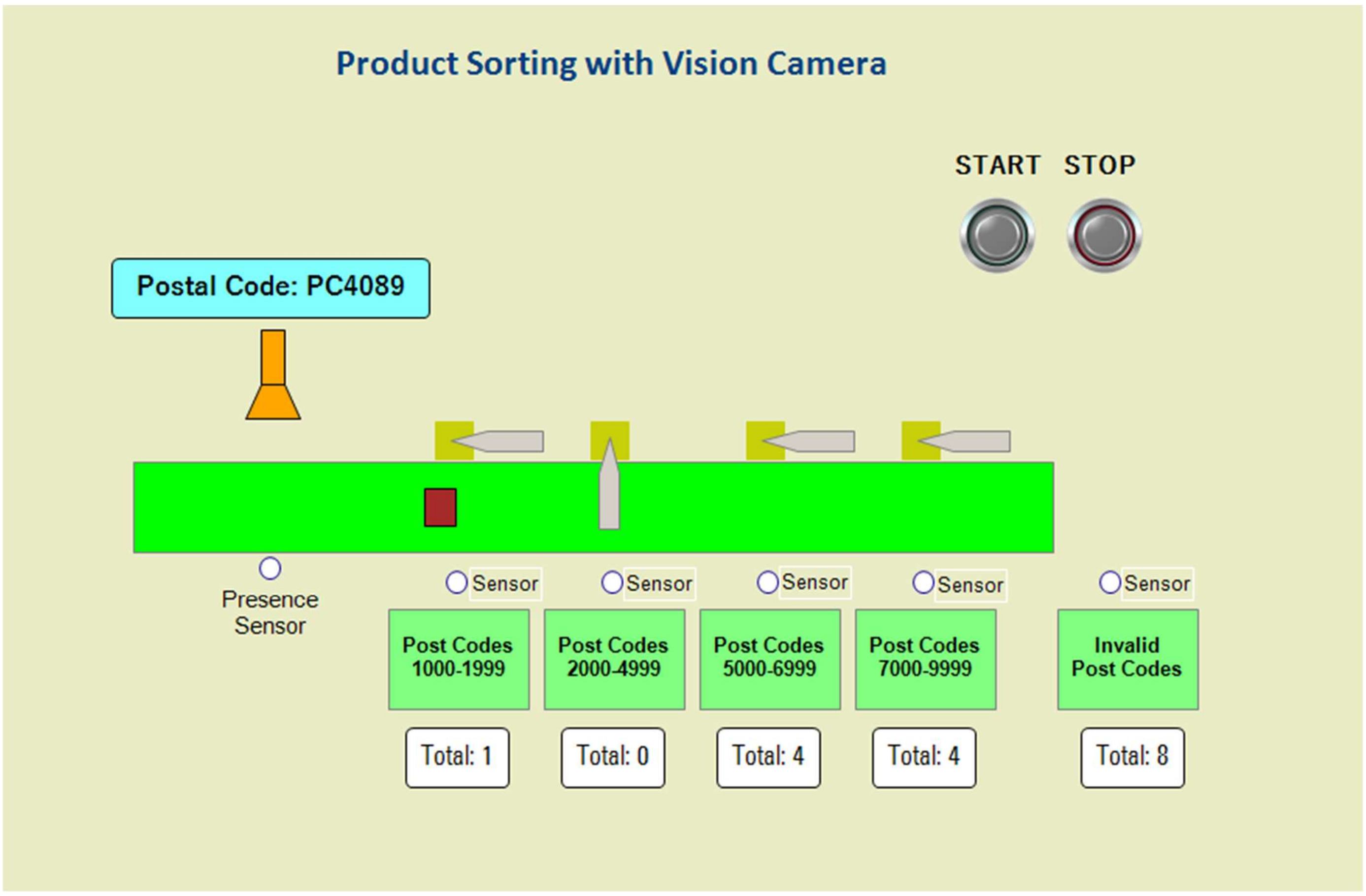
Sorting Products with Camera

TwinCAT 3 Project

The products are transported on a conveyor belt driven by motor. There are four swing arms which can swing out according to the postal code on the product.

Each product has a label with postal code that should be in range between PC0000 to PC9999.

Vision camera reads the postal code when the product is on the proper position (*PresenceSensor*)
Sometimes there can be no valid post code on the product. Products without a postal code, with unknown postal code or with invalid postal code should go to the last container.
There is only one product on the conveyor belt at a time. When a product is dropped into proper container , the swing arms are reset and next product can be passed by. Products of respective postal codes are counted.



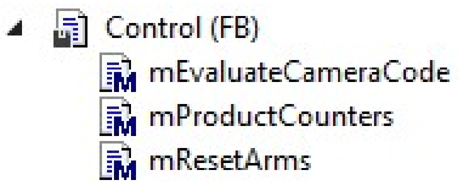
```
PROGRAM MAIN
VAR
      Control                 :     Control;
END_VAR

Simulation();

Control(StartPb   := GVL.StartButton,
            StopPb      := GVL.StopButton,
            PresenceSensor := GVL.ProductPresenceSensor,
            PostCodeSensor01 := GVL.PostCodeSensor01,
            PostCodeSensor02 := GVL.PostCodeSensor02,
            PostCodeSensor03 := GVL.PostCodeSensor03,
            PostCodeSensor04 := GVL.PostCodeSensor04,
            InvalidPostCodeSensor := GVL.InvalidPostCodesSensor,
            ConveyorMotorCmd => GVL.ConveyorMotor,
            Arm01Cmd => GVL.SwingArm01,
            Arm02Cmd => GVL.SwingArm02,
            Arm03Cmd => GVL.SwingArm03,
            Arm04Cmd => GVL.SwingArm04,
            );
```

```
TYPE stProductPos :
STRUCT
      X           :     REAL;
      y           :     REAL;
END_STRUCT
END_TYPE
```

```
{attribute 'qualified_only'}
VAR_GLOBAL
      ProductPresenceSensor    : BOOL;
      PostCodeSensor01         : BOOL;
      PostCodeSensor02         : BOOL;
      PostCodeSensor03         : BOOL;
      PostCodeSensor04         : BOOL;
      InvalidPostCodesSensor   : BOOL;
      StartButton              : BOOL;
      StopButton               : BOOL;
      ConveyorMotor            : BOOL;
      SwingArm01               : BOOL;
      SwingArm02               : BOOL;
      SwingArm03               : BOOL;
      SwingArm04               : BOOL;
END_VAR
```



```
Control (FB)
    mEvaluateCameraCode
    mProductCounters
    mResetArms
```

```
FUNCTION_BLOCK Control
VAR_INPUT
      StartPb                  :     BOOL;       // start pushbutton
      StopPb                   :     BOOL;       // stop pushbutton
      PresenceSensor           :     BOOL;       // presence sensor at camera
      PostCodeSensor01         :     BOOL;       // presence sensor at post codes 1000-1999 container
      PostCodeSensor02         :     BOOL;       // presence sensor at post codes 2000-4999 container
      PostCodeSensor03         :     BOOL;       // presence sensor at post codes 5000-6999 container
      PostCodeSensor04         :     BOOL;       // presence sensor at post codes 7000-9999 container
      InvalidPostCodeSensor    :     BOOL;       // presence sensor at invalid post code container
END_VAR

VAR_OUTPUT
      ConveyorMotorCmd         :     BOOL;       // command for conveyor motor
      Arm01Cmd                 :     BOOL;       // swing arm 01
      Arm02Cmd                 :     BOOL;       // swing arm 02
      Arm03Cmd                 :     BOOL;       // swing arm 03
      Arm04Cmd                 :     BOOL;       // swing arm 04
      ArmScrap                 :     BOOL;       // swing arm for scrap product

      PostCodeProducts01Count :     UINT;       // amount of products with post codes 1000-1999
      PostCodeProducts02Count :     UINT;       // amount of products with post codes 2000-4999
      PostCodeProducts03Count :     UINT;       // amount of products with post codes 5000-6999
      PostCodeProducts04Count :     UINT;       // amount of products with post codes 7000-9999
      InvalidPostCodeCount     :     UINT;       // amount of products with invalid post code
END_VAR
```

```
VAR

      SeqState           :      (IDLE,
                                RUN,
                                WAIT_FOR_PRODUCT,
                                PROCESSING_CAMERA,
                                EVALUATE_POSTCODE,
                                STOPPED);                // sequence state

      ConveyorMotor      :      ConveyorMotor;           // simulation of conveyor motor
      CameraDetect       :      CameraDetection;         // camera evaluation
      ProductCode        :      STRING;                  // post code evaluated by camera
      CameraReady        :      BOOL;
      rtProductSensor    :      R_TRIG;                  // edge of product sensor detection
      Arm                :      UINT;                    // swing arm to be activated
      tonProductDeliver :      ton;                      // time between camera read and proper container
      rtPostCodeSensor01      :      R_TRIG;             // detect edge of post code product 01
      rtPostCodeSensor02      :      R_TRIG;             // detect edge of post code product 02
      rtPostCodeSensor03      :      R_TRIG;             // detect edge of post code product 03
      rtPostCodeSensor04      :      R_TRIG;             // detect edge of post code product 04
      rtInvalidPostCodeSensor :      R_TRIG;             // detect edge of invalid post code product
END_VAR
```

```
(*
      FB controls transport conveyor and swing arms to place incoming products
      into proper containers according to postal code read out by vision camera.

      Presence Sensor : detect product presence at the camera position.
      PostCodeSensor01..04 : detect falling products with deteced post code into respective containers
    InvalidPostCodeSensor : detect falling product with invalid post code into container
*)

// FB for conveyor
ConveyorMotor(RunCmd=>ConveyorMotorCmd);

// FB for camera detection
CameraDetect(Trigger:=PresenceSensor,
                     Code=>ProductCode,
                     CodeReady=>CameraReady);

// cyclically evaluate amount of products of each type
THIS^.mProductCounters();


CASE SeqState OF

      IDLE: // wait to start
            IF StartPb AND NOT StopPb THEN
                  SeqState := RUN;
            END_IF

      RUN:  // run conveyor motor
            ConveyorMotor.Start:=TRUE;
            // check if motor running
            IF ConveyorMotor.RunCmd THEN
                  SeqState := WAIT_FOR_PRODUCT;
            END_IF
```

```
WAIT_FOR_PRODUCT: // wait for product presence & stop the conveyor
      rtProductSensor(CLK:=PresenceSensor);
      IF rtProductSensor.Q THEN
            ConveyorMotor.Start:=FALSE;
      END_IF

      // wait till conveyor stopped
      IF NOT ConveyorMotor.RunCmd THEN
            SeqState := PROCESSING_CAMERA;
      END_IF

PROCESSING_CAMERA:      // conveyor stopped & read out postal code from camera
      IF NOT CameraDetect.Busy THEN
                  SeqState := EVALUATE_POSTCODE;
      END_IF

EVALUATE_POSTCODE:      // evaluate the postal code and activate respective swing arm

      THIS^.mEvaluateCameraCode(CameraCode:=ProductCode, Arm=>Arm);
      Arm01Cmd := (Arm = 1);
      Arm02Cmd := (Arm = 2);
      Arm03Cmd := (Arm = 3);
      Arm04Cmd := (Arm = 4);
      ArmScrap := (Arm = 5);
      // delay to allow product transport into proper
      // container with activated swing arm
      tonProductDeliver.IN := TRUE;
      ConveyorMotor.Start := TRUE;

      IF tonProductDeliver.Q THEN
            tonProductDeliver.IN := FALSE;
            THIS^.mResetArms();
            SeqState := RUN;
      END_IF
      STOPPED:    // conveyor stopped
            ConveyorMotor.Start := FALSE;
            SeqState := IDLE;

END_CASE

tonProductDeliver(PT := T#12S);

IF StopPb AND SeqState <> IDLE AND SeqState <> STOPPED THEN
      SeqState := STOPPED;
END_IF
```

```
METHOD mEvaluateCameraCode : BOOL
VAR_INPUT
      CameraCode                  :      STRING;
END_VAR

VAR
      GetNum                      :      UINT;
END_VAR

VAR_OUTPUT
      Arm                         :      UINT;
END_VAR

(*
Check if postal code is valid i.e. starts with 'PC' and has following 4 digits
*)
IF LEN(CameraCode) = 6 THEN
      IF LEFT(CameraCode, 2) = 'PC' THEN
            GetNum := STRING_TO_UINT(MID(STR:=CameraCode, LEN:=1, POS:=3));

            IF GetNum = 1 THEN
                  Arm := 1;
            ELSIF  GetNum >=2 AND GetNum < 5 THEN
                  Arm := 2;
            ELSIF  GetNum >=5 AND GetNum < 7 THEN
                  Arm := 3;
            ELSIF  GetNum >=7 AND GetNum <= 9 THEN
                  Arm := 4;
            ELSE
                  Arm := 5;
            END_IF
      ELSE
            Arm := 5;
      END_IF

ELSE
      Arm := 5;
END_IF
```

```
METHOD mProductCounters : BOOL
VAR_INPUT
END_VAR

// counters for products of respective post codes
rtPostCodeSensor01(CLK := PostCodeSensor01);
rtPostCodeSensor02(CLK := PostCodeSensor02);
rtPostCodeSensor03(CLK := PostCodeSensor03);
rtPostCodeSensor04(CLK := PostCodeSensor04);
rtInvalidPostCodeSensor(CLK := InvalidPostCodeSensor);

// detect falling products into respective container
IF rtPostCodeSensor01.Q THEN
      PostCodeProducts01Count := PostCodeProducts01Count + 1;
ELSIF rtPostCodeSensor02.Q THEN
      PostCodeProducts02Count := PostCodeProducts02Count + 1;
ELSIF rtPostCodeSensor03.Q THEN
      PostCodeProducts03Count := PostCodeProducts03Count + 1;
ELSIF rtPostCodeSensor04.Q THEN
      PostCodeProducts04Count := PostCodeProducts04Count + 1;
ELSIF rtInvalidPostCodeSensor.Q THEN
      InvalidPostCodeCount := InvalidPostCodeCount + 1;
END_IF

IF rtPostCodeSensor01.Q OR
      rtPostCodeSensor02.Q OR
      rtPostCodeSensor03.Q OR
      rtPostCodeSensor04.Q OR
      rtInvalidPostCodeSensor.Q THEN

      tonProductDeliver.IN := FALSE;
      THIS^.mResetArms();
      SeqState := RUN;
END_IF
```

```
METHOD mResetArms : BOOL
VAR_INPUT
END_VAR

Arm01Cmd := FALSE;
Arm02Cmd := FALSE;
Arm03Cmd := FALSE;
Arm04Cmd := FALSE;
ArmScrap := FALSE;
```

```
FUNCTION_BLOCK CameraDetection
VAR_INPUT
      Trigger             :     BOOL;       // trigger for camera read
END_VAR
VAR_OUTPUT
      Code                :     STRING;     // code read out by camera
      CodeReady           :     BOOL;       // code ready
      Busy                :     BOOL;       // processing image
END_VAR
VAR
      ftReadCamera        :     F_TRIG;
      tpReadCamera        :     TP;
      CharCodes           :     ARRAY[0..9] OF STRING(2) := ['PC', 'PC', 'PC', 'PC',
'PC', 'PC', 'KL', 'PC', 'PC', 'CG'];
      RandCode            :     DRAND;
      RandNumber          :     DRAND;
      RandVal             :     LREAL;
      Idx                 :     UINT;
      CodeNumber          :     UINT;
END_VAR

(*
      Simulation of camera read detection
      Trigger starts reading process.
      Detected code is passed as Code on FB output.
      Detected code should have XXYYYY format, where
      XX - two char alphabetical code
      YYYY - four digit number
*)

// detect trigger and delay some time for stability
tpReadCamera(IN:=Trigger,
                    PT:=T#400MS,
                    Q=>,
                    ET=> );

IF tpReadCamera.Q THEN
      Busy := TRUE;
      CodeReady := FALSE;
ELSE
      Busy := FALSE;
END_IF

ftReadCamera(CLK:=tpReadCamera.Q);
CodeReady := FALSE;

IF ftReadCamera.Q THEN
      // create XX part of the code
      RandCode(Seed:=1, Num=>RandVal);
      Idx := LIMIT(0, LREAL_TO_UINT(RandVal * 10.0) - 1, 9);

      // create YYYY part of the code -> max value=9999
      RandNumber(Seed:=3, Num=>RandVal);
```

```
        CodeNumber := LREAL_TO_UINT(RandVal * 10000.0) - 1;

        IF Idx <= 9 THEN
                Code := CONCAT(CharCodes[Idx], UINT_TO_STRING(CodeNumber));
                CodeReady := TRUE;
        END_IF
END_IF
```

```
FUNCTION_BLOCK ConveyorMotor
VAR_INPUT
        Start               :       BOOL;
END_VAR
VAR_OUTPUT
        RunCmd              :       BOOL;
END_VAR
VAR
        tonStartTime        :       TON;
        tofStopTime         :       TOF;
END_VAR

(*
        Simulation of conveyor motor control
*)

tonStartTime(IN:=Start , PT:=T#300MS);
tofStopTime(IN:=Start , PT:=T#200MS);

RunCmd := tonStartTime.Q OR tofStopTime.Q;
```