```
PROGRAM  PLC_PRG
VAR
    PID_SERVO_X         :  fbdiscretePID ;
    PID_SERVO_Y         :  fbdiscretePID ;
    PushBMoveXY_enable :  BOOL ;          // PushButton
    PushBErr_ACK        :  BOOL ;          // PushButton
    Start_XYmovement    :  R_TRIG ;  // now in use for testing purposes, can be used to activate your application
    Error_ACK           :  R_TRIG ;  // now in use for testing purposes, can be used for other activities
    ProductOrderTable  :  ARRAY  [ 1 .. MAX_ArrayLength ]  OF  DosingRecipe2 ;

    PRoductActive       :  BOOL  :=  FALSE ;

    ServoXYState        :  XYState :=  IDLE ;  // Init state
    getXYsetpoints      :  fbGetXYCoordinates ;
    LocationNr          :  INT ;     // Obtained by the slider. See Vizualisation
    SaturationWarning  :  BOOL ;
    CurrentRecipeNr     :  SINT  := 0 ;
    LocationBits        :  BYTE ;  // Bit Patterns with component locations
    DropLocation        :  SINT ;  // Location to drop or deliver the components
    SetLocation        :  SINT ;  // variable used for the moving function
    NozzleLocation     :  SINT  :=  1 ;

    SettlingTime        :  TON ;    // Predetermined settling time
    NozzleChange        :  TON ;    // Time neded to Nozzle change at position 1
    DoseCompTime        :  TON ;   // Time needed to get or to drop component
    StartSettling       :  BOOL  :=  FALSE ;   // Predetermined Settling Time duration will start
    StartNozzleChange  :  BOOL  :=  FALSE ;   // Start Nozzle Change on Position 1
    StartDosing         :  BOOL  :=  FALSE ;   // to get new dosage or to deliver the dosage

    NextCompLoc         :  SINT ;   // Next location of Component [2..6]
    Recipes_aval       :  BOOL  :=  FALSE ;
    Recipe_OK           :  BOOL  :=  FALSE ;

    Err_Sign           :  BOOL  :=  FALSE ;   //Generic alarm
END_VAR
VAR  CONSTANT
    MAX_ArrayLength  :  SINT  :=  24 ;
//
//
END_VAR

```

```
// Create Recipe
IF  NOT  Recipes_aval  THEN
        CreateRecipe ( Table := ProductOrderTable ) ;
        Recipes_aval :=  TRUE ;
        Recipe_OK   :=  TRUE ;   // Is Set for testing purposes
```

```
//
END_IF

//The below code is for PID testing purposes ONLY. comment out the state machine for use.
// IF (Start_XYmovement.Q)THEN
// //
// Recipe_OK := TRUE; // Only for testing. should be removed later
//        getXYsetpoints(Pos:=LocationNr, // get setpoints XY for location number;
//        XCoor=>PID_SERVO_X.Setpoint,
//        YCoor=>PID_SERVO_Y.Setpoint);
// END_IF

// Following Function Blocks should be executed every cycle
SettlingTime ( in :=  StartSettling , PT :=  T#2500MS ) ;
NozzleChange ( in :=  StartNozzleChange , PT :=  T#3000MS ) ;   // Nozzle Change Time; SEE PART 2 DOC
DoseCompTime ( in := StartDosing , PT  :=  T#2000MS ) ;    // Picking / Dropping Time ; SEE PART 2 DOC

//Push buttons
Error_ACK ( CLK :=  PushBErr_ACK ) ;
Start_XYmovement ( CLK := PushBMoveXY_enable ) ;

CASE  ServoXYState  OF
    IDLE :            //IDLE state; no actrions to be taken.
        Err_Sign  :=  FALSE ;
        IF  Start_XYmovement . Q  OR  PRoductActive  THEN
            PRoductActive  :=  TRUE ;
            CurrentRecipeNr  :=  CurrentRecipeNr  +  1 ;
            ServoXYState  :=  CHECKING ;   // Start with checking the recipe
        END_IF

    CHECKING :
        recipe_OK  :=  CheckRecipeNr ( ProductOrderTable [ CurrentRecipeNr ] ) ;

        IF  ( recipe_OK )  THEN
            LocationBits  :=  ProductOrderTable [ CurrentRecipeNr ] . PosLocBits ;   // Location position bits
            DropLocation  :=  ProductOrderTable [ CurrentRecipeNr ] . DeliveryLocation ;  // Location for delivery
            StartSettling  :=  FALSE ;
            NextCompLoc  :=  NxtComp_Locnr ( LocationBits ) ;  // Go to (next)component location
            SetLocation  :=  NozzleLocation ;
            ServoXYState :=  MOVING ;
            recipe_OK :=  FALSE ;

        ELSE        //error; recipe not oke
          ServoXYState :=  CHECKING ;
          Err_Sign  :=  TRUE ;
          PRoductActive :=  FALSE ;   // Production on Hold ( Stopped) due recipe error..

          //go to nozzle location
```

```
     getXYsetpoints ( pos := NozzleLocation ,
     XCoor => PID_SERVO_X . Setpoint ,
     YCoor => PID_SERVO_Y . Setpoint ) ;

     IF Error_ACK . Q THEN      //Resume production on error button press.
       PRoductActive := TRUE ;
       ServoXYState := IDLE ;
     END_IF
    END_IF

  MOVING :
    IF ( NextCompLoc <> 0 ) THEN      // Next Comp location is found; move to designated location
      IF ( StartSettling = FALSE ) THEN
        getXYsetpoints ( pos := SetLocation ,
        XCoor => PID_SERVO_X . Setpoint ,
        YCoor => PID_SERVO_Y . Setpoint ) ;
        StartSettling := TRUE ;
      END_IF

      IF ( SettlingTime . Q ) THEN
        ServoXYState := NOZZLING ;
      END_IF
    ELSE
      ServoXYState := IDLE ;
    END_IF

  // Consider to merge State 2, State 3 and State 4 as one State : "ServoMove_State" (combination of MOVING & NOZZLING states)
  NOZZLING :

    IF ( SetLocation = NozzleLocation ) THEN
      StartNozzleChange := TRUE ;
    ELSIF ( ( SetLocation = NextCompLoc ) OR ( SetLocation = DropLocation ) ) THEN
      StartDosing := TRUE ;
    END_IF

    IF NozzleChange . Q THEN
      StartNozzleChange := FALSE ;
      StartSettling := FALSE ;
      SetLocation := NextCompLoc ;
    ELSIF ( ( DoseCompTime . Q ) AND ( SetLocation <> DropLocation ) ) THEN
      StartDosing := FALSE ;
      StartSettling := FALSE ;
      SetLocation := DropLocation ;
    ELSIF DoseCompTime . Q THEN
      StartDosing := FALSE ;
      StartSettling := FALSE ;
      NextCompLoc := NxtComp_Locnr ( LocationBits ) ;
      SetLocation := NozzleLocation ;
    END_IF
    ServoXYState := MOVING ;
END_CASE ;
```

```

(*
// ============================================================
// Executing PID controllers every cycle. Do not change or modify the source code.
// =======================================================
*)
PID_SERVO_X . Auto_Mode := TRUE ;
PID_SERVO_X . Kp := 0.837600052042507 ;  // tuned
PID_SERVO_X . Ki := 0 ;  // tuned
PID_SERVO_X . Kd := 0.171610226044254 ;  // tuned
PID_SERVO_X . MV_max_sat := 10 ;
PID_SERVO_X . MV_min_sat := - 10 ;
PID_SERVO_X . PrGain := 10 ;
PID_SERVO_X ( PoccessValue := SERVO_XPOS . Yout ,
            q_MV_out  =>  SERVO_XPOS . Xin ,
            qx_Saturation => SaturationWarning ) ;

PID_SERVO_Y . Auto_Mode := TRUE ;
PID_SERVO_Y . Kp := 0.390618492352991 ;  // tuned
PID_SERVO_Y . Ki := 0 ;  // tuned
PID_SERVO_Y . Kd := 0.177236237453083 ;  // tuned (very little overshoot from pos40 to pos1)
PID_SERVO_Y . MV_max_sat := 10 ;
PID_SERVO_Y . MV_min_sat := - 10 ;
PID_SERVO_Y . PrGain := 10 ;
PID_SERVO_Y ( PoccessValue := SERVO_YPOS2 . Yout ,
            q_MV_out  =>  SERVO_YPOS2 . Xin ,
            qx_Saturation => SaturationWarning ) ;

```