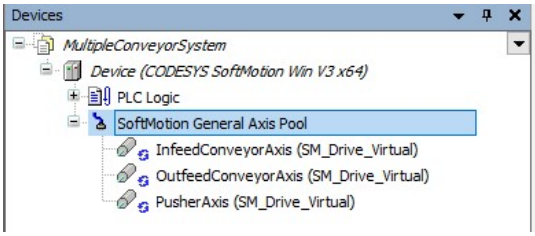
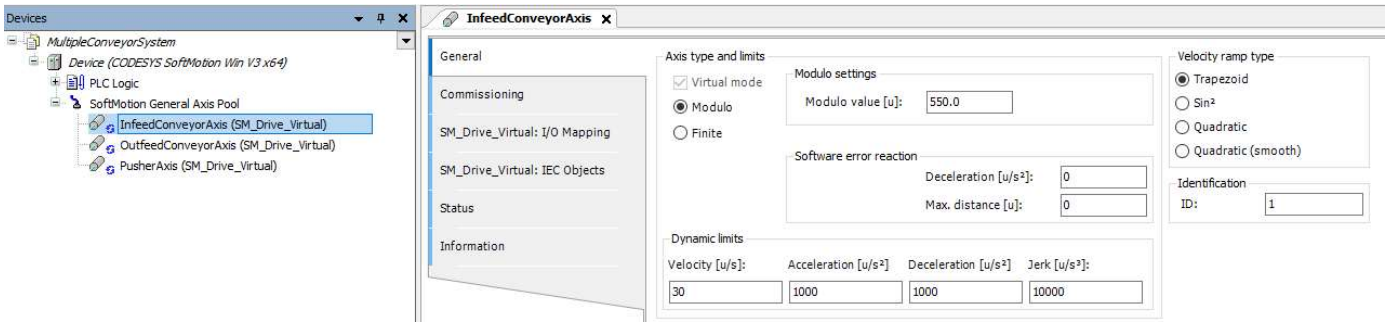
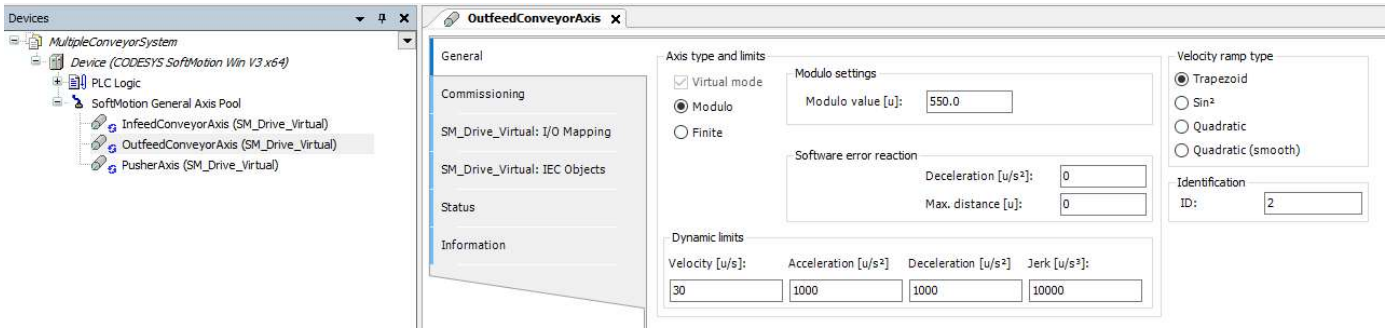
PLCopen – Multiple Conveyors Control

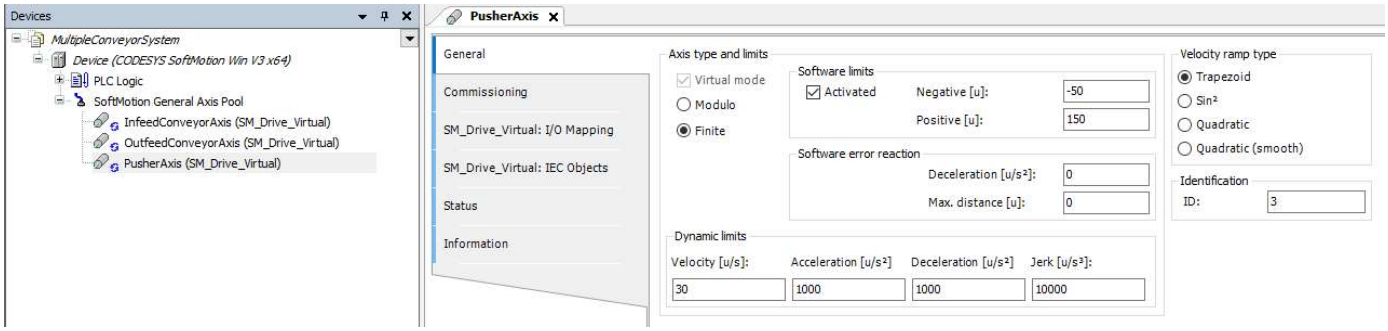
Project in CoDeSys for mulitple conveyors control with PLCopen motion function blocks.



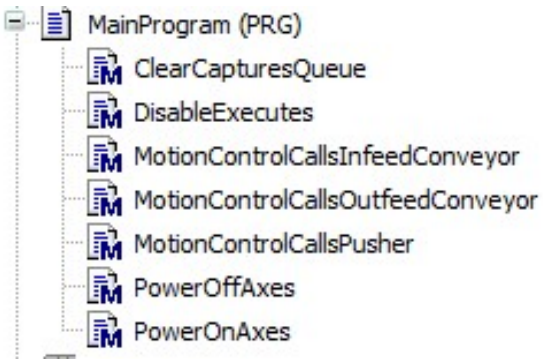









```
VAR_GLOBAL
	// HMI Commands
	CmdEnable                       :	BOOL;
	CmdStart                        :	BOOL;
	CmdStop                         :	BOOL;
	Reset                           :	BOOL;
	Emergency                       :	BOOL := TRUE;
	CmdAlarmLed                     :	BOOL;

	// Status / Feedbacks
	StsEnabled                      :	BOOL;
	StsRunning                      :	BOOL;
	InfeedActualPosition            :	LREAL;
	OutfeedActualPosition           :	LREAL;
	PusherActualPosition            :	LREAL;
	ProductsProcessedInCycle        :	UINT;
	TotalProductsProcessed          :	UINT;

	// Sensors
	OutputSensorOutfeedConveyor             :	BOOL;
	FastSensorInfeedConveyor                :	BOOL;
	FastSensorInfeedConveyorPosition        :	LREAL;
END_VAR

VAR_GLOBAL CONSTANT
	FastSensorPosition              :	LREAL := 300;	// position in [mm]
	PusherPositionOnConveyor        :	LREAL := 450;	// position in [mm]
	PusherStartPosition             :	LREAL := 0;	// position in [mm]
	PusherEndPosition               :	LREAL := 100;	// position in [mm]

	ProductSize                     :	LREAL := 50;	// size in [mm]
	InfeedVelocity                  :	LREAL := 60;	// velocity in [mm/s]
	OutfeedVelocity                 :	LREAL := 60;	// velocity in [mm/s]
	PusherVelocity                  :	LREAL := 100;	// velocity in [mm/s]

	Acceleration                    :	LREAL := 200;	// [mm/s2]
	Deceleration                    :	LREAL := 200;	// [mm/s2]
	EmergencyDeceleration           :	LREAL := 400;	// [mm/s2]
END_VAR
```



```
PROGRAM MainProgram
VAR
	// administrative & diagnostic FBs
	fbPowerInfeedConveyor				:	MC_Power;
	fbPowerOutfeedConveyor				:	MC_Power;
	fbPowerPusher					:	MC_Power;

	fbReadAxisErrorInfeedConveyor			:	MC_ReadAxisError;
	fbReadAxisErrorOutfeedConveyor			:	MC_ReadAxisError;
	fbReadAxisErrorPusher				:	MC_ReadAxisError;

	fbResetInfeedConveyor				:	MC_Reset;
	fbResetOutfeedConveyor				:	MC_Reset;
	fbResetPusher					:	MC_Reset;

	fbReadActualPositionInfeedConveyor		:	MC_ReadActualPosition;
	fbReadActualPositionOutfeedConveyor		:	MC_ReadActualPosition;
	fbReadActualPositionPusher			:	MC_ReadActualPosition;

	// MC Movement FBs
	fbStopInfeedConveyor				:	MC_Stop;
	fbStopOutfeedConveyor				:	MC_Stop;
	fbStopPusher					:	MC_Stop;

	fbMoveVelocityInfeedConveyor			:	MC_MoveVelocity;
	fbMoveRelativeInfeedConveyor			:	MC_MoveRelative;
	fbMoveRelativeOutfeedConveyor			:	MC_MoveRelative;
	fbMoveAbsolutePusher				:	MC_MoveAbsolute;

	SeqState		:	(DISABLED, POWER_ON_AXES,
					AUTO_MODE_WAIT_FOR_START,
					AUTO_MODE_PUSHER_POSITIONING,
					AUTO_MODE_START_MOVE_VELOCITY_INFEED_CONVEYOR,
					AUTO_MODE_WAIT_FOR_SENSOR_CAPTURE,
					AUTO_MODE_START_MOVE_RELATIVE_INFEED_CONVEYOR,
					AUTO_MODE_MOVE_PUSHER_FORWARD,
					AUTO_MODE_MOVE_PUSHER_BACKWARD_OUTFEED_CONVEYOR_FORWARD,
					AUTO_MODE_STOPPING_AXES,
					EMERGENCY_STOPPING_AXES,
					EMERGENCY_DISABLING_AXES,
					EMERGENCY_WAIT_FOR_RESET);

rtStart, rtStop			:	R_TRIG;		// triggers for start/stop buttons
rtFastSensor, rtEndSensor	:	R_TRIG;		// triggers for fastsensor, output sensor

FastSensorCaptures		:	ARRAY[1..MAX_NUMBER_OF_QUEUED_CAPTURES] OF LREAL;
NumberOfCaptures		:	UINT;
Index				:	UINT;
FastSensorDifference		:	LREAL;
END_VAR

VAR CONSTANT
	MAX_NUMBER_OF_QUEUED_CAPTURES		:	UINT := 5;
END_VAR
```

```
// Call motion FBs for all axes
MotionControlCallsInfeedConveyor();
MotionControlCallsOutfeedConveyor();
MotionControlCallsPusher();


// Rising Trigger Calls
rtStart(CLK:= CmdStart, Q=> );
rtStop(CLK:= CmdStop, Q=> );
rtFastSensor(CLK:= FastSensorInfeedConveyor, Q=> );
rtEndSensor(CLK:= OutputSensorOutfeedConveyor, Q=> );

// Adding Sensor Value to Captures Queue
IF rtFastSensor.Q THEN
	NumberOfCaptures := NumberOfCaptures + 1;
	FastSensorCaptures[NumberOfCaptures] := FastSensorInfeedConveyorPosition;
END_IF
```

```
IF rtEndSensor.Q THEN
	ProductsProcessedInCycle := ProductsProcessedInCycle + 1;
	TotalProductsProcessed := TotalProductsProcessed + 1;
END_IF


CASE SeqState OF

	DISABLED:	// Machine Disabled State
		StsEnabled := FALSE;
		StsRunning := FALSE;
		CmdAlarmLed := FALSE;
		IF CmdEnable AND Emergency THEN
			SeqState := POWER_ON_AXES;
		END_IF

	POWER_ON_AXES:	// Power On Axes
		PowerOnAxes();
		IF fbPowerInfeedConveyor.Status AND
			fbPowerOutfeedConveyor.Status AND
			fbPowerPusher.Status THEN

			SeqState := AUTO_MODE_WAIT_FOR_START;
			StsEnabled := TRUE;
		END_IF


	AUTO_MODE_WAIT_FOR_START:  // Auto Mode: Wait For Start
		IF rtStart.Q THEN
			ProductsProcessedInCycle := 0;
			SeqState := AUTO_MODE_PUSHER_POSITIONING;
			StsRunning := TRUE;
		END_IF


	AUTO_MODE_PUSHER_POSITIONING:	// Auto Mode: Pusher Repositioning
		fbMoveAbsolutePusher.Position := PusherStartPosition;
		fbMoveAbsolutePusher.Execute := TRUE;

		IF fbMoveAbsolutePusher.Done THEN
			fbMoveAbsolutePusher.Execute := FALSE;
			SeqState := AUTO_MODE_START_MOVE_VELOCITY_INFEED_CONVEYOR;
		END_IF


	AUTO_MODE_START_MOVE_VELOCITY_INFEED_CONVEYOR:
	// Auto Mode: Start MoveVelocity Infeed Conveyor
		fbMoveVelocityInfeedConveyor.Execute := TRUE;

		IF fbMoveVelocityInfeedConveyor.InVelocity THEN
			fbMoveVelocityInfeedConveyor.Execute := FALSE;
			SeqState := AUTO_MODE_WAIT_FOR_SENSOR_CAPTURE;
		END_IF


	AUTO_MODE_WAIT_FOR_SENSOR_CAPTURE:	// Auto Mode: Wait for Sensor Capture
		IF NumberOfCaptures > 0 THEN
			SeqState := AUTO_MODE_START_MOVE_RELATIVE_INFEED_CONVEYOR;
		END_IF


	AUTO_MODE_START_MOVE_RELATIVE_INFEED_CONVEYOR:
	// Auto Mode: Start MoveRelative Infeed Conveyor

		FastSensorDifference := InfeedActualPosition - FastSensorCaptures[1];
		// Manage rollover
		IF FastSensorDifference < 0 THEN
			FastSensorDifference := FastSensorDifference + 550;
		END_IF;

		fbMoveRelativeInfeedConveyor.Execute := TRUE;
		fbMoveRelativeInfeedConveyor.Distance := PusherPositionOnConveyor -
		FastSensorPosition - FastSensorDifference;
```

```
	IF fbMoveRelativeInfeedConveyor.Done THEN
		fbMoveRelativeInfeedConveyor.Execute := FALSE;

		// Remove capture from the list
		NumberOfCaptures := NumberOfCaptures - 1;

		FOR Index := 1 TO NumberOfCaptures DO
			FastSensorCaptures[Index] := FastSensorCaptures[Index + 1];
		END_FOR
		FastSensorCaptures[NumberOfCaptures + 1] := 0;

		SeqState := AUTO_MODE_MOVE_PUSHER_FORWARD;
	END_IF


AUTO_MODE_MOVE_PUSHER_FORWARD:    // Auto Mode: Move Pusher Forward
	fbMoveAbsolutePusher.Position := PusherEndPosition;
	fbMoveAbsolutePusher.Execute := TRUE;

	IF fbMoveAbsolutePusher.Done THEN
		fbMoveAbsolutePusher.Execute := FALSE;
		SeqState := AUTO_MODE_MOVE_PUSHER_BACKWARD_OUTFEED_CONVEYOR_FORWARD;
	END_IF


AUTO_MODE_MOVE_PUSHER_BACKWARD_OUTFEED_CONVEYOR_FORWARD:
// Auto Mode: Move Pusher Backward and OutfeedConveyor Forward

	fbMoveAbsolutePusher.Position := PusherStartPosition;
	fbMoveAbsolutePusher.Execute := TRUE;

	fbMoveRelativeOutfeedConveyor.Distance := ProductSize + 5;
	fbMoveRelativeOutfeedConveyor.Execute := TRUE;

	IF fbMoveAbsolutePusher.Done AND fbMoveRelativeOutfeedConveyor.Done THEN
		fbMoveAbsolutePusher.Execute := FALSE;
		fbMoveRelativeOutfeedConveyor.Execute := FALSE;
		SeqState := AUTO_MODE_START_MOVE_VELOCITY_INFEED_CONVEYOR;
	END_IF


AUTO_MODE_STOPPING_AXES:   // Auto: Stopping the Axes
	IF fbStopInfeedConveyor.Done AND
		fbStopOutfeedConveyor.Done AND
		fbStopPusher.Done THEN

		fbStopInfeedConveyor.Execute := FALSE;
		fbStopOutfeedConveyor.Execute := FALSE;
		fbStopPusher.Execute := FALSE;
		SeqState := AUTO_MODE_WAIT_FOR_START;
	END_IF


EMERGENCY_STOPPING_AXES:   // Emergency: Stopping the Axes
	IF fbStopInfeedConveyor.Done AND
		fbStopOutfeedConveyor.Done AND
		fbStopPusher.Done THEN

		fbStopInfeedConveyor.Execute := FALSE;
		fbStopOutfeedConveyor.Execute := FALSE;
		fbStopPusher.Execute := FALSE;
		SeqState := EMERGENCY_DISABLING_AXES;
	END_IF


EMERGENCY_DISABLING_AXES:  // Emergency: Disabling the Axes
	PowerOffAxes();
	IF NOT fbPowerInfeedConveyor.Status
		AND NOT fbPowerOutfeedConveyor.Status
		AND NOT fbPowerPusher.Status THEN

		SeqState := EMERGENCY_WAIT_FOR_RESET;
		StsEnabled := FALSE;
	END_IF
```

```
        EMERGENCY_WAIT_FOR_RESET:  // Emergency: Wait for Reset
                IF Reset THEN
                        SeqState := DISABLED;
                END_IF

END_CASE

// Stop Transition
IF rtStop.Q AND SeqState > AUTO_MODE_WAIT_FOR_START AND SeqState < EMERGENCY_STOPPING_AXES THEN
        DisableExecutes();
        ClearCapturesQueue();
        fbStopInfeedConveyor.Execute := TRUE;
        fbStopOutfeedConveyor.Execute := TRUE;
        fbStopPusher.Execute := TRUE;
        fbStopInfeedConveyor.Deceleration := Deceleration;
        fbStopOutfeedConveyor.Deceleration := Deceleration;
        fbStopPusher.Deceleration := Deceleration;
        StsRunning := FALSE;
        SeqState := AUTO_MODE_STOPPING_AXES;
END_IF

// Emergency Transition
IF (NOT Emergency OR
        fbReadAxisErrorInfeedConveyor.Error OR
        fbReadAxisErrorOutfeedConveyor.Error OR
        fbReadAxisErrorPusher.Error OR
        fbReadAxisErrorPusher.SWEndSwitchActive) AND
        SeqState > DISABLED AND SeqState < EMERGENCY_STOPPING_AXES THEN

        DisableExecutes();
        ClearCapturesQueue();
        CmdAlarmLed := TRUE;
        fbStopInfeedConveyor.Execute := TRUE;
        fbStopOutfeedConveyor.Execute := TRUE;
        fbStopPusher.Execute := TRUE;
        fbStopInfeedConveyor.Deceleration := EmergencyDeceleration;
        fbStopOutfeedConveyor.Deceleration := EmergencyDeceleration;
        fbStopPusher.Deceleration := EmergencyDeceleration;
        StsRunning := FALSE;
        SeqState := EMERGENCY_STOPPING_AXES;
END_IF

// Disable Transition
IF NOT CmdEnable THEN
        DisableExecutes();
        ClearCapturesQueue();
        PowerOffAxes();
        SeqState := DISABLED;
END_IF
```

```
METHOD ClearCapturesQueue
VAR_INPUT
END_VAR

FOR Index := 1 TO NumberOfCaptures DO
        FastSensorCaptures[Index] := 0.0;
END_FOR
NumberOfCaptures := 0;
```

```
METHOD DisableExecutes
VAR_INPUT
END_VAR

fbMoveVelocityInfeedConveyor.Execute := FALSE;
fbMoveRelativeInfeedConveyor.Execute := FALSE;
fbStopInfeedConveyor.Execute := FALSE;
fbMoveRelativeOutfeedConveyor.Execute := FALSE;
fbStopOutfeedConveyor.Execute := FALSE;
fbMoveAbsolutePusher.Execute := FALSE;
fbStopPusher.Execute := FALSE;
```

```
METHOD MotionControlCallsInfeedConveyor
VAR_INPUT
END_VAR

// Infeed Conveyor FBs

fbPowerInfeedConveyor(
       Axis:= InfeedConveyorAxis,
       Enable:= TRUE,
       bRegulatorOn:= ,
       bDriveStart:= ,
       Status=> ,
       bRegulatorRealState=> ,
       bDriveStartRealState=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> );

fbReadAxisErrorInfeedConveyor(
       Axis:= InfeedConveyorAxis,
       Enable:= TRUE,
       Valid=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> ,
       AxisError=> ,
       AxisErrorID=> ,
       SWEndSwitchActive=> );

fbResetInfeedConveyor(
       Axis:= InfeedConveyorAxis,
       Execute:= Reset,
       Done=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> );

fbReadActualPositionInfeedConveyor(
       Axis:= InfeedConveyorAxis,
       Enable:= TRUE,
       Valid=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> ,
       Position=> InfeedActualPosition);

fbStopInfeedConveyor(
       Axis:= InfeedConveyorAxis,
       Execute:= ,
       Deceleration:= ,
       Jerk:= ,
       Done=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> );

fbMoveVelocityInfeedConveyor(
       Axis:= InfeedConveyorAxis,
       Execute:= ,
       Velocity:= InfeedVelocity,
       Acceleration:= Acceleration,
       Deceleration:= Deceleration,
       Jerk:= ,
       Direction:= ,
       BufferMode:= ,
       InVelocity=> ,
       Busy=> ,
       Active=> ,
       CommandAborted=> ,
       Error=> ,
       ErrorID=> );
```

```
fbMoveRelativeInfeedConveyor(
       Axis:= InfeedConveyorAxis,
       Execute:= ,
       Distance:= ,
       Velocity:= InfeedVelocity,
       Acceleration:= Acceleration,
       Deceleration:= Deceleration ,
       Jerk:= ,
       BufferMode:= ,
       Done=> ,
       Busy=> ,
       Active=> ,
       CommandAborted=> ,
       Error=> ,
       ErrorID=> );
```

```
METHOD MotionControlCallsOutfeedConveyor
VAR_INPUT
END_VAR

// Outfeed Conveyor FBs

fbPowerOutfeedConveyor(
       Axis:= OutfeedConveyorAxis,
       Enable:= TRUE,
       bRegulatorOn:= ,
       bDriveStart:= ,
       Status=> ,
       bRegulatorRealState=> ,
       bDriveStartRealState=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> );

fbReadAxisErrorOutfeedConveyor(
       Axis:= OutfeedConveyorAxis,
       Enable:= TRUE,
       Valid=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> ,
       AxisError=> ,
       AxisErrorID=> ,
       SWEndSwitchActive=> );

fbResetOutfeedConveyor(
       Axis:= OutfeedConveyorAxis,
       Execute:= Reset,
       Done=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> );

fbReadActualPositionOutfeedConveyor(
       Axis:= OutfeedConveyorAxis,
       Enable:= TRUE,
       Valid=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> ,
       Position=> OutfeedActualPosition);

fbStopOutfeedConveyor(
       Axis:= OutfeedConveyorAxis,
       Execute:= ,
       Deceleration:= ,
       Jerk:= ,
       Done=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> );
```

```
fbMoveRelativeOutfeedConveyor(
       Axis:= OutfeedConveyorAxis,
       Execute:= ,
       Distance:= ,
       Velocity:= OutfeedVelocity,
       Acceleration:= Acceleration,
       Deceleration:= Deceleration ,
       Jerk:= ,
       BufferMode:= ,
       Done=> ,
       Busy=> ,
       Active=> ,
       CommandAborted=> ,
       Error=> ,
       ErrorID=> );
```

```
METHOD MotionControlCallsPusher
VAR_INPUT
END_VAR

// Pusher Conveyor FBs

fbPowerPusher(
       Axis:= PusherAxis,
       Enable:= TRUE,
       bRegulatorOn:= ,
       bDriveStart:= ,
       Status=> ,
       bRegulatorRealState=> ,
       bDriveStartRealState=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> );

fbReadAxisErrorPusher(
       Axis:= PusherAxis,
       Enable:= TRUE,
       Valid=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> ,
       AxisError=> ,
       AxisErrorID=> ,
       SWEndSwitchActive=> );

fbResetPusher(
       Axis:= PusherAxis,
       Execute:= Reset,
       Done=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> );

fbReadActualPositionPusher(
       Axis:= PusherAxis,
       Enable:= TRUE,
       Valid=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> ,
       Position=> PusherActualPosition);

fbStopPusher(
       Axis:= PusherAxis,
       Execute:= ,
       Deceleration:= ,
       Jerk:= ,
       Done=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> );
```

```
fbMoveAbsolutePusher(
       Axis:= PusherAxis,
       Execute:= ,
       Position:= ,
       Velocity:= PusherVelocity,
       Acceleration:= Acceleration,
       Deceleration:= Deceleration,
       Jerk:= ,
       Direction:= ,
       BufferMode:= ,
       Done=> ,
       Busy=> ,
       Active=> ,
       CommandAborted=> ,
       Error=> ,
       ErrorID=> );
```

```
METHOD PowerOffAxes
VAR_INPUT
END_VAR

fbPowerInfeedConveyor.bDriveStart := FALSE;
fbPowerOutfeedConveyor.bDriveStart := FALSE;
fbPowerPusher.bDriveStart := FALSE;

fbPowerInfeedConveyor.bRegulatorOn := FALSE;
fbPowerOutfeedConveyor.bRegulatorOn := FALSE;
fbPowerPusher.bRegulatorOn := FALSE;
```

```
METHOD PowerOnAxes
VAR_INPUT
END_VAR

fbPowerInfeedConveyor.bDriveStart := TRUE;
fbPowerOutfeedConveyor.bDriveStart := TRUE;
fbPowerPusher.bDriveStart := TRUE;

fbPowerInfeedConveyor.bRegulatorOn := TRUE;
fbPowerOutfeedConveyor.bRegulatorOn := TRUE;
fbPowerPusher.bRegulatorOn := TRUE;
```