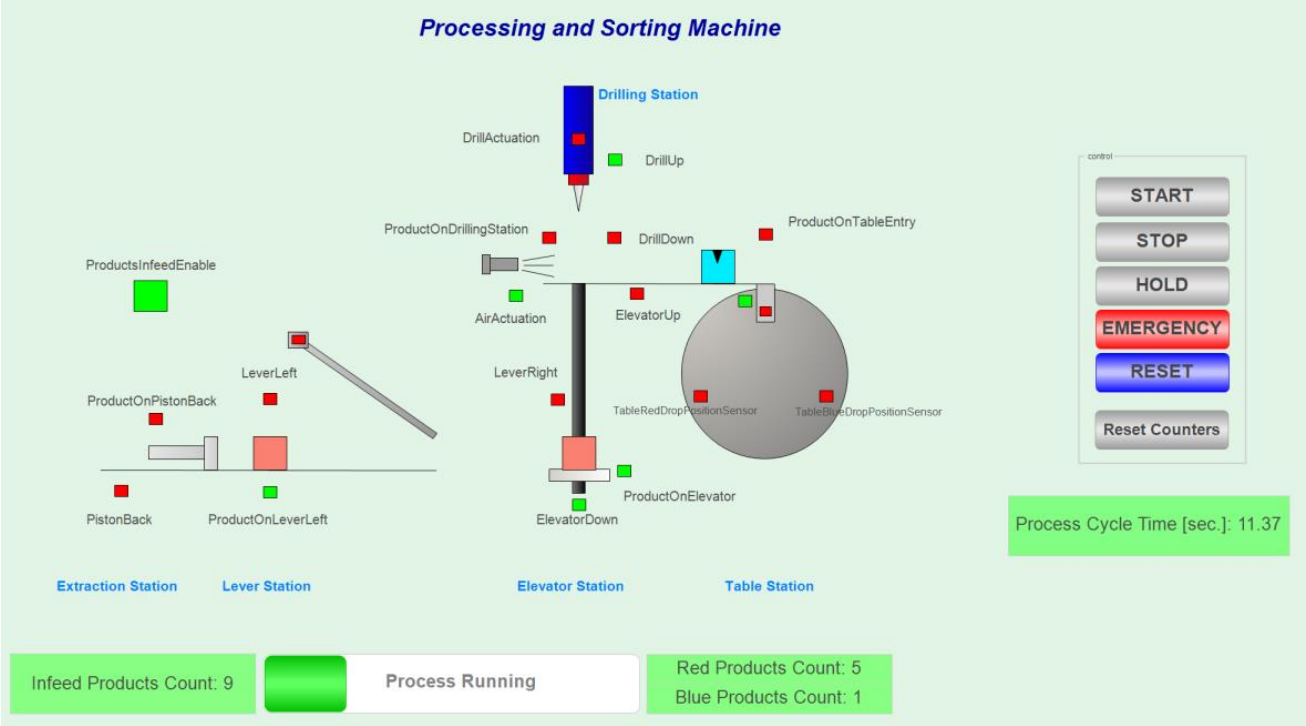
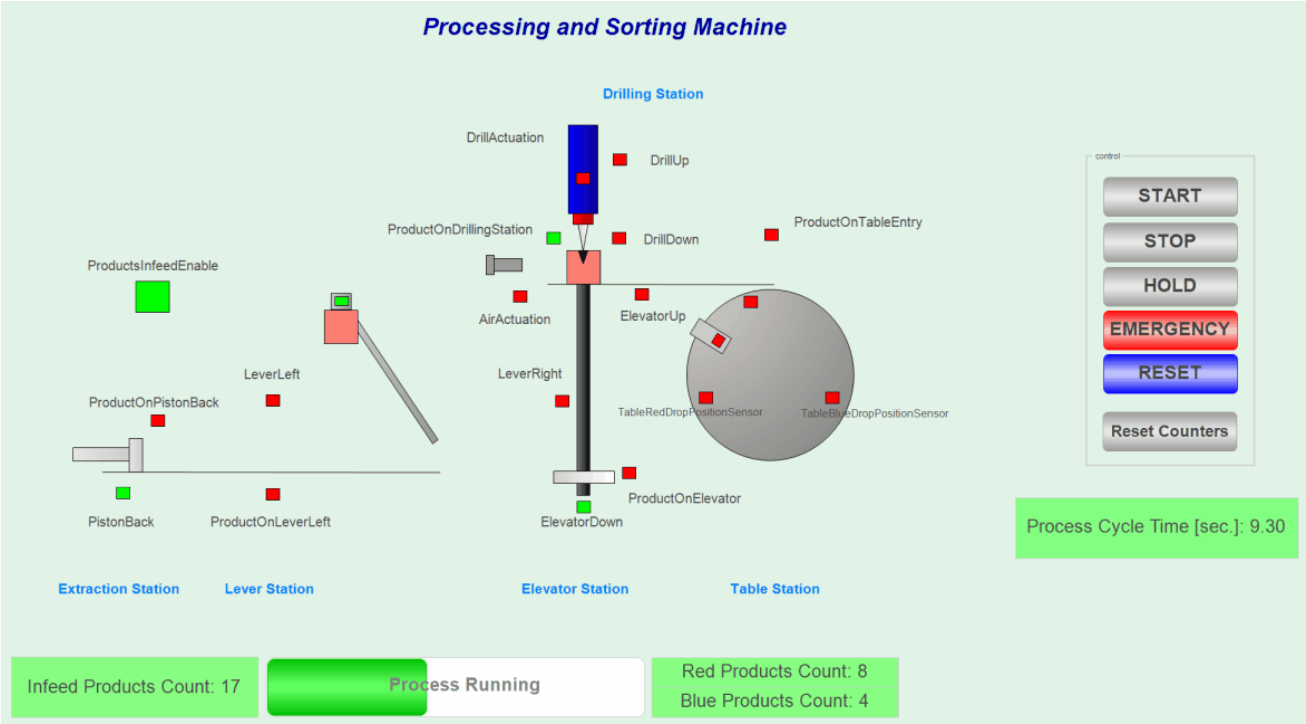
Processing and Sorting Machine

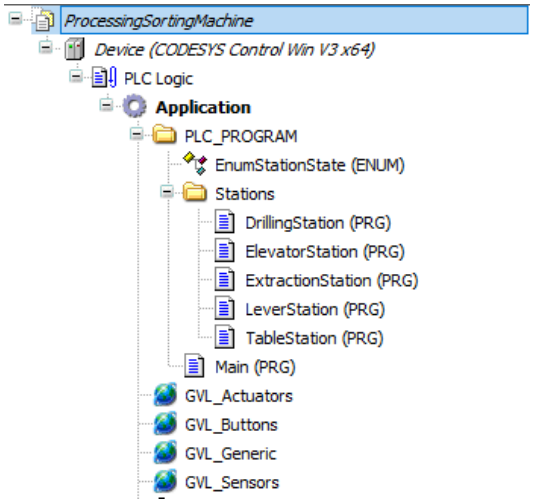
Sequential machine control with multiple stations working independently.

Product is transferred through many stations and product type detecion at the last station decides about final drop location.

Production statistics are implemented.








```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE EnumStationState :
(
	NotRunning		:= 0,
	Running			:= 10,
	Stopping		:= 20,
	Holding			:= 30,
	Fault			:= 40
);
END_TYPE
```

```
PROGRAM ExtractionStation
VAR_INPUT
	Start			: BOOL;
	Stop			: BOOL;
	Hold			: BOOL;
	Emergency		: BOOL;
	ResetCounter	: BOOL;
END_VAR

VAR
	State			: UINT;
	HoldState		: UINT;
	HoldRequest		: BOOL;
END_VAR

VAR_OUTPUT
	StationState		: EnumStationState;
	ProductInfeedCount	: UINT;		// number of products supplied at the infeed
END_VAR

CASE State OF

	// disabled
	0:
		ProductsInfeedEnable := FALSE;
		IF Start AND NOT Stop AND NOT Emergency AND NOT Hold THEN
			State := 5;
		END_IF
```

```
        // move piston back
        5:
                MovePistonForward := FALSE;
                IF PistonStartPositionSensor THEN
                        State := 10;
                END_IF

        // wait for product
        10:
                MovePistonForward := FALSE;
                ProductsInfeedEnable := TRUE;
                IF ProductOnPistonStartSensor AND NOT ProductForLeverPickup THEN
                        State := 20;

                        // start of cycle time calculation
                        IF NOT CalculatingCycleTime THEN
                                CalculatingCycleTime := TRUE;
                                StartCycleTime := DateTimeProvider.GetDateTime();
                        END_IF
                END_IF

        // move product forward
        20:
                MovePistonForward := TRUE;
                IF ProductForLeverPickup THEN
                        State := 5;
                        // count supplied products
                        ProductInfeedCount := ProductInfeedCount + 1;
                END_IF


        // hold button released
        500:
                IF NOT Hold THEN
                        State := HoldState;
                        HoldState := 0;
                END_IF

        // stopping state
        1000:
                IF (MovePistonForward AND ProductForLeverPickup) OR (NOT MovePistonForward
                    AND PistonStartPositionSensor) THEN
                        State := 0;
                END_IF


END_CASE

// e-stop pressed
IF State <> 0 AND Emergency THEN
        State := 0;
        MovePistonForward := FALSE;
END_IF
```

```
// stop button pressed
IF State <> 0 AND State <> 1000 AND Stop THEN
	State := 1000;
END_IF

// hold button pressed
IF State<>0 AND State <>500 AND State<>1000 AND Hold THEN
	HoldRequest := TRUE;
END_IF

IF State <> 500 AND HoldRequest THEN
		HoldRequest := FALSE;
		HoldState := State;
		State := 500;
END_IF

IF ResetCounter THEN
	ProductInfeedCount := 0;
END_IF

// ==============  station state management =============
IF	State=0 THEN
				StationState := EnumStationState.NotRunning;
ELSIF State=500 THEN
					StationState := EnumStationState.Holding;
ELSIF State=1000 THEN
					StationState := EnumStationState.Stopping;
ELSE
	StationState := EnumStationState.Running;
END_IF
```

```
PROGRAM LeverStation

VAR_INPUT
	Start		: BOOL;
	Stop		: BOOL;
	Hold		: BOOL;
	Emergency	: BOOL;
END_VAR

VAR
	State		: UINT;
	HoldState	: UINT;
	HoldRequest	: BOOL;
END_VAR

VAR_OUTPUT
	StationState : EnumStationState;
END_VAR
```

```
CASE State OF
	// disabled
	0:
		ProductsInfeedEnable := FALSE;
		IF Start AND NOT Stop AND NOT Emergency AND NOT Hold THEN
			State := 10;
		END_IF

	// move lever left
	10:
		MoveLeverRight	:= FALSE;
		LeverGripper	:= FALSE;
		IF LeverLeftPositionSensor THEN
			State := 20;
		END_IF

	// wait for product
	20:
		IF ProductForLeverPickup AND NOT ProductOnElevator AND ElevatorBottomSensor
THEN
			State := 30;
		END_IF

	// move lever right with gripper enabled
	30:
		LeverGripper	:= TRUE;
		MoveLeverRight	:= TRUE;
		IF LeverRightPositionSensor THEN
			// if lever on right position then start to move left
			State := 10;
		END_IF


		// hold button released
	500:
		IF NOT Hold THEN
			State := HoldState;
			HoldState := 0;
		END_IF

	// stopping state
	1000:
		IF (MoveLeverRight AND LeverRightPositionSensor) OR (NOT MoveLeverRight AND
		    LeverLeftPositionSensor) THEN
			LeverGripper := FALSE;
			State := 0;
		END_IF

END_CASE

// e-stop pressed
IF State <> 0 AND Emergency THEN
	State := 0;
	LeverGripper := FALSE;
	MoveLeverRight := FALSE;
END_IF
```

```
// stop button pressed
IF State <> 0 AND State <> 1000 AND Stop THEN
	State := 1000;
END_IF

// hold button pressed
IF State<>0 AND State <>500 AND State<>1000 AND Hold THEN
	HoldRequest := TRUE;
END_IF

IF State <> 500 AND HoldRequest THEN
		HoldRequest := FALSE;
		HoldState := State;
		State := 500;
END_IF

// ==============  station state management =============
IF	State=0 THEN  StationState  := EnumStationState.NotRunning;
ELSIF
	State=500 THEN	StationState  := EnumStationState.Holding;
ELSIF
	State=1000 THEN StationState	:= EnumStationState.Stopping;
ELSE
				StationState  := EnumStationState.Running;
END_IF
```

```
PROGRAM ElevatorStation
VAR_INPUT
	Start		: BOOL;
	Stop		: BOOL;
	Hold		: BOOL;
	Emergency	: BOOL;
END_VAR

VAR
	State		: UINT;
	HoldState	: UINT;
	HoldRequest	: BOOL;
END_VAR

VAR_OUTPUT
	StationState  : EnumStationState;
END_VAR

CASE State OF
	// disable
	0:
		IF Start AND NOT Stop AND NOT emergency AND NOT Hold THEN
			State := 10;
		END_IF

	// wait for lever left
	10:
		IF LeverLeftPositionSensor THEN
			State := 20;
```

```
            END_IF

        // move elevator down to bottom sensor
        20:
            MoveElevatorDown    := TRUE;
            MoveElevatorUp              := FALSE;
            IF ElevatorBottomSensor THEN
                MoveElevatorDown := FALSE;
                MoveElevatorUp       := FALSE;
                State := 30;
            END_IF

        // wait for product and elevator ready
        30:
            IF ProductOnElevator AND LeverLeftPositionSensor AND NOT
                ProductOnDrillingStation THEN
                State := 40;
            END_IF

        // move elevator up
        40:
            MoveElevatorDown    := FALSE;
            MoveElevatorUp              := TRUE;
            IF ElevatorTopSensor THEN
                State := 50;
            END_IF

        // wait for product to be unloaded onto the drilling station
        50:
            IF NOT ProductOnElevator THEN
                // move elevator down
                State := 10;
            END_IF

        // hold button released
        500:
            IF NOT Hold THEN
                State := HoldState;
                HoldState := 0;
            END_IF

        // stopping state
        1000:
            IF (ElevatorTopSensor OR ElevatorBottomSensor) THEN
                State := 0;
                MoveElevatorDown := FALSE;
                MoveElevatorUp      := FALSE;
            END_IF


END_CASE

// e-stop pressed
IF State <> 0 AND Emergency THEN
        State := 0;
        MoveElevatorDown := FALSE;
        MoveElevatorUp      := FALSE;
```

```
END_IF

// stop button pressed
IF State <> 0 AND State <> 1000 AND Stop THEN
	State := 1000;
END_IF

// hold button pressed
IF State<>0 AND State <>500 AND State<>1000 AND Hold THEN
	HoldRequest := TRUE;
END_IF

// hold not allowed during state 20 and 40
IF State <> 500 AND State <> 20 AND State <> 40 AND HoldRequest THEN
		HoldRequest := FALSE;
		HoldState := State;
		State := 500;
END_IF

// ==============  station state management =============
IF	State=0 THEN StationState := EnumStationState.NotRunning;
ELSIF
	State=500 THEN StationState := EnumStationState.Holding;
ELSIF
	State=1000 THEN StationState := EnumStationState.Stopping;
ELSE
	StationState := EnumStationState.Running;
END_IF
```

```
PROGRAM DrillingStation

VAR_INPUT
	Start			: BOOL;
	Stop			: BOOL;
	Hold			: BOOL;
	Emergency		: BOOL;
END_VAR

VAR
	State			: UINT;
	HoldState		: UINT;
	HoldRequest		: BOOL;
	DrillingTimer		: TON;
END_VAR

VAR_OUTPUT
	StationState : EnumStationState;
END_VAR
```

```
CASE State OF
	// disabled
	0:
		DrillActuation	:= FALSE;
		AirActuation	:= FALSE;
		DrillingTimer.IN := False;
		IF Start AND NOT Stop AND NOT Emergency AND NOT Hold THEN
			State := 10;
		END_IF

	// move drill up
	10:
		MoveDrillUp := TRUE;
		MoveDrillDown := FALSE;
		IF DrillTopSensor THEN
			MoveDrillUp := FALSE;
			MoveDrillDown := FALSE;
			State := 20;
		END_IF

	// wait for product
	20:
		IF ProductOnDrillingStation THEN
			State := 30;
		END_IF

	// start drill & move drill down
	30:
		DrillActuation	:= TRUE;
		MoveDrillUp	:= FALSE;
		MoveDrillDown := TRUE;

		IF DrillBottomSensor THEN
			MoveDrillUp	:= FALSE;
			MoveDrillDown	:= FALSE;
			State := 40;
		END_IF

	// wait for drilling
	40:
		DrillingTimer.PT := T#2S;
		DrillingTimer.IN := TRUE;

		IF DrillingTimer.Q THEN
			DrillActuation	:= FALSE;
			DrillingTimer.IN := FALSE;
			State := 50;
		END_IF

	// move drill up
	50:
		MoveDrillUp	:= TRUE;
		MoveDrillDown := FALSE;

		IF DrillTopSensor THEN
			MoveDrillUp	:= FALSE;
			MoveDrillDown	:= FALSE;
```

```
                State := 60;
            END_IF

        // wait for table ready
        60:
            IF NOT ProductOnTableEntry THEN
                State := 70;
            END_IF

        // move product with air
        70:
            AirActuation := TRUE;
            IF ProductOnTableEntry THEN
                AirActuation := FALSE;
                State := 20;
            END_IF

        // hold button released
        500:
            IF NOT Hold THEN
                State := HoldState;
                HoldState := 0;
            END_IF

        // stopping state
        1000:
            IF (DrillTopSensor  OR DrillBottomSensor) THEN
                MoveDrillUp          :=     FALSE;
                MoveDrillDown :=     FALSE;
                State := 0;
            END_IF

END_CASE

DrillingTimer();


// e-stop pressed
IF State <> 0 AND Emergency THEN
        State := 0;
        MoveDrillUp   := FALSE;
        MoveDrillDown        := FALSE;
END_IF

// stop button pressed
IF State <> 0 AND State <> 1000 AND Stop THEN
        State := 1000;
END_IF

// hold button pressed
IF State<>0 AND State <>500 AND State<>1000 AND Hold THEN
        HoldRequest := TRUE;
END_IF
```

```
IF State <> 500 AND State <> 10 AND State <> 30 AND State <> 40 AND State <>50 AND State
<> 70 AND HoldRequest THEN
              HoldRequest := FALSE;
              HoldState := State;
              State := 500;
END_IF

// ==============  station state management =============
IF     State=0 THEN StationState := EnumStationState.NotRunning;
ELSIF
       State=500 THEN StationState := EnumStationState.Holding;
ELSIF
       State=1000 THEN StationState := EnumStationState.Stopping;
ELSE
       StationState := EnumStationState.Running;
END_IF
```

```
PROGRAM TableStation

VAR_INPUT
       Start                : BOOL;
       Stop                 : BOOL;
       Hold                 : BOOL;
       Emergency            : BOOL;
       ResetCounter  : BOOL;
END_VAR

VAR
       State                : UINT;
       HoldState            : UINT;
       HoldRequest          : BOOL;
       GenericTimer  : TON;
       Color                : BOOL;
END_VAR

VAR_OUTPUT
       StationState         : EnumStationState;
       RedProductCount             : UINT;
       BlueProductCount     : UINT;
END_VAR

CASE State OF

       // disabled
       0:
              GenericTimer.IN      :=  False;
              TableGripper  :=     FALSE;
              MoveTable            :=     FALSE;
              IF Start AND NOT Stop AND NOT Emergency AND NOT Hold THEN
                     State := 10;
              END_IF

       // move table to pickup position
       10:
              MoveTable            :=     TRUE;
              IF TablePickupPositionSensor THEN
```

```
                MoveTable           :=     FALSE;
                State               :=     20;
        END_IF

    // wait for product
    20:
        IF ProductOnTableEntry THEN
                State := 30;
        END_IF

    // wair for color read
    30:
        GenericTimer.IN := TRUE;
        GenericTimer.PT := T#0.5S;
        IF GenericTimer.Q THEN
                GenericTimer.IN := FALSE;
                Color               := ColorSensor;
                // count red and blue products
                IF ColorSensor THEN
                        BlueProductCount := BlueProductCount + 1;
                ELSE
                        RedProductCount := RedProductCount + 1;
                END_IF;

                State               := 40;
        END_IF

    // move table
    40:
        TableGripper        := TRUE;
        MoveTable                   := TRUE;

        IF (NOT Color AND TableRedDropPositionSensor) OR (Color AND
            TableBlueDropPositionSensor) THEN
                MoveTable           := FALSE;
                TableGripper  := FALSE;
                State               := 50;
        END_IF

    // wait for product to drop
    50:
        GenericTimer.IN := TRUE;
        GenericTimer.PT := T#0.5S;
        IF GenericTimer.Q THEN
                GenericTimer.IN := FALSE;
                State               := 10;
                // end of cycle time calculation
                IF CalculatingCycleTime THEN
                        CalculatingCycleTime := FALSE;
                        EndCycleTime := DateTimeProvider.GetDateTime();
                        CycleTime := ULINT_TO_REAL(EndCycleTime - StartCycleTime) /
1000;
                END_IF
        END_IF
```

```
        // hold button released
        500:
                IF NOT Hold THEN
                        State := HoldState;
                        HoldState := 0;
                END_IF

        // stopping state
        1000:
                IF (TablePickupPositionSensor OR  (Color AND TableBlueDropPositionSensor) OR
(NOT Color AND TableRedDropPositionSensor)) THEN
                        State := 0;
                END_IF

END_CASE

GenericTimer();


// e-stop pressed
IF State <> 0 AND Emergency THEN
        State := 0;
        MoveTable           := FALSE;
        TableGripper  := FALSE;
END_IF

// stop button pressed
IF State <> 0 AND State <> 1000 AND Stop THEN
        State := 1000;
END_IF

// hold button pressed
IF State<>0 AND State <>500 AND State<>1000 AND Hold THEN
        HoldRequest := TRUE;
END_IF

IF State <> 500 AND State <> 10 AND State <> 40 AND HoldRequest THEN
                HoldRequest := FALSE;
                HoldState := State;
                State := 500;
END_IF

// reset product counter
IF ResetCounter THEN
        RedProductCount  := 0;
        BlueProductCount := 0;
END_IF

// ==============  station state management =============
IF      State=0 THEN StationState := EnumStationState.NotRunning;
ELSIF
        State=500 THEN StationState := EnumStationState.Holding;
ELSIF
        State=1000 THEN StationState := EnumStationState.Stopping;
ELSE
        StationState := EnumStationState.Running;
END_IF
```

```
PROGRAM Main
VAR
	State		:	UINT;
END_VAR

ExtractionStation(Emergency:=EmergencyButton, ResetCounter := ResetCountersButton);
LeverStation(Emergency:=EmergencyButton);
ElevatorStation(Emergency:=EmergencyButton);
DrillingStation(Emergency:=EmergencyButton);
TableStation(Emergency:=EmergencyButton, ResetCounter := ResetCountersButton);


CASE State OF
	// machine not running
	0:
		ProcessProgress := 0;
		ExtractionStation.Start := FALSE;
		LeverStation.Start		:= FALSE;
		ElevatorStation.Start	:= FALSE;
		DrillingStation.Start	:= FALSE;
		TableStation.Start		:= FALSE;

		ExtractionStation.Stop  := FALSE;
		LeverStation.Stop		:= FALSE;
		ElevatorStation.Stop	:= FALSE;
		DrillingStation.Stop	:= FALSE;
		TableStation.Stop		:= FALSE;

		IF StartButton AND NOT StopButton AND NOT HoldButton AND NOT EmergencyButton
		THEN
			ExtractionStation.Start := TRUE;
			LeverStation.Start		:= TRUE;
			ElevatorStation.Start	:= TRUE;
			DrillingStation.Start	:= TRUE;
			TableStation.Start		:= TRUE;
			State := 10;
		END_IF

	// wait for all stations to be running
	10:
		IF ExtractionStation.StationState = EnumStationState.Running AND
			LeverStation.StationState    = EnumStationState.Running AND
			ElevatorStation.StationState  = EnumStationState.Running AND
			DrillingStation.StationState  = EnumStationState.Running AND
			TableStation.StationState     = EnumStationState.Running THEN

			ExtractionStation.Start := FALSE;
			LeverStation.Start		:= FALSE;
			ElevatorStation.Start	:= FALSE;
			DrillingStation.Start	:= FALSE;
			TableStation.Start		:= FALSE;

			State := 20;
		END_IF
```

```
	// machine running
	20:
		// update progress bar
		ProcessProgress := SEL(ProcessProgress<1600, 0, ProcessProgress + 1);

	// hold state
	500:
		IF ExtractionStation.StationState = EnumStationState.Holding AND
			LeverStation.StationState    = EnumStationState.Holding AND
			ElevatorStation.StationState  = EnumStationState.Holding AND
			DrillingStation.StationState  = EnumStationState.Holding AND
			TableStation.StationState     = EnumStationState.Holding AND
			NOT HoldButton THEN

			ExtractionStation.Hold  := FALSE;
			LeverStation.Hold       := FALSE;
			ElevatorStation.Hold    := FALSE;
			DrillingStation.Hold    := FALSE;
			TableStation.Hold       := FALSE;

			// resume to running state
			State := 10;
		END_IF


	// stop state
	1000:
		IF ExtractionStation.StationState = EnumStationState.NotRunning AND
			LeverStation.StationState    = EnumStationState.NotRunning AND
			ElevatorStation.StationState  = EnumStationState.NotRunning AND
			DrillingStation.StationState  = EnumStationState.NotRunning AND
			TableStation.StationState     = EnumStationState.NotRunning THEN

			State := 0;
		END_IF

END_CASE

// e-stop management
IF EmergencyButton THEN
	State := 0;
END_IF

// stop management
IF StopButton AND State <>0 AND State <> 1000 THEN
	ExtractionStation.Stop := TRUE;
	LeverStation.Stop       := TRUE;
	ElevatorStation.Stop    := TRUE;
	DrillingStation.Stop    := TRUE;
	TableStation.Stop       := TRUE;

	State := 1000;
END_IF

// hold management
IF HoldButton AND State<>0 AND State<>1000 AND State<>500THEN
```

```
        ExtractionStation.Hold  := TRUE;
        LeverStation.Hold       := TRUE;
        ElevatorStation.Hold    := TRUE;
        DrillingStation.Hold    := TRUE;
        TableStation.Hold       := TRUE;

        State := 500;
END_IF
```

```
VAR_GLOBAL
    ProductsInfeedEnable    :   BOOL;  // Command to enable the Product Production
    MovePistonForward       :   BOOL;  // Command to move the Piston Forward
    LeverGripper            :   BOOL;  // Command to grab products on the Lever
    MoveLeverRight          :   BOOL;  // Command to move the lever toward the left
    MoveElevatorUp          :   BOOL;  // Command to move the elevator up
    MoveElevatorDown        :   BOOL;  // Command to move the elevator down
    MoveDrillUp             :   BOOL;  // Command to move the drill up
    MoveDrillDown           :   BOOL;  // Command to move the drill down
    DrillActuation          :   BOOL;  // Command to run the drill
    AirActuation            :   BOOL;  // Command to run the air actuator
    MoveTable               :   BOOL;  // Command to move the rotating table
    TableGripper            :   BOOL;  // Command to grab products on the Table
END_VAR

VAR_GLOBAL
        StartButton                     :   BOOL;
        StopButton                      :   BOOL;
        EmergencyButton                 :   BOOL;
        ResetButton                     :   BOOL;
        HoldButton                      :   BOOL;
        ResetCountersButton             :   BOOL;
END_VAR

VAR_GLOBAL
        DateTimeProvider        :   DateTimeProvider;
        StartCycleTime          :   ULINT;
        EndCycleTime            :   ULINT;
        CycleTime               :   REAL;
        CalculatingCycleTime    :   BOOL;
        ProcessProgress         :   UINT;
END_VAR
```

```
VAR_GLOBAL
	ProductOnPistonStartSensor		:	BOOL;	// TRUE: Product present at the Start Position of the Piston
	PistonStartPositionSensor		:	BOOL;	// TRUE: Piston is at its Negative Limit

	ProductForLeverPickup			:	BOOL;	// TRUE: Product present at the Lever Left Position
	LeverLeftPositionSensor			:	BOOL;	// TRUE: The lever is at the Left Position
	LeverRightPositionSensor		:	BOOL;	// TRUE: The lever is at the Right Position

	ProductOnElevator				:	BOOL;	// TRUE: Product present on the Elevetor
	ElevatorBottomSensor			:	BOOL;	// TRUE: The elevator is at its Bottom Position
	ElevatorTopSensor				:	BOOL;	// TRUE: The elevator is at its Top Position

	DrillBottomSensor				:	BOOL;	// TRUE: The drill is at its Bottom Position
	DrillTopSensor					:	BOOL;	// TRUE: The drill is at its Top Position

	ProductOnDrillingStation		:	BOOL;	// TRUE: Product present below the Drill
	ProductOnTableEntry				:	BOOL;	// TRUE: Product present at the Table Entry

	ColorSensor						:	BOOL;	// Color detected at the Table Pickup Position - FALSE: Red, TRUE: Blue
	TablePickupPositionSensor		:	BOOL;	// TRUE: The rotating table grippper is at the pickup position
	TableBlueDropPositionSensor		:	BOOL;	// TRUE: The rotating table grippper is at the Blue Drop Position
	TableRedDropPositionSensor		:	BOOL;	// TRUE: The rotating table grippper is at the Red Drop Position
END_VAR
```