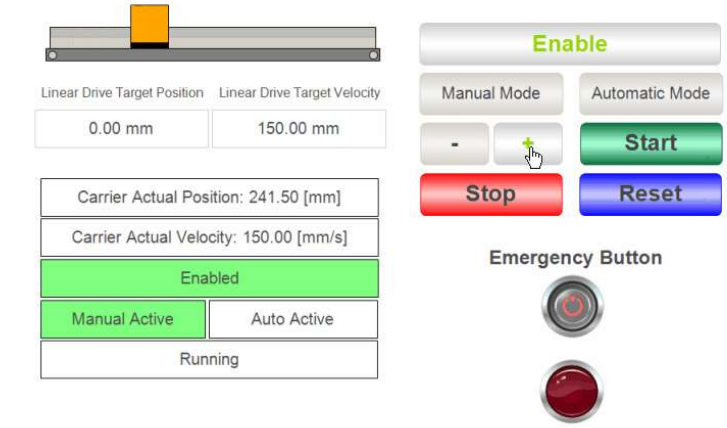
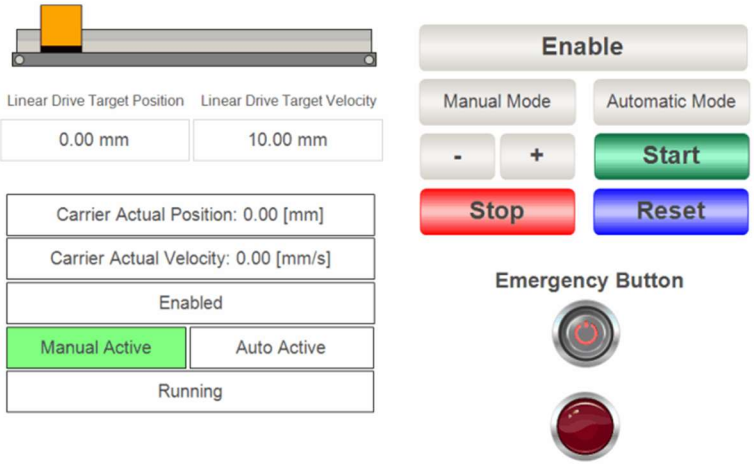
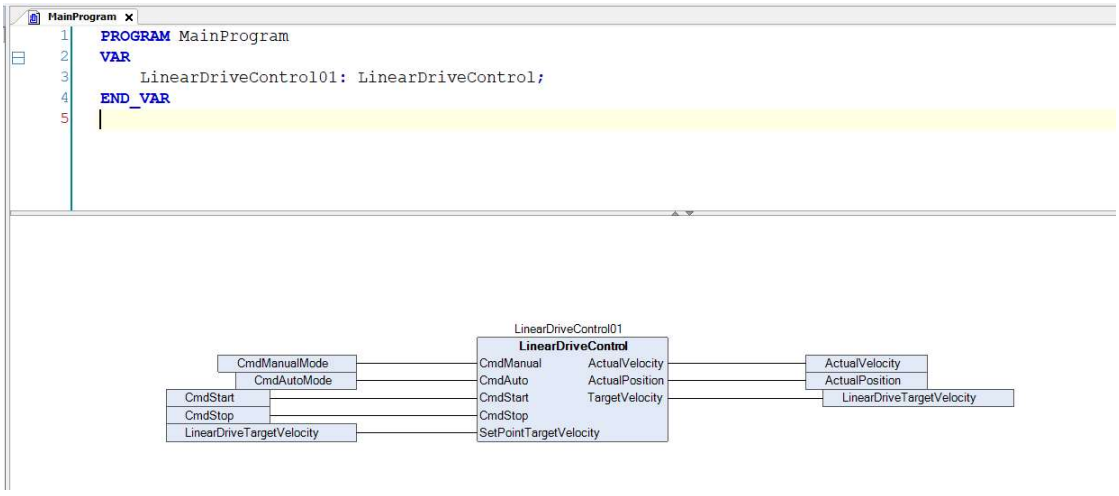
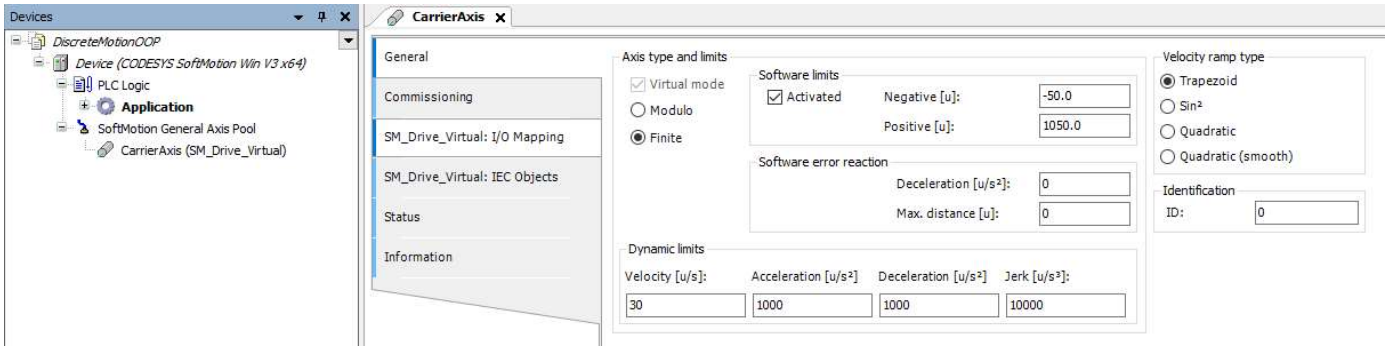
PLCopen – Linear Drive Control / Discret Motion

Simple project in CoDeSys for linear drive control with PLCopen motion function blocks.







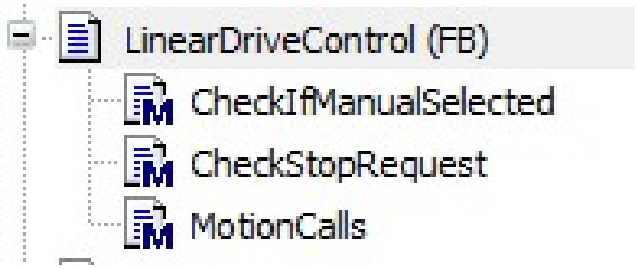



```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE ET_MachineMode :
(
	Manual := 0,
	Auto := 1
);
END_TYPE
```

```
VAR_GLOBAL
	// HMI Commands
	CmdEnable                       :	BOOL;
	CmdManualMode                   :	BOOL;
	CmdAutoMode                     :	BOOL;
	CmdJogFwd                       :	BOOL;
	CmdJogBwd                       :	BOOL;
	CmdStart                        :	BOOL;
	CmdStop                         :	BOOL;
	CmdReset                        :	BOOL;
	CmdEmergencyButton              :	BOOL := TRUE;
	LinearDriveTargetPosition       :	LREAL;
	LinearDriveTargetVelocity       :	LREAL;
	CmdAlarmLed                     :	BOOL;

	// HMI Leds - Feedbacks
	StsEnabled                      :	BOOL;
	StsMachineMode                  :	ET_MachineMode;
	StsRunning                      :	BOOL;
	ActualPosition                  :	LREAL;
	ActualVelocity                  :	LREAL;

	// Global Parameters
	MinPosition                     :	LREAL   := 0.0;
	MaxPosition                     :	LREAL   := 1000.0;
	MinVelocity                     :	LREAL   := 10.0;
	MaxVelocity                     :	LREAL   := 200.0;
	Acceleration                    :	LREAL   := 300.0;
	Deceleration                    :	LREAL   := 300.0;
	EmergencyDeceleration           :	LREAL   := 500.0;
END_VAR
```



```
FUNCTION_BLOCK LinearDriveControl
VAR_INPUT
	CmdManual			:	BOOL;
	CmdAuto				:	BOOL;
	CmdStart			:	BOOL;
	CmdStop				:	BOOL;
	SetPointTargetVelocity	:	LREAL;
END_VAR
VAR_OUTPUT
	ActualVelocity			:	LREAL;
	ActualPosition			:	LREAL;
	TargetVelocity			:	LREAL;
END_VAR
VAR
	// state machine
	SeqState			:	(DISABLED,
						WAIT_FOR_POWER_ON,
						MANUAL_MODE,
						AUTO_MODE_WAIT_FOR_START,
						AUTO_MODE_MOVE_TO_START_POSITION,
						AUTO_MODE_MOVE_TO_TARGET_POSITION,
						AUTO_MODE_STOPPING,
						SELECT_AUTO_MANUAL,
						EMERGENCY_STOPPING,
						EMERGENCY_DISABLING,
						EMERGENCY_WAIT_FOR_RESET,
						EMERGENCY_RESET_DONE);

	rtStart				:	R_TRIG;

	fbPowerCarrier			:	MC_Power;
	fbMoveAbsoluteLinearDrive	:	MC_MoveAbsolute;
	fbMoveRelativeLinearDrive	:	MC_MoveRelative;
	fbStopLinearDrive		:	MC_Stop;
	fbJogLinearDrive		:	MC_Jog;
	fbReadActualPosition		:	MC_ReadActualPosition;
	fbReadActualVelocity		:	MC_ReadActualVelocity;
	fbReadAxisError			:	MC_ReadAxisError;
	fbReset				:	MC_Reset;
END_VAR
```

```
MotionCalls();

// Handle Manual - Auto Buttons
IF CmdManual THEN StsMachineMode := ET_MachineMode.Manual; END_IF
IF CmdAuto THEN StsMachineMode := ET_MachineMode.Auto; END_IF

// Handle Target Velocity Limitation
TargetVelocity := LIMIT(MinVelocity,SetPointTargetVelocity,MaxVelocity);

// Rising Trigger Start
rtStart(CLK:= CmdStart, Q=> );

CASE SeqState OF


	DISABLED:	// Machine Disabled State
		StsEnabled := FALSE;
		StsRunning := FALSE;
		CmdAlarmLed := FALSE;
		IF CmdEnable AND CmdEmergencyButton THEN
			fbPowerCarrier.Enable := TRUE;
			fbPowerCarrier.bDriveStart := TRUE;
			fbPowerCarrier.bRegulatorOn := TRUE;
			SeqState := WAIT_FOR_POWER_ON;
		END_IF
```

```
WAIT_FOR_POWER_ON:  // Wait for Power On
	IF fbPowerCarrier.Status THEN
		StsEnabled := TRUE;
		IF StsMachineMode = ET_MachineMode.Manual THEN
			SeqState := MANUAL_MODE;
		ELSIF StsMachineMode = ET_MachineMode.Auto THEN
			SeqState := AUTO_MODE_WAIT_FOR_START;
		END_IF
	END_IF


MANUAL_MODE:  // Manual Active
	fbJogLinearDrive.JogForward := CmdJogFwd;
	fbJogLinearDrive.JogBackward := CmdJogBwd;
	fbJogLinearDrive.Velocity := TargetVelocity;

	// Transition to Auto Mode
	IF StsMachineMode = ET_MachineMode.Auto THEN
		fbJogLinearDrive.JogForward := FALSE;
		fbJogLinearDrive.JogBackward := FALSE;
		fbStopLinearDrive.Execute := TRUE;
		fbStopLinearDrive.Deceleration := Deceleration;
		SeqState := SELECT_AUTO_MANUAL;
	END_IF


AUTO_MODE_WAIT_FOR_START:  // Auto Mode: wait for start
	// wait for start request
	IF rtStart.Q THEN
		SeqState := AUTO_MODE_MOVE_TO_START_POSITION;
		StsRunning := TRUE;
	END_IF

	// transition to manual mode
	IF StsMachineMode = ET_MachineMode.Manual THEN
		fbStopLinearDrive.Deceleration := Deceleration;
		fbStopLinearDrive.Execute := TRUE;
		SeqState := SELECT_AUTO_MANUAL;
	END_IF

AUTO_MODE_MOVE_TO_START_POSITION:// auto mode active - move to position 0
	fbMoveAbsoluteLinearDrive.Execute := TRUE;
	fbMoveAbsoluteLinearDrive.Position := 0;
	fbMoveAbsoluteLinearDrive.Velocity := TargetVelocity;

	// Position reached
	IF fbMoveAbsoluteLinearDrive.Done THEN
		fbMoveAbsoluteLinearDrive.Execute := FALSE;
		SeqState := AUTO_MODE_MOVE_TO_TARGET_POSITION;
	END_IF

	// check stop request
	CheckStopRequest();

	// transition to manual mode
	CheckIfManualSelected();


AUTO_MODE_MOVE_TO_TARGET_POSITION:	// Auto Mode: move to target position

	fbMoveAbsoluteLinearDrive.Execute := TRUE;
	fbMoveAbsoluteLinearDrive.Position := LinearDriveTargetPosition;
	fbMoveAbsoluteLinearDrive.Velocity := TargetVelocity;


	// Position reached
	IF fbMoveAbsoluteLinearDrive.Done THEN
		fbMoveAbsoluteLinearDrive.Execute := FALSE;
		SeqState := AUTO_MODE_MOVE_TO_START_POSITION;
	END_IF

	// check stop request
	CheckStopRequest();
```

```
			// transition to manual mode
			CheckIfManualSelected();


		AUTO_MODE_STOPPING: // auto mode: stopping
			IF fbStopLinearDrive.Done AND NOT CmdStop THEN
				fbStopLinearDrive.Execute := FALSE;
				SeqState := AUTO_MODE_WAIT_FOR_START;
			END_IF


		SELECT_AUTO_MANUAL: // Select Auto/Manual
			IF fbStopLinearDrive.Done THEN
				fbStopLinearDrive.Execute := FALSE;
				IF StsMachineMode = ET_MachineMode.Auto THEN
					SeqState := AUTO_MODE_WAIT_FOR_START;
				ELSIF StsMachineMode = ET_MachineMode.Manual THEN
					SeqState := MANUAL_MODE;
				END_IF
			END_IF


		EMERGENCY_STOPPING: // Emergency - Stopping
			CmdAlarmLed := TRUE;
			IF fbStopLinearDrive.Done THEN
				fbStopLinearDrive.Execute := FALSE;
				SeqState := EMERGENCY_DISABLING;
			END_IF


		EMERGENCY_DISABLING:  // Emergency - Disabling
			fbPowerCarrier.bDriveStart := FALSE;
			fbPowerCarrier.bRegulatorOn := FALSE;
			IF NOT fbPowerCarrier.Status THEN
				SeqState := EMERGENCY_WAIT_FOR_RESET;
			END_IF


		EMERGENCY_WAIT_FOR_RESET:  // Emergency - Wait for Reset
			IF CmdReset THEN
				IF fbReadAxisError.Error OR fbReadAxisError.SWEndSwitchActive THEN
					fbReset.Execute := TRUE;
					SeqState := EMERGENCY_RESET_DONE;
				ELSE
					SeqState := DISABLED;
				END_IF
			END_IF


		EMERGENCY_RESET_DONE:	// Emergency: Wait for Reset Done
			IF fbReset.Done THEN
				fbReset.Execute := FALSE;
				SeqState := DISABLED;
			END_IF

END_CASE

// Disable Transition
IF NOT CmdEnable THEN
	fbPowerCarrier.bDriveStart := FALSE;
	fbPowerCarrier.bRegulatorOn := FALSE;
	fbMoveAbsoluteLinearDrive.Execute := FALSE;
	fbMoveRelativeLinearDrive.Execute := FALSE;
	fbJogLinearDrive.JogBackward := FALSE;
	fbJogLinearDrive.JogForward := FALSE;
	SeqState := DISABLED;
END_IF

// Emergency - Error Transition
IF (NOT CmdEmergencyButton OR fbReadAxisError.Error OR fbReadAxisError.SWEndSwitchActive) AND
	SeqState > DISABLED AND SeqState < EMERGENCY_STOPPING THEN

	fbStopLinearDrive.Deceleration := EmergencyDeceleration;
	fbStopLinearDrive.Execute := TRUE;
```

```
        fbMoveAbsoluteLinearDrive.Execute := FALSE;
        fbMoveRelativeLinearDrive.Execute := FALSE;
        fbJogLinearDrive.JogBackward := FALSE;
        fbJogLinearDrive.JogForward := FALSE;
        StsRunning := FALSE;
        StsEnabled := FALSE;
        SeqState := EMERGENCY_STOPPING;
END_IF

// Update Actual Position and Actual Velocity
IF fbReadActualPosition.Valid THEN
        ActualPosition := fbReadActualPosition.Position;
END_IF

IF fbReadActualVelocity.Valid THEN
        ActualVelocity := fbReadActualVelocity.Velocity;
END_IF
```

```
METHOD CheckIfManualSelected
VAR_INPUT
END_VAR

// This method checks if manual mode was selcted and if yes sets respectively commands

IF StsMachineMode = ET_MachineMode.Manual THEN
        fbMoveAbsoluteLinearDrive.Execute := FALSE;
        fbStopLinearDrive.Execute := TRUE;
        fbStopLinearDrive.Deceleration := Deceleration;
        SeqState := SELECT_AUTO_MANUAL;
END_IF
```

```
METHOD CheckStopRequest
VAR_INPUT
END_VAR

// check stop request

IF CmdStop THEN
        fbMoveAbsoluteLinearDrive.Execute := FALSE;
        StsRunning := FALSE;
        fbStopLinearDrive.Execute := TRUE;
        fbStopLinearDrive.Deceleration := Deceleration;
        SeqState := AUTO_MODE_STOPPING;
END_IF
```

```
METHOD MotionCalls
VAR_INPUT
END_VAR

fbPowerCarrier(
        Axis:= CarrierAxis,
        Enable:= ,
        bRegulatorOn:= ,
        bDriveStart:= ,
        Status=> ,
        bRegulatorRealState=> ,
        bDriveStartRealState=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> );

fbMoveAbsoluteLinearDrive(
        Axis:= CarrierAxis,
        Execute:= ,
        Position:= ,
        Velocity:= ,
        Acceleration:= Acceleration,
        Deceleration:= Deceleration,
        Jerk:= ,
        Direction:= ,
        BufferMode:= ,
```

```
        Done=> ,
        Busy=> ,
        Active=> ,
        CommandAborted=> ,
        Error=> ,
        ErrorID=> );

fbMoveRelativeLinearDrive(
        Axis:= CarrierAxis,
        Execute:= ,
        Distance:= ,
        Velocity:= ,
        Acceleration:= Acceleration,
        Deceleration:= Deceleration,
        Jerk:= ,
        BufferMode:= ,
        Done=> ,
        Busy=> ,
        Active=> ,
        CommandAborted=> ,
        Error=> ,
        ErrorID=> );

fbJogLinearDrive(
        Axis:= CarrierAxis,
        JogForward:= ,
        JogBackward:= ,
        Velocity:= ,
        Acceleration:= Acceleration,
        Deceleration:= Deceleration,
        Jerk:= ,
        Busy=> ,
        CommandAborted=> ,
        Error=> ,
        ErrorId=> );

fbStopLinearDrive(
        Axis:= CarrierAxis,
        Execute:= ,
        Deceleration:= ,
        Jerk:= ,
        Done=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> );

fbReadActualPosition(
        Axis:= CarrierAxis,
        Enable:= TRUE,
        Valid=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> ,
        Position=> );

fbReadActualVelocity(
        Axis:= CarrierAxis,
        Enable:= TRUE,
        Valid=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> ,
        Velocity=> );

fbReset(
        Axis:= CarrierAxis,
        Execute:= ,
        Done=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> );
```

```
fbReadAxisError(
       Axis:= CarrierAxis,
       Enable:= ,
       Valid=> ,
       Busy=> ,
       Error=> ,
       ErrorID=> ,
       AxisError=> ,
       AxisErrorID=> ,
       SWEndSwitchActive=> );
```