Project Documentation

Table of contents

1 Device: Device_1

Users and Groups

Users:

Groups

Access Rights

View
Modify
Execute
Add/remove children

Symbol Rights

Information

Name:	CODESYS Control Win V3 x64
Vendor:	3S - Smart Software Solutions GmbH
Categories:	PLCs
Type:	4096
ID:	0000 0004
Version:	3.5.19.20
Order number:	???
Description:	CODESYS V3 Soft-PLC for Windows with non realtime capabilities (CODESYS Control Win V3 x64)

1.1 PLC Logic: Plc Logic

1.1.1 Application: Application

1.1.1.1 Folder: POUs

1.1.1.1.1 Folder: UM_PalletWrapper

1.1.1.1.1.1 Folder: DUTs

1.1.1.1.1.1.1 DUT: AplicatorMode

```
1   TYPE AplicatorMode :
2   STRUCT
3       sModeName : STRING;
4       arrLayeringSystem : ARRAY[1..8] OF USINT;
5   END_STRUCT
6   END_TYPE
7
```

1.1.1.1.1.1.2 Folder: PalletWrapperError

1.1.1.1.1.1.2.1 DUT: PW_ERROR_ID

```
1   {attribute 'qualified_only'}
2   {attribute 'strict'}
3   TYPE PW_ERROR_ID :
4   (
5       PW_NO_ERROR := 0,
6       PW_FILM_NOT_DETECTED_ERROR := 1,
7       PW_CLAMP_CLOSE_ERROR := 2,
8       PW_CLAMP_RELEASE_ERROR := 3,
9
10      PW_STICKER_ATTACH_ERROR := 4,
11      PW_STICKER_DETACH_ERROR := 5,
12
13      PW_PROTECTOR_EXTEND_ERROR := 6,
14      PW_PROTECTOR_RELEASE_ERROR := 7,
15
16      PW_APLICATOR_POWER_ERROR := 8,
17      PW_ROTATOR_POWER_ERROR := 9,
18      PW_APLICATOR_HOME_ERROR := 10,
19      PW_ROTATOR_HOME_ERROR := 11,
20      PW_APLICATOR_HALT_ERROR := 12,
21      PW_ROTATOR_HALT_ERROR := 13,
22      PW_APLICATOR_STOP_ERROR := 14,
23      PW_ROTATOR_STOP_ERROR := 15,
24      PW_APLICATOR_MOVE_ERROR := 16,
25      PW_ROTATOR_MOVE_ERROR := 17,
26      PW_ENTER_CONVEYOR_DRIVE_ERROR := 18,
27      PW_CENTER_CONVEYOR_DRIVE_ERROR := 19,
28      PW_EXIT_CONVEYOR_DRIVE_ERROR := 20,
```

```
        PW_ENTER_CONVEYOR_SENSOR_ERROR  := 21,
        PW_CENTER_CONVEYOR_SENSOR_ERROR  := 22,
        PW_EXIT_CONVEYOR_SENSOR_ERROR  := 23,
        PW_APLICATOR_PALLET_SENSOR_ERROR  := 24,
        PW_ENTER_CONVEYOR_USAGE_ERROR  := 25,
        PW_CENTER_CONVEYOR_USAGE_ERROR  := 26,
        PW_EXIT_CONVEYOR_USAGE_ERROR  := 27,
        PW_ROTATOR_USAGE_ERROR  := 28,
        PW_APLICATOR_USAGE_ERROR  := 29
    );
    END_TYPE

```

1.1.1.1.1.1.2.2 POU: PW_ERROR

```
    FUNCTION_BLOCK PUBLIC PW_ERROR
    VAR_INPUT
        PW_ErrorID : PW_ERROR_ID;
        sErrorCauseMessage : STRING;
        sErrorSolutionMessage : STRING;
    END_VAR

```

```

```

1.1.1.1.1.1.2.3 POU: PW_getErrorSolution

```
    FUNCTION PW_getErrorSolution : STRING
    VAR_INPUT
        PALLET_WRAPPER_ERROR : PW_ERROR_ID;
    END_VAR

```

```
    CASE PALLET_WRAPPER_ERROR OF
        0: PW_getErrorSolution := 'No errors';
        1: PW_getErrorSolution := 'The foil has run out or broken. Insert the
    foil between the clamping jaws.';
        2: PW_getErrorSolution := 'Clamp actuator does not extend properly.
    Check the extension sensor.';
        3: PW_getErrorSolution := 'Clamp actuator does not retract properly.
    Check the retraction sensor.';
        4: PW_getErrorSolution := 'Sticker actuator does not extend properly.
    Check the extension sensor.';
        5: PW_getErrorSolution := 'Sticker actuator does not retract properly.
    Check the retraction sensor.';
        6: PW_getErrorSolution := 'Protector actuator does not extend
    properly. Check the extension sensor.';
        7: PW_getErrorSolution := 'Protector actuator does not retract
    properly. Check the retraction sensor.';
        8: PW_getErrorSolution := 'Servo Axis error: ';
    END_CASE

```

1.1.1.1.1.2 Folder: Globals

1.1.1.1.1.2.1 Global Variable List: AplicatorModeNames

```
{attribute 'qualified_only'}
VAR_GLOBAL
    arrModeNames : ARRAY [0..2] OF STRING := ['Eco', 'Normal', 'High Performance'];
END_VAR
```

1.1.1.1.1.2.2 Global Variable List: AplicatorParameters

```
{attribute 'qualified_only'}
VAR_GLOBAL
    LR_HOME_POSITION : LREAL := 0;
    LR_JERK : LREAL := 0;
    LR_DECELERATION : LREAL := 1000;
    LR_ACCELERATION : LREAL := 100;
    LR_VELOCITY : LREAL := 3000;
    LR_SLOW_VELOCITY : LREAL := 1000;
    LR_SLOW_ACCELERATION : LREAL := 20;
    LRA_LAYER_POSITIONS : ARRAY [1..8] OF LREAL := [
        50,
        500,
        950,
        1400,
        1400,
        950,
        500,
        50];

END_VAR

```

1.1.1.1.1.2.3 Global Variable List: ConveyorParameters

```
{attribute 'qualified_only'}
VAR_GLOBAL
    LR_JERK : LREAL := 0;
    LR_DECELERATION : LREAL := 1000;
    LR_ACCELERATION : LREAL := 100;
    LR_VELOCITY : LREAL := 3000;
END_VAR

```

1.1.1.1.1.2.4 Global Variable List: LayerSystems

```
VAR_GLOBAL CONSTANT
    //Layering : base, 1, 2, top, top, 2, 1, base
    arrEcoSystem : ARRAY[1..8] OF USINT := [2,1,1,1,1,0,0,1]; // max 7 wraps per pallet
    arrNormalSystem : ARRAY[1..8] OF USINT := [2,1,1,1,1,1,0,1]; // max 8 wraps per pallet
    arrHighPerformanceSystem : ARRAY[1..8] OF USINT := [2,1,1,1,1,1,1,2]; // max 10 wraps per pallet
END_VAR

```

1.1.1.1.1.2.5 Global Variable List: RotatorParameters

```
{attribute 'qualified_only'}
VAR_GLOBAL
    LR_HOME_POSITION : LREAL := 0;
    LR_JERK : LREAL := 0;
    LR_DECELERATION : LREAL := 1000;
    LR_ACCELERATION : LREAL := 100;
    LR_VELOCITY : LREAL := 3000;
    LR_FIXED_POSITION : LREAL := 360;
END_VAR

```

1.1.1.1.1.2.6 Global Variable List: UnitModeNames

```
VAR_GLOBAL CONSTANT
    arrUnitModeNames : ARRAY [0..1] OF STRING := ['Production','Manual'];
END_VAR

```

1.1.1.1.1.3 Folder: POUs

1.1.1.1.1.3.1 POU: CM_CenterConveyor

```
FUNCTION_BLOCK PUBLIC CM_CenterConveyor EXTENDS ConveyorAxis
VAR_INPUT
    Ref_CenterConveyorAxis : REFERENCE TO AXIS_REF_SM3;
END_VAR
VAR_OUTPUT
    PW_Error : PW_ERROR;
END_VAR

```

1.1.1.1.1.3.1.1 Method: EnableDrivePower

```
METHOD EnableDrivePower : BOOL
VAR
    bError : BOOL;
    ErrorID : SMC_ERROR;
END_VAR
VAR_OUTPUT
    bPowerStatus : BOOL;
END_VAR

```

```
CASE THIS^.Ref_CenterConveyorAxis.nAxisState OF
    0,1,2 : SUPER^.EnablePower(Ref_Axis := THIS^.Ref_CenterConveyorAxis,
                        bPowerStatus => bPowerStatus,
                        bError => bError,
                        ErrorID => ErrorID);
ELSE
    bPowerStatus := TRUE;
END_CASE
IF bError AND ErrorID <> 0 THEN
    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
    THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR);
END_IF

```

1.1.1.1.1.3.1.2 Method: HaltDrive

```
METHOD PUBLIC HaltDrive
VAR
    bError : BOOL;
    ErrorID : SMC_ERROR;
    sErrorCauseMessage : STRING;
    sErrorSolutionMessage : STRING;
END_VAR
VAR_OUTPUT
    bHaltDone : BOOL;
    bBusy : BOOL;
END_VAR

```

```
SUPER^.ExecuteHalt(Ref_Axis := THIS^.Ref_CenterConveyorAxis,
            lrDeceleration := ConveyorParameters.LR_DECELERATION,
            lrJerk := ConveyorParameters.LR_JERK,
            bStatus => bHaltDone,
            bBusy => bBusy,
            bError => bError,
            ErrorID => ErrorID);

```

```
    IF bError AND ErrorID <> 0 THEN
        sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
        sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR);

        THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR;
        THIS^.PW_Error.sErrorCauseMessage := sErrorCauseMessage;
        THIS^.PW_Error.sErrorSolutionMessage := sErrorSolutionMessage;
    END_IF
```

1.1.1.1.1.3.1.3 Method: MoveVelocity

```
    METHOD PUBLIC MoveVelocity
    VAR_INPUT
        intDirection : MC_DIRECTION;
    END_VAR
    VAR
        bError : BOOL;
        ErrorID : SMC_ERROR;
        sErrorCauseMessage : STRING;
        sErrorSolutionMessage : STRING;
    END_VAR
    VAR_OUTPUT
        bVelocityReached : BOOL;
    END_VAR
```

```
    SUPER^.ExecuteMoveVel(Ref_Axis := THIS^.Ref_CenterConveyorAxis,
                lrVelocity := ConveyorParameters.LR_VELOCITY,
                lrAcceleration := ConveyorParameters.LR_ACCELERATION,
                lrDeceleration := ConveyorParameters.LR_DECELERATION,
                lrJerk := ConveyorParameters.LR_JERK,
                intDirection := intDirection,
                bVelocityReached => bVelocityReached,
                bError => bError,
                ErrorID => ErrorID);

    IF bError AND ErrorID <> 0 THEN
        sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
        sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR);

        THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR;
        THIS^.PW_Error.sErrorCauseMessage := sErrorCauseMessage;
        THIS^.PW_Error.sErrorSolutionMessage := sErrorSolutionMessage;

    END_IF
```

1.1.1.1.1.3.1.4 Method: ResetFaults

```
METHOD PUBLIC ResetFaults
VAR
    bErrorCancelled : BOOL := TRUE ;   // HMI confiramtion
    sErrorCause : STRING ;
    sErrorSolution : STRING ;
END_VAR
VAR_OUTPUT
    bResetFaultDone : BOOL := FALSE ;
END_VAR

```

```
IF THIS^.PW_Error.PW_ErrorID <> PW_ERROR_ID.PW_NO_ERROR THEN
    sErrorCause := THIS^.PW_Error.sErrorCauseMessage ;
    sErrorSolution := THIS^.PW_Error.sErrorSolutionMessage ;
    IF bErrorCancelled THEN
        THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR ;
        THIS^.PW_Error.sErrorCauseMessage := '' ;
        THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution (
PW_ERROR_ID.PW_NO_ERROR ) ;
    bResetFaultDone := TRUE ;
    END_IF
    bResetFaultDone := FALSE ;
ELSE
    bResetFaultDone := TRUE ;
END_IF


```

1.1.1.1.1.3.1.5 Method: StopDrive

```
METHOD PUBLIC StopDrive
VAR
    bBusy : BOOL ;
    bError : BOOL ;
    ErrorID : SMC_ERROR ;
    sErrorCauseMessage : STRING ;
    sErrorSolutionMessage : STRING ;
END_VAR
VAR_OUTPUT
    bStopDone : BOOL ;
END_VAR

```

```
CASE THIS^.Ref_CenterConveyorAxis.nAxisState OF
    4, 5, 6, 7 : SUPER^.ExecuteStop ( Ref_Axis := THIS^.
Ref_CenterConveyorAxis ,
                lrDeceleration := ConveyorParameters.LR_DECELERATION ,
                lrJerk := ConveyorParameters.LR_JERK ,
                bStatus => bStopDone ,
                bBusy => bBusy ,
                bError => bError ,
```

```
                    ErrorID  =>  ErrorID ) ;

        IF  bError  AND  ErrorID  <>  0  THEN
            sErrorCauseMessage  :=  SMC_ErrorString ( ErrorId  :=  ErrorID ,
    Language  :=  0 ) ;
            sErrorSolutionMessage  :=  PW_getErrorSolution ( PW_ERROR_ID .
    PW_ENTER_CONVEYOR_DRIVE_ERROR ) ;

            THIS ^ . PW_Error . PW_ErrorID  :=  PW_ERROR_ID .
    PW_ENTER_CONVEYOR_DRIVE_ERROR ;
            THIS ^ . PW_Error . sErrorCauseMessage  :=  sErrorCauseMessage ;
            THIS ^ . PW_Error . sErrorSolutionMessage  :=  sErrorSolutionMessage ;
        END_IF
    END_CASE

```

1.1.1.1.1.3.1.6 Method: ThrowError

```
    METHOD  PUBLIC  ThrowError
    VAR_INPUT
        ThrownError  :  PW_ERROR ;
    END_VAR

```

```
    THIS ^ . PW_Error . PW_ErrorID  :=  ThrownError . PW_ErrorID ;
    THIS ^ . PW_Error . sErrorCauseMessage  :=  ThrownError . sErrorCauseMessage ;
    THIS ^ . PW_Error . sErrorSolutionMessage  :=  ThrownError .
    sErrorSolutionMessage ;

```

1.1.1.1.1.3.2 POU: CM_EnterConveyor

```
    FUNCTION_BLOCK  PUBLIC  CM_EnterConveyor  EXTENDS  ConveyorAxis
    VAR_INPUT
        Ref_EnterConveyorAxis  :  REFERENCE  TO  AXIS_REF_SM3 ;
    END_VAR
    VAR_OUTPUT
        PW_Error  :  PW_ERROR ;
    END_VAR

```

1.1.1.1.1.3.2.1 Method: EnableDrivePower

```
    METHOD  PUBLIC  EnableDrivePower
    VAR
        bError  :  BOOL ;
        ErrorID  :  SMC_ERROR ;
        PW_Error  :  PW_ERROR ;
    END_VAR
    VAR_OUTPUT
        bPowerStatus  :  BOOL ;
    END_VAR

```

```
CASE THIS^.Ref_EnterConveyorAxis.nAxisState OF
    0,1,2 : SUPER^.EnablePower(Ref_Axis := THIS^.Ref_EnterConveyorAxis,
                        bPowerStatus => bPowerStatus,
                        bError => bError,
                        ErrorID => ErrorID);
ELSE bPowerStatus := TRUE;
END_CASE

IF bError AND ErrorID <> 0 THEN
    PW_Error.PW_ErrorID    := PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR;
    PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
    PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR);

    ThrowError(ThrownError := PW_Error);
END_IF

```

1.1.1.1.1.3.2.2 Method: HaltDrive

```
METHOD PUBLIC HaltDrive
VAR
    bError : BOOL;
    ErrorID : SMC_ERROR;
    PW_Error : PW_ERROR;
END_VAR
VAR_OUTPUT
    bHaltDone : BOOL;
    bBusy : BOOL;
END_VAR

```

```
SUPER^.ExecuteHalt(Ref_Axis := THIS^.Ref_EnterConveyorAxis,
            lrDeceleration := ConveyorParameters.LR_DECELERATION,
            lrJerk := ConveyorParameters.LR_JERK,
            bStatus => bHaltDone,
            bBusy => bBusy,
            bError => bError,
            ErrorID => ErrorID);

IF bError AND ErrorID <> 0 THEN
    PW_Error.PW_ErrorID    := PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR;
    PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
    PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR);

    ThrowError(ThrownError := PW_Error);
END_IF

```

1.1.1.1.1.3.2.3 Method: MoveVelocity

```
METHOD PUBLIC MoveVelocity
VAR_INPUT
    intDirection : MC_DIRECTION;
END_VAR
VAR
    bError : BOOL;
    ErrorID : SMC_ERROR;;
    PW_Error : PW_ERROR;
END_VAR
VAR_OUTPUT
    bVelocityReached : BOOL;
END_VAR

```

```
SUPER^.ExecuteMoveVel(Ref_Axis := THIS^.Ref_EnterConveyorAxis,
                lrVelocity := ConveyorParameters.LR_VELOCITY,
                lrAcceleration := ConveyorParameters.LR_ACCELERATION,
                lrDeceleration := ConveyorParameters.LR_DECELERATION,
                lrJerk := ConveyorParameters.LR_JERK,
                intDirection := intDirection,
                bVelocityReached => bVelocityReached,
                bError => bError,
                ErrorID => ErrorID);

IF bError AND ErrorID <> 0 THEN
    PW_Error.PW_ErrorID    := PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR;
    PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
    PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR);

    ThrowError(ThrownError := PW_Error);
END_IF

```

1.1.1.1.1.3.2.4 Method: ResetFaults

```
METHOD PUBLIC ResetFaults
VAR
    bErrorCancelled : BOOL := TRUE;   // HMI confiramtion
    sErrorCause : STRING;
    sErrorSolution : STRING;
END_VAR
VAR_OUTPUT
    bResetFaultDone : BOOL := FALSE;
END_VAR

```

```
IF THIS^.PW_Error.PW_ErrorID <> PW_ERROR_ID.PW_NO_ERROR THEN
    sErrorCause := THIS^.PW_Error.sErrorCauseMessage;
    sErrorSolution := THIS^.PW_Error.sErrorSolutionMessage;
    IF bErrorCancelled THEN
```

```
            THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
            THIS^.PW_Error.sErrorCauseMessage := '';
            THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_NO_ERROR);
        bResetFaultDone := TRUE;
        END_IF
        bResetFaultDone := FALSE;
    ELSE
        bResetFaultDone := TRUE;
    END_IF
```

1.1.1.1.1.3.2.5 Method: StopDrive

```
    METHOD PUBLIC StopDrive
    VAR
        bBusy : BOOL;
        bError : BOOL;
        ErrorID : SMC_ERROR;
        PW_Error : PW_ERROR;
    END_VAR
    VAR_OUTPUT
        bStopDone : BOOL;
    END_VAR
```

```
    CASE THIS^.Ref_EnterConveyorAxis.nAxisState OF
        4,5,6,7 : SUPER^.ExecuteStop(Ref_Axis := THIS^.Ref_EnterConveyorAxis,
                    lrDeceleration := ConveyorParameters.LR_DECELERATION,
                    lrJerk := ConveyorParameters.LR_JERK,
                    bStatus => bStopDone,
                    bBusy => bBusy,
                    bError => bError,
                    ErrorID => ErrorID);

        IF bError AND ErrorID <> 0 THEN
            PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR;
            PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
            PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR);

            ThrowError(ThrownError := PW_Error);
        END_IF
    END_CASE
```

1.1.1.1.1.3.2.6 Method: ThrowError

```
METHOD PUBLIC ThrowError
VAR_INPUT
    ThrownError : PW_ERROR;
END_VAR

```

```
THIS^.PW_Error.PW_ErrorID := ThrownError.PW_ErrorID;
THIS^.PW_Error.sErrorCauseMessage := ThrownError.sErrorCauseMessage;
THIS^.PW_Error.sErrorSolutionMessage := ThrownError.sErrorSolutionMessage;

```

1.1.1.1.1.3.3 POU: CM_ExitConveyor

```
FUNCTION_BLOCK PUBLIC CM_ExitConveyor EXTENDS ConveyorAxis
VAR_INPUT
    Ref_ExitConveyorAxis : REFERENCE TO AXIS_REF_SM3;
END_VAR
VAR_OUTPUT
    PW_Error : PW_ERROR;
END_VAR

```

1.1.1.1.1.3.3.1 Method: EnableDrivePower

```
METHOD EnableDrivePower : BOOL
VAR
    bError : BOOL;
    ErrorID : SMC_ERROR;
END_VAR
VAR_OUTPUT
    bPowerStatus : BOOL;
END_VAR

```

```
CASE THIS^.Ref_ExitConveyorAxis.nAxisState OF
    0,1,2 : SUPER^.EnablePower(Ref_Axis := THIS^.Ref_ExitConveyorAxis,
                               bPowerStatus => bPowerStatus,
                               bError => bError,
                               ErrorID => ErrorID);
ELSE
    bPowerStatus := TRUE;
END_CASE
IF bError AND ErrorID <> 0 THEN
    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
```

```
12          THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
        PW_ERROR_ID.PW_APLICATOR_POWER_ERROR);
13      END_IF
14
```

1.1.1.1.1.3.3.2 Method: HaltDrive

```
1       METHOD PUBLIC HaltDrive
2       VAR
3           bError : BOOL;
4           ErrorID : SMC_ERROR;
5           sErrorCauseMessage : STRING;
6           sErrorSolutionMessage : STRING;
7       END_VAR
8       VAR_OUTPUT
9           bHaltDone : BOOL;
10          bBusy : BOOL;
11      END_VAR
12
```

```
1       SUPER^.ExecuteHalt(Ref_Axis := THIS^.Ref_ExitConveyorAxis,
2                   lrDeceleration := ConveyorParameters.LR_DECELERATION,
3                   lrJerk := ConveyorParameters.LR_JERK,
4                   bStatus => bHaltDone,
5                   bBusy => bBusy,
6                   bError => bError,
7                   ErrorID => ErrorID);
8
9       IF bError AND ErrorID <> 0 THEN
10          sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language
        := 0);
11          sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.
        PW_ENTER_CONVEYOR_DRIVE_ERROR);
12
13          THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.
        PW_ENTER_CONVEYOR_DRIVE_ERROR;
14          THIS^.PW_Error.sErrorCauseMessage := sErrorCauseMessage;
15          THIS^.PW_Error.sErrorSolutionMessage := sErrorSolutionMessage;
16      END_IF
17
```

1.1.1.1.1.3.3.3 Method: MoveVelocity

```
1       METHOD PUBLIC MoveVelocity
2       VAR_INPUT
3           intDirection : MC_DIRECTION;
4       END_VAR
5       VAR
6           bError : BOOL;
7           ErrorID : SMC_ERROR;
8           sErrorCauseMessage : STRING;
9           sErrorSolutionMessage : STRING;
10      END_VAR
11      VAR_OUTPUT
12          bVelocityReached : BOOL;
```

```
13      END_VAR
14
```

```
1       SUPER^.ExecuteMoveVel(Ref_Axis := THIS^.Ref_ExitConveyorAxis,
2                   lrVelocity := ConveyorParameters.LR_VELOCITY,
3                   lrAcceleration := ConveyorParameters.LR_ACCELERATION,
4                   lrDeceleration := ConveyorParameters.LR_DECELERATION,
5                   lrJerk := ConveyorParameters.LR_JERK,
6                   intDirection := intDirection,
7                   bVelocityReached => bVelocityReached,
8                   bError => bError,
9                   ErrorID => ErrorID);
10
11      IF bError AND ErrorID <> 0 THEN
12          sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language
        := 0);
13          sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.
        PW_ENTER_CONVEYOR_DRIVE_ERROR);
14
15          THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.
        PW_ENTER_CONVEYOR_DRIVE_ERROR;
16          THIS^.PW_Error.sErrorCauseMessage := sErrorCauseMessage;
17          THIS^.PW_Error.sErrorSolutionMessage := sErrorSolutionMessage;
18
19      END_IF
20
```

1.1.1.1.1.3.3.4 Method: ResetFaults

```
1       METHOD PUBLIC ResetFaults
2       VAR
3           bErrorCancelled : BOOL := TRUE;   // HMI confiramtion
4           sErrorCause : STRING;
5           sErrorSolution : STRING;
6       END_VAR
7       VAR_OUTPUT
8           bResetFaultDone : BOOL := FALSE;
9       END_VAR
10
```

```
1       IF THIS^.PW_Error.PW_ErrorID <> PW_ERROR_ID.PW_NO_ERROR THEN
2           sErrorCause := THIS^.PW_Error.sErrorCauseMessage;
3           sErrorSolution := THIS^.PW_Error.sErrorSolutionMessage;
4           IF bErrorCancelled THEN
5               THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
6               THIS^.PW_Error.sErrorCauseMessage := '';
7               THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
        PW_ERROR_ID.PW_NO_ERROR);
8           bResetFaultDone := TRUE;
9           END_IF
10          bResetFaultDone := FALSE;
11      ELSE
12          bResetFaultDone := TRUE;
13      END_IF
14
```

```
15
16
```

1.1.1.1.1.3.3.5 Method: StopDrive

```
METHOD PUBLIC StopDrive
VAR
    bBusy : BOOL;
    bError : BOOL;
    ErrorID : SMC_ERROR;
    sErrorCauseMessage : STRING;
    sErrorSolutionMessage : STRING;
END_VAR
VAR_OUTPUT
    bStopDone : BOOL;
END_VAR
```

```
CASE THIS^.Ref_ExitConveyorAxis.nAxisState OF
    4,5,6,7 : SUPER^.ExecuteStop(Ref_Axis := THIS^.Ref_ExitConveyorAxis,
                lrDeceleration := ConveyorParameters.LR_DECELERATION,
                lrJerk := ConveyorParameters.LR_JERK,
                bStatus => bStopDone,
                bBusy => bBusy,
                bError => bError,
                ErrorID => ErrorID);

    IF bError AND ErrorID <> 0 THEN
        sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
        sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR);

        THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ENTER_CONVEYOR_DRIVE_ERROR;
        THIS^.PW_Error.sErrorCauseMessage := sErrorCauseMessage;
        THIS^.PW_Error.sErrorSolutionMessage := sErrorSolutionMessage;
    END_IF
END_CASE
```

1.1.1.1.1.3.3.6 Method: ThrowError

```
METHOD PUBLIC ThrowError
VAR_INPUT
    ThrownError : PW_ERROR;
END_VAR
```

```
THIS^.PW_Error.PW_ErrorID := ThrownError.PW_ErrorID;
THIS^.PW_Error.sErrorCauseMessage := ThrownError.sErrorCauseMessage;
THIS^.PW_Error.sErrorSolutionMessage := ThrownError.sErrorSolutionMessage;
```

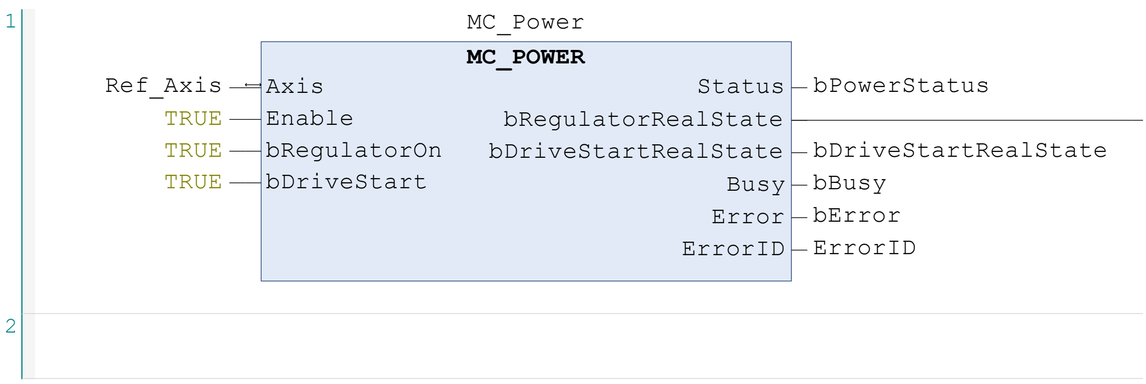
1.1.1.1.1.3.4 POU: CM_FilmAplicator

```
FUNCTION_BLOCK PUBLIC CM_FilmAplicator
VAR_INPUT
    Ref_AplicatorAxis : REFERENCE TO AXIS_REF_SM3 ;
END_VAR
VAR_OUTPUT
    HeightSensor : BOOL := TRUE ;
    PW_Error : PW_ERROR ;
END_VAR
```

1.1.1.1.1.3.4.1 Folder: Private Methods

1.1.1.1.1.3.4.1.1 Method: EnablePower

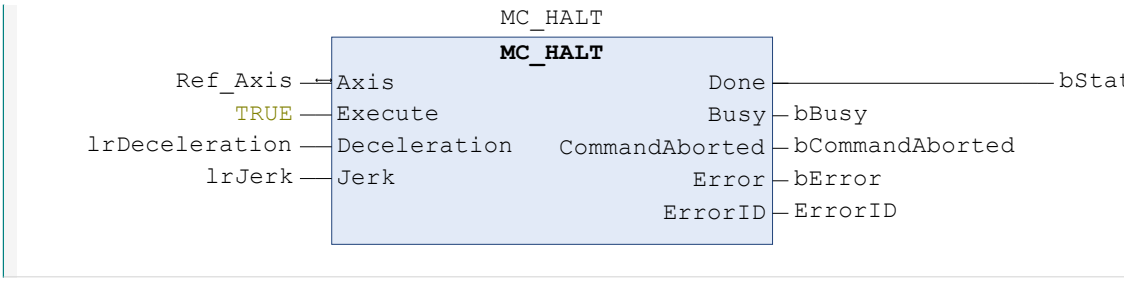
```
METHOD PRIVATE EnablePower
VAR_INPUT
    Ref_Axis : REFERENCE TO AXIS_REF_SM3 ;
END_VAR
VAR
    MC_Power : MC_Power ;
END_VAR
VAR_OUTPUT
    bPowerStatus : BOOL ;
    bRegulatorRealState : BOOL ;
    bDriveStartRealState : BOOL ;
    bBusy : BOOL ;
    bError : BOOL ;
    ErrorID : SMC_ERROR ;
END_VAR
```



```
-bRegulatorRealState
```

1.1.1.1.1.3.4.1.2 Method: ExecuteHalt

```
1   METHOD PRIVATE ExecuteHalt
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrDeceleration : LREAL;
5       lrJerk : LREAL;
6   END_VAR
7   VAR
8       MC_HALT : MC_HALT;
9   END_VAR
10  VAR_OUTPUT
11      bStatus : BOOL;
12      bBusy : BOOL;
13      bCommandAborted : BOOL;
14      bError : BOOL;
15      ErrorID : SMC_ERROR;
16  END_VAR
17
```

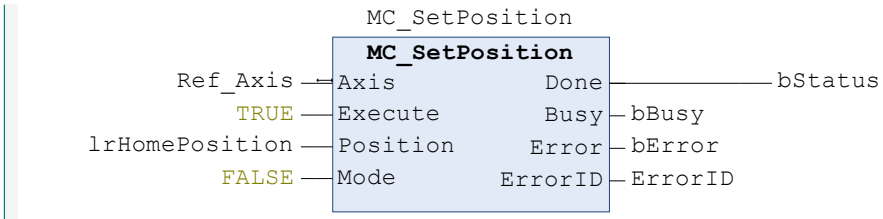


tus

1.1.1.1.1.3.4.1.3 Method: ExecuteHome

```
1   METHOD PRIVATE ExecuteHome
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrHomePosition : LREAL;
5   END_VAR
6   VAR
7       MC_SetPosition : MC_SetPosition;
8   END_VAR
9   VAR_OUTPUT
10      bStatus : BOOL;
11      bBusy : BOOL;
12      bError : BOOL;
13      ErrorID : SMC_ERROR;
14  END_VAR
15
```

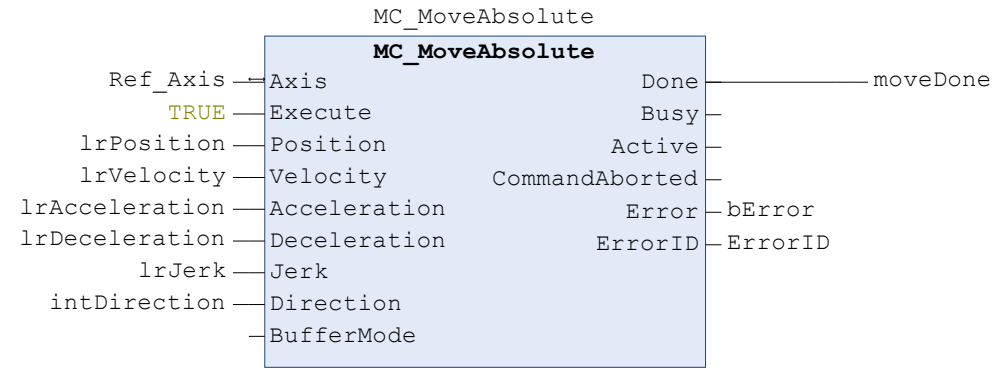
```
1
```



1.1.1.1.1.3.4.1.4 Method: ExecuteMoveAbs

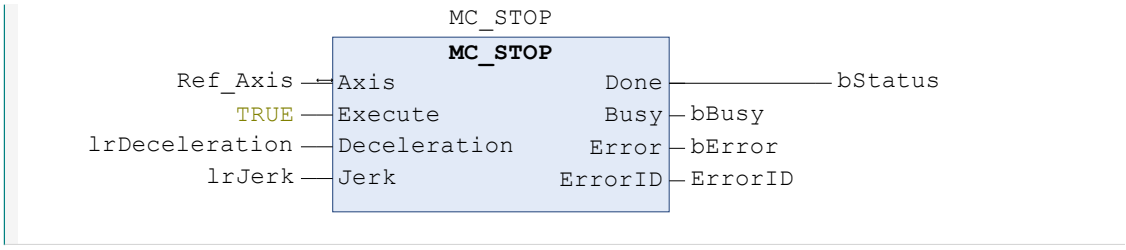
```
1   METHOD PRIVATE ExecuteMoveAbs : BOOL
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrPosition : LREAL;
5       lrVelocity : LREAL;
6       lrAcceleration : LREAL;
7       lrDeceleration : LREAL;
8       lrJerk : LREAL;
9       intDirection : INT;
10  END_VAR
11  VAR
12      MC_MoveAbsolute : MC_MoveAbsolute;
13  END_VAR
14  VAR_OUTPUT
15      moveDone : BOOL;
16      bError : BOOL;
17      ErrorID : SMC_ERROR;
18  END_VAR
19
```

1



1.1.1.1.1.3.4.1.5 Method: ExecuteStop

```
1   METHOD PRIVATE ExecuteStop
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrDeceleration : LREAL;
5       lrJerk : LREAL;
6   END_VAR
7   VAR
8       MC_STOP : MC_STOP;
9   END_VAR
10  VAR_OUTPUT
11      bStatus : BOOL;
12      bBusy : BOOL;
13      bError : BOOL;
14      ErrorID : SMC_ERROR;
15  END_VAR
16
```



1.1.1.1.1.3.4.2 Method: EnableDrivePower

```
METHOD PUBLIC EnableDrivePower : BOOL
VAR
    bError : BOOL;
    ErrorID : SMC_ERROR;
END_VAR
VAR_OUTPUT
    bPowerStatus : BOOL;
END_VAR


```

```
CASE THIS^.Ref_AplicatorAxis.nAxisState OF
    0,1,2 : EnablePower(Ref_Axis := THIS^.Ref_AplicatorAxis,
                        bPowerStatus => bPowerStatus,
                        bError => bError,
                        ErrorID => ErrorID);
ELSE
    bPowerStatus := TRUE;
END_CASE
IF bError AND ErrorID <> 0 THEN
    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_APLICATOR_POWER_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
    THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_APLICATOR_POWER_ERROR);
END_IF


```

1.1.1.1.1.3.4.3 Method: ExecuteDriveHome

```
METHOD PUBLIC ExecuteDriveHome
VAR
    bBusy : BOOL;
    bError : BOOL;
    ErrorID : SMC_ERROR;
    sErrorCauseMessage : STRING;
    sErrorSolutionMessage : STRING;
END_VAR
VAR_OUTPUT
    bHomeDone : BOOL;
END_VAR

```

```
ExecuteHome(Ref_Axis := THIS^.Ref_AplicatorAxis,
            lrHomePosition := AplicatorParameters.LR_HOME_POSITION,
```

```
                    bStatus  =>  bHomeDone ,
                    bBusy  =>  bBusy ,
                    bError  =>  bError ,
                    ErrorID  =>  ErrorID ) ;

    IF  bError  AND  ErrorID  <>  0  THEN
        sErrorCauseMessage  :=  SMC_ErrorString ( ErrorId  :=  ErrorID ,  Language
    :=  0 ) ;
        sErrorSolutionMessage  :=  PW_getErrorSolution ( PW_ERROR_ID .
    PW_APLICATOR_HOME_ERROR ) ;

        THIS ^ . PW_Error . PW_ErrorID  :=  PW_ERROR_ID . PW_APLICATOR_HOME_ERROR ;
        THIS ^ . PW_Error . sErrorCauseMessage  :=  sErrorCauseMessage ;
        THIS ^ . PW_Error . sErrorSolutionMessage  :=  sErrorSolutionMessage ;

    END_IF
```

1.1.1.1.1.3.4.4 Method: HaltDrive

```
    METHOD  PUBLIC  HaltDrive
    VAR
        bError  :  BOOL ;
        ErrorID  :  SMC_ERROR ;
        sErrorCauseMessage  :  STRING ;
        sErrorSolutionMessage  :  STRING ;
    END_VAR
    VAR_OUTPUT
        bHaltDone  :  BOOL ;
        bBusy  :  BOOL ;
    END_VAR
```

```
    ExecuteHalt ( Ref_Axis  :=  THIS ^ . Ref_AplicatorAxis ,
                lrDeceleration  :=  AplicatorParameters . LR_DECELERATION ,
                lrJerk  :=  AplicatorParameters . LR_JERK ,
                bStatus  =>  bHaltDone ,
                bBusy  =>  bBusy ,
                bError  =>  bError ,
                ErrorID  =>  ErrorID ) ;

    IF  bError  AND  ErrorID  <>  0  THEN
        sErrorCauseMessage  :=  SMC_ErrorString ( ErrorId  :=  ErrorID ,  Language
    :=  0 ) ;
        sErrorSolutionMessage  :=  PW_getErrorSolution ( PW_ERROR_ID .
    PW_APLICATOR_HALT_ERROR ) ;

        THIS ^ . PW_Error . PW_ErrorID  :=  PW_ERROR_ID . PW_APLICATOR_HALT_ERROR ;
        THIS ^ . PW_Error . sErrorCauseMessage  :=  sErrorCauseMessage ;
        THIS ^ . PW_Error . sErrorSolutionMessage  :=  sErrorSolutionMessage ;

    END_IF
```

1.1.1.1.1.3.4.5 Method: MoveAbsolute

```
METHOD PUBLIC MoveAbsolute
VAR_INPUT
    lrPosition : LREAL;
END_VAR
VAR
    bError : BOOL;
    ErrorID : SMC_ERROR;
    sErrorCauseMessage : STRING;
    sErrorSolutionMessage : STRING;
END_VAR
VAR_OUTPUT
    bMoveDone : BOOL;
END_VAR

```

```
ExecuteMoveAbs(Ref_Axis := THIS^.Ref_AplicatorAxis,
               lrPosition := lrPosition,
               lrVelocity := AplicatorParameters.LR_VELOCITY,
               lrAcceleration := AplicatorParameters.LR_ACCELERATION,
               lrDeceleration := AplicatorParameters.LR_DECELERATION,
               lrJerk := AplicatorParameters.LR_JERK,
               intDirection := MC_DIRECTION.shortest,
               moveDone => bMoveDone,
               bError => bError,
               ErrorID => ErrorID);

IF bError AND ErrorID <> 0 THEN
    sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
    sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_APLICATOR_MOVE_ERROR);

    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_APLICATOR_MOVE_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := sErrorCauseMessage;
    THIS^.PW_Error.sErrorSolutionMessage := sErrorSolutionMessage;

END_IF

```

1.1.1.1.1.3.4.6 Method: PositionToPalletHeight

```
METHOD PUBLIC PositionToPalletHeight
VAR_INPUT
    usintCurrentLayerNumber : USINT;
    lrPalletHeight : LREAL;
END_VAR
VAR
    ThrownError : PW_ERROR;
    bAplicatorHaltDone : BOOL := FALSE;
    bError : BOOL;
    ErrorID : SMC_ERROR;
END_VAR
```

```
VAR_OUTPUT
    bMoveDone : BOOL;
END_VAR
```

```
ExecuteMoveAbs(Ref_Axis := THIS^.Ref_AplicatorAxis,
               lrPosition := lrPalletHeight,
               lrVelocity := AplicatorParameters.LR_SLOW_VELOCITY,
               lrAcceleration := AplicatorParameters.LR_SLOW_ACCELERATION,
               lrDeceleration := AplicatorParameters.LR_DECELERATION,
               lrJerk := AplicatorParameters.LR_JERK,
               intDirection := MC_DIRECTION.shortest,
               moveDone => bMoveDone,
               bError => bError,
               ErrorID => ErrorID);
IF bError AND ErrorID <> 0 THEN
    ThrownError.PW_ErrorID := PW_ERROR_ID.PW_APLICATOR_MOVE_ERROR;
    ThrownError.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
    ThrownError.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_APLICATOR_MOVE_ERROR);
    ThrowError(ThrownError := ThrownError);
END_IF
```

1.1.1.1.1.3.4.7 Method: ResetFaults

```
METHOD PUBLIC ResetFaults
VAR
    bErrorCancelled : BOOL := TRUE;   // HMI confiramtion
    sErrorCause : STRING;
    sErrorSolution : STRING;
END_VAR
VAR_OUTPUT
    bResetFaultDone : BOOL := FALSE;
END_VAR
```

```
IF THIS^.PW_Error.PW_ErrorID <> PW_ERROR_ID.PW_NO_ERROR THEN
    sErrorCause := THIS^.PW_Error.sErrorCauseMessage;
    sErrorSolution := THIS^.PW_Error.sErrorSolutionMessage;
    IF bErrorCancelled THEN
        THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
        THIS^.PW_Error.sErrorCauseMessage := '';
        THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_NO_ERROR);
    bResetFaultDone := TRUE;
    END_IF
    bResetFaultDone := FALSE;
ELSE
```

```
12          bResetFaultDone  :=  TRUE ;
13      END_IF
14
15
16
```

1.1.1.1.1.3.4.8 Method: StopDrive

```
1       METHOD  PUBLIC  StopDrive
2       VAR
3           bBusy  :  BOOL ;
4           bError  :  BOOL ;
5           ErrorID  :  SMC_ERROR ;
6           sErrorCauseMessage  :  STRING ;
7           sErrorSolutionMessage  :  STRING ;
8       END_VAR
9       VAR_OUTPUT
10          bStopDone :  BOOL ;
11      END_VAR
12
```

```
1       CASE  THIS ^ . Ref_AplicatorAxis . nAxisState  OF
2           4 , 5 , 6 , 7  :  ExecuteStop ( Ref_Axis  :=  THIS ^ . Ref_AplicatorAxis ,
3                       lrDeceleration  :=  AplicatorParameters . LR_DECELERATION ,
4                       lrJerk  :=  AplicatorParameters . LR_JERK ,
5                       bStatus  =>  bStopDone ,
6                       bBusy  =>  bBusy ,
7                       bError  =>  bError ,
8                       ErrorID  =>  ErrorID ) ;
9       IF  bError  AND  ErrorID  <>  0  THEN
10          sErrorCauseMessage  :=  SMC_ErrorString ( ErrorId  :=  ErrorID ,  Language
        :=  0 ) ;
11          sErrorSolutionMessage  :=  PW_getErrorSolution ( PW_ERROR_ID .
        PW_ENTER_CONVEYOR_DRIVE_ERROR ) ;
12
13          THIS ^ . PW_Error . PW_ErrorID  :=  PW_ERROR_ID .
        PW_ENTER_CONVEYOR_DRIVE_ERROR ;
14          THIS ^ . PW_Error . sErrorCauseMessage  :=  sErrorCauseMessage ;
15          THIS ^ . PW_Error . sErrorSolutionMessage  :=  sErrorSolutionMessage ;
16      END_IF
17
18
19      END_CASE
20
```

1.1.1.1.1.3.4.9 Method: ThrowError

```
METHOD PUBLIC ThrowError
VAR_INPUT
    ThrownError : PW_ERROR;
END_VAR

```

```
THIS^.PW_Error.PW_ErrorID := ThrownError.PW_ErrorID;
THIS^.PW_Error.sErrorCauseMessage := ThrownError.sErrorCauseMessage;
THIS^.PW_Error.sErrorSolutionMessage := ThrownError.sErrorSolutionMessage;

```

1.1.1.1.1.3.5 POU: CM_FilmClamp

```
FUNCTION_BLOCK PUBLIC CM_FilmClamp
VAR
    bExtensionValve : BOOL := FALSE;              {* Pneumatic electro-valve 3/2 DC, normmally close, stable position BY spring.

Pneumatic actuator with pull spring, *}
    bExtensionSensor : BOOL := FALSE;             // Piston position sensor, TRUE -> extended, jaws closed.
    bRetractionSensor : BOOL := TRUE;             // Piston position sensor, TRUE -> retracted, jaws opened.
    bPresenceSensor : BOOL := FALSE;          // Film detection sensor, TRUE -> film detected. Film does not exist if Film detection sensor = FALSE AND bExtensionSensor = TRUE.
    bInitConfirmation : BOOL := TRUE;         // HMI operator init confirmation.
END_VAR
VAR_OUTPUT
    PW_Error : PW_ERROR;
END_VAR

```

1.1.1.1.1.3.5.1 Method: ClampCorrect

```
METHOD PRIVATE ClampCorrect

```

```
IF THIS^.bExtensionValve AND NOT THIS^.bPresenceSensor THEN
    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_FILM_NOT_DETECTED_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := '';
    THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_FILM_NOT_DETECTED_ERROR);;

    // bInitConfirmation := FALSE;
```

```
7       ELSIF THIS^.bExtensionValve AND NOT THIS^.bExtensionSensor THEN
8           THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_CLAMP_RELEASE_ERROR;
9           THIS^.PW_Error.sErrorCauseMessage := '';
10          THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
        PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_CLAMP_RELEASE_ERROR);;
11
12      ELSIF NOT THIS^.bExtensionValve AND NOT THIS^.bRetractionSensor THEN
13          THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_CLAMP_CLOSE_ERROR;
14          THIS^.PW_Error.sErrorCauseMessage := '';
15          THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
        PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_CLAMP_CLOSE_ERROR);;
16
17      ELSIF (THIS^.bRetractionSensor AND THIS^.bExtensionSensor) XOR
18            (NOT THIS^.bRetractionSensor AND NOT THIS^.bExtensionSensor) THEN
19            IF THIS^.bExtensionValve THEN
20                THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.
        PW_CLAMP_RELEASE_ERROR;
21                THIS^.PW_Error.sErrorCauseMessage := '';
22                THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution
        (PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_CLAMP_CLOSE_ERROR);;
23          END_IF
24          IF NOT THIS^.bExtensionValve THEN
25              THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.
        PW_CLAMP_RELEASE_ERROR;
26              THIS^.PW_Error.sErrorCauseMessage := '';
27              THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution
        (PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_CLAMP_CLOSE_ERROR);;
28          END_IF
29      ELSE
30          THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
31          THIS^.PW_Error.sErrorCauseMessage := '';
32          THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
        PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_NO_ERROR);;
33      END_IF
34
35
```

1.1.1.1.1.3.5.2 Method: CloseClamp

```
1       METHOD PUBLIC CloseClamp
2       VAR_OUTPUT
3           bCloseDone : BOOL := FALSE;
4       END_VAR
5       VAR
6           bCloseBusy : BOOL := FALSE;
7       END_VAR
8
```

```
1       ClampCorrect();
2       IF THIS^.PW_Error.PW_ErrorID = PW_ERROR_ID.PW_NO_ERROR THEN
3           bCloseDone := THIS^.bExtensionSensor;
4           IF NOT bCloseDone AND (THIS^.bRetractionSensor XOR bCloseBusy)
        THEN
5               bCloseBusy := TRUE;
6               THIS^.bExtensionValve := TRUE;
```

```
            // For simulation testing
            THIS^.bRetractionSensor := FALSE;
            // For simulation testing
            THIS^.bExtensionSensor := TRUE;
            // For simulation testing
            THIS^.bPresenceSensor := TRUE;
        END_IF
        bCloseDone := THIS^.bExtensionSensor;
        IF bCloseDone THEN
            bCloseBusy := FALSE;
        END_IF
    ELSE bCloseDone := FALSE;
    END_IF

```

1.1.1.1.1.3.5.3 Method: InitializeClamp

```
    METHOD PUBLIC InitializeClamp
    VAR_OUTPUT
        bInitializationDone : BOOL;
    END_VAR

```

```
    IF THIS^.bInitConfirmation THEN
            CloseClamp(bCloseDone => bInitializationDone);
    END_IF

```

1.1.1.1.1.3.5.4 Method: ProvideFilm

```
    METHOD PUBLIC ProvideFilm
    VAR_OUTPUT
        bFilmProvided : BOOL;
    END_VAR

```

```
    bFilmProvided := true;

```

1.1.1.1.1.3.5.5 Method: ReleaseClamp

```
    METHOD PUBLIC ReleaseClamp
    VAR_OUTPUT
        bReleaseDone : BOOL;
    END_VAR
    VAR
        bReleaseBusy : BOOL;
    END_VAR

```

```
    ClampCorrect();
    IF THIS^.PW_Error.PW_ErrorID = PW_ERROR_ID.PW_NO_ERROR THEN
        bReleaseDone := THIS^.bRetractionSensor;
        IF NOT bReleaseDone AND (THIS^.bExtensionSensor XOR bReleaseBusy)
```

```
    THEN
        bReleaseBusy := TRUE;
        THIS^.bExtensionValve := FALSE;
        // For simulation testing
        THIS^.bRetractionSensor := TRUE;
        // For simulation testing
        THIS^.bExtensionSensor := FALSE;
        // For simulation testing
        THIS^.bPresenceSensor := FALSE;
    END_IF

    bReleaseDone := THIS^.bRetractionSensor;
    IF bReleaseDone THEN
        bReleaseBusy := FALSE;
    END_IF
ELSE RETURN;
END_IF

```

1.1.1.1.1.3.5.6 Method: ResetFaults

```
METHOD PUBLIC ResetFaults
VAR
    bErrorCancelled : BOOL := TRUE;   // HMI confiramtion
    bErrorOccured : BOOL := FALSE;        // HMI confiramtion
    sErrorCause : STRING;
    sErrorSolution : STRING;
END_VAR
VAR_OUTPUT
    bResetFaultDone : BOOL := FALSE;
END_VAR

```

```
IF THIS^.PW_Error.PW_ErrorID <> PW_ERROR_ID.PW_NO_ERROR THEN
    bErrorOccured := TRUE;
    sErrorCause := THIS^.PW_Error.sErrorCauseMessage;
    sErrorSolution := THIS^.PW_Error.sErrorSolutionMessage;
    IF bErrorCancelled THEN
        THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
        THIS^.PW_Error.sErrorCauseMessage := '';
        THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
PW_ERROR_ID.PW_NO_ERROR);
    bResetFaultDone := TRUE;
    END_IF
    bResetFaultDone := FALSE;
ELSE
    bResetFaultDone := TRUE;
END_IF

```

1.1.1.1.1.3.5.7 Method: ThrowError

```
METHOD PUBLIC ThrowError
VAR_INPUT
    ThrownError : PW_ERROR;
END_VAR

```

```
THIS^.PW_Error.PW_ErrorID := ThrownError.PW_ErrorID;
THIS^.PW_Error.sErrorCauseMessage := ThrownError.sErrorCauseMessage;
THIS^.PW_Error.sErrorSolutionMessage := ThrownError.sErrorSolutionMessage;

```

1.1.1.1.1.3.6 POU: CM_FilmSticker

```
FUNCTION_BLOCK PUBLIC CM_FilmSticker
VAR
    bExtensionValve : BOOL := FALSE;              {* Pneumatic electro-valve 3/2 DC, normmally close, stable position BY spring.

Pneumatic actuator with pull spring, *}
    bExtensionSensor : BOOL := FALSE;                 // Piston position sensor, TRUE -> extended, film attached.
    bRetractionSensor : BOOL := TRUE;                 // Piston position sensor, TRUE -> retracted.
END_VAR
VAR_OUTPUT
    PW_Error : PW_ERROR;
END_VAR

```

1.1.1.1.1.3.6.1 Method: AttachSticker

```
METHOD PUBLIC AttachSticker
VAR_OUTPUT
    bAttachDone : BOOL;
END_VAR
VAR
    bAttachBusy : BOOL;
END_VAR

```

```
StickerCorrect();
IF THIS^.PW_Error.PW_ErrorID = PW_ERROR_ID.PW_NO_ERROR THEN
    bAttachDone := THIS^.bExtensionSensor;
    IF NOT bAttachDone AND (THIS^.bRetractionSensor XOR bAttachBusy) THEN
        bAttachBusy := TRUE;
        THIS^.bExtensionValve := TRUE;
```

```
            // For simulation testing
            THIS^.bRetractionSensor := FALSE;
            // For simulation testing
            THIS^.bExtensionSensor := TRUE;
        END_IF

        bAttachDone := THIS^.bExtensionSensor;
        IF bAttachDone THEN
            bAttachBusy := FALSE;
        END_IF
    ELSE RETURN;
    END_IF

```

1.1.1.1.1.3.6.2 Method: DetachSticker

```
    METHOD PUBLIC DetachSticker
    VAR_OUTPUT
        bDetachDone : BOOL := FALSE;
    END_VAR
    VAR
        bDetachBusy : BOOL := FALSE;
    END_VAR

```

```
    StickerCorrect();
    IF THIS^.PW_Error.PW_ErrorID = PW_ERROR_ID.PW_NO_ERROR THEN
        bDetachDone := THIS^.bRetractionSensor;
        IF NOT bDetachDone AND (THIS^.bExtensionSensor XOR bDetachBusy)
    THEN
            bDetachBusy := TRUE;
            THIS^.bExtensionValve := FALSE;
            // For simulation testing
            THIS^.bRetractionSensor := TRUE;
            // For simulation testing
            THIS^.bExtensionSensor := FALSE;
        END_IF
        bDetachDone := THIS^.bRetractionSensor;
        IF bDetachDone THEN
            bDetachBusy := FALSE;
        END_IF
    ELSE RETURN;
    END_IF

```

1.1.1.1.1.3.6.3 Method: InitializeSticker

```
METHOD PUBLIC InitializeSticker
VAR_OUTPUT
    bInitializationDone : BOOL;
END_VAR

```

```
DetachSticker(bDetachDone => bInitializationDone);

```

1.1.1.1.1.3.6.4 Method: ResetFaults

```
METHOD PUBLIC ResetFaults
VAR
    bErrorCancelled : BOOL := TRUE;   // HMI confiramtion
    bErrorOccured : BOOL := FALSE;
    sErrorCause : STRING;
    sErrorSolution : STRING;
END_VAR
VAR_OUTPUT
    bResetFaultDone : BOOL := FALSE;
END_VAR

```

```
IF THIS^.PW_Error.PW_ErrorID <> PW_ERROR_ID.PW_NO_ERROR THEN
    bErrorOccured := TRUE;
    sErrorCause := THIS^.PW_Error.sErrorCauseMessage;
    sErrorSolution := THIS^.PW_Error.sErrorSolutionMessage;
    IF bErrorCancelled THEN
        THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
        THIS^.PW_Error.sErrorCauseMessage := '';
        THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
PW_ERROR_ID.PW_NO_ERROR);
    bResetFaultDone := TRUE;
    END_IF
    bResetFaultDone := FALSE;
ELSE
    bResetFaultDone := TRUE;
END_IF

```

1.1.1.1.1.3.6.5 Method: StickerCorrect

```
METHOD PRIVATE StickerCorrect

```

```
IF THIS^.bExtensionValve AND NOT THIS^.bExtensionSensor THEN
    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_STICKER_ATTACH_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := '';
    THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_STICKER_ATTACH_ERROR);;
ELSIF NOT THIS^.bExtensionValve AND NOT THIS^.bRetractionSensor THEN
```

```
6           THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_STICKER_DETACH_ERROR;
7           THIS^.PW_Error.sErrorCauseMessage := '';
8           THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
        PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_STICKER_DETACH_ERROR);;
9
10      ELSIF (THIS^.bRetractionSensor AND THIS^.bExtensionSensor) XOR
11            (NOT THIS^.bRetractionSensor AND NOT THIS^.bExtensionSensor) THEN
12              IF THIS^.bExtensionValve THEN
13                  THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.
        PW_STICKER_DETACH_ERROR;
14                  THIS^.PW_Error.sErrorCauseMessage := '';
15                  THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution
        (PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_STICKER_DETACH_ERROR);;
16              END_IF
17              IF NOT THIS^.bExtensionValve THEN
18                  THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.
        PW_STICKER_DETACH_ERROR;
19                  THIS^.PW_Error.sErrorCauseMessage := '';
20                  THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution
        (PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_STICKER_DETACH_ERROR);;
21              END_IF
22      ELSE
23          THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
24          THIS^.PW_Error.sErrorCauseMessage := '';
25          THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
        PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_NO_ERROR);
26      END_IF
27
```

1.1.1.1.1.3.6.6 Method: ThrowError

```
1       METHOD PUBLIC ThrowError
2       VAR_INPUT
3           ThrownError : PW_ERROR;
4       END_VAR
5
```

```
1       THIS^.PW_Error.PW_ErrorID := ThrownError.PW_ErrorID;
2       THIS^.PW_Error.sErrorCauseMessage := ThrownError.sErrorCauseMessage;
3       THIS^.PW_Error.sErrorSolutionMessage := ThrownError.
        sErrorSolutionMessage;
4
```

1.1.1.1.1.3.7 POU: CM_PalletProtector

```
1       FUNCTION_BLOCK PUBLIC CM_PalletProtector
2       VAR
3           bExtensionValve : BOOL := FALSE;            {* Pneumatic
        electro-valve 3/2 DC, normmally close, stable position BY spring.
4
        Pneumatic actuator with pull spring, *}
5           bExtensionSensor : BOOL := FALSE;           // Piston position
        sensor, TRUE -> extended, protector vertically.
6           bRetractionSensor : BOOL := TRUE;                       // Piston
```

```
    position sensor, TRUE -> retracted, protector horizontally.
END_VAR
VAR_OUTPUT
    PW_Error : PW_ERROR;
END_VAR
```

1.1.1.1.1.3.7.1 Method: ExtendProtector

```
METHOD PUBLIC ExtendProtector
VAR_OUTPUT
    bExtendDone : BOOL;
END_VAR
VAR
    bCloseBusy : BOOL;
END_VAR
```

```
ProtectorCorrect();
IF THIS^.PW_Error.PW_ErrorID = PW_ERROR_ID.PW_NO_ERROR THEN
    bExtendDone := THIS^.bExtensionSensor;
    IF NOT bExtendDone AND (THIS^.bRetractionSensor XOR bCloseBusy)
THEN
        bCloseBusy := TRUE;
        THIS^.bExtensionValve := TRUE;
        // For simulation testing
        THIS^.bRetractionSensor := FALSE;
        // For simulation testing
        THIS^.bExtensionSensor := TRUE;
    END_IF
    bExtendDone := THIS^.bExtensionSensor;
    IF bExtendDone THEN
        bCloseBusy := FALSE;
    END_IF
ELSE RETURN;
END_IF
```

1.1.1.1.1.3.7.2 Method: InitializeProtector

```
METHOD PUBLIC InitializeProtector
VAR_OUTPUT
    bInitializationDone : BOOL;
END_VAR
```

```
ReleaseProtector(bReleaseDone => bInitializationDone);
```

1.1.1.1.1.3.7.3 Method: ProtectorCorrect

```
METHOD PRIVATE ProtectorCorrect

```

```
IF THIS^.bExtensionValve AND NOT THIS^.bExtensionSensor THEN
    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_PROTECTOR_EXTEND_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := '';
    THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_PROTECTOR_EXTEND_ERROR);;
ELSIF NOT THIS^.bExtensionValve AND NOT THIS^.bRetractionSensor THEN
    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_PROTECTOR_RELEASE_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := '';
    THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_PROTECTOR_RELEASE_ERROR);;
ELSIF (THIS^.bRetractionSensor AND THIS^.bExtensionSensor) XOR
      (NOT THIS^.bRetractionSensor AND NOT THIS^.bExtensionSensor) THEN
        IF THIS^.bExtensionValve THEN
            THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_PROTECTOR_EXTEND_ERROR;
            THIS^.PW_Error.sErrorCauseMessage := '';
            THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_PROTECTOR_EXTEND_ERROR);;
        END_IF
        IF NOT THIS^.bExtensionValve THEN
            THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_PROTECTOR_RELEASE_ERROR;
            THIS^.PW_Error.sErrorCauseMessage := '';
            THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_PROTECTOR_RELEASE_ERROR);;
        END_IF
ELSE
    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := '';
    THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PALLET_WRAPPER_ERROR := PW_ERROR_ID.PW_NO_ERROR);
END_IF

```

1.1.1.1.1.3.7.4 Method: ReleaseProtector

```
METHOD PUBLIC ReleaseProtector
VAR_OUTPUT
    bReleaseDone : BOOL := FALSE;
END_VAR
VAR
    bReleaseBusy : BOOL := FALSE;
END_VAR

```

```
ProtectorCorrect();
IF THIS^.PW_Error.PW_ErrorID = PW_ERROR_ID.PW_NO_ERROR THEN
    bReleaseDone := THIS^.bRetractionSensor;
```

```
4           IF NOT bReleaseDone AND (THIS^.bExtensionSensor XOR bReleaseBusy)
    THEN
5               bReleaseBusy := TRUE;
6               THIS^.bExtensionValve := FALSE;
7               // For simulation testing
8               THIS^.bRetractionSensor := TRUE;
9               // For simulation testing
10              THIS^.bExtensionSensor := FALSE;
11          END_IF
12
13          bReleaseDone := THIS^.bRetractionSensor;
14          IF bReleaseDone THEN
15              bReleaseBusy := FALSE;
16          END_IF
17      ELSE RETURN;
18      END_IF
19
```

1.1.1.1.1.3.7.5 Method: ResetFaults

```
1       METHOD PUBLIC ResetFaults
2       VAR
3           bErrorCancelled : BOOL := TRUE;   // HMI confiramtion
4           bErrorOccured : BOOL := FALSE;
5           sErrorCause : STRING;
6           sErrorSolution : STRING;
7       END_VAR
8       VAR_OUTPUT
9           bResetFaultDone : BOOL := FALSE;
10      END_VAR
11
```

```
1       IF THIS^.PW_Error.PW_ErrorID <> PW_ERROR_ID.PW_NO_ERROR THEN
2           bErrorOccured := TRUE;
3           sErrorCause := THIS^.PW_Error.sErrorCauseMessage;
4           sErrorSolution := THIS^.PW_Error.sErrorSolutionMessage;
5           IF bErrorCancelled THEN
6               THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
7               THIS^.PW_Error.sErrorCauseMessage := '';
8               THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
    PW_ERROR_ID.PW_NO_ERROR);
9           bResetFaultDone := TRUE;
10          END_IF
11          bResetFaultDone := FALSE;
12      ELSE
13          bResetFaultDone := TRUE;
14      END_IF
15
```

1.1.1.1.1.3.7.6 Method: ThrowError

```
1       METHOD PUBLIC ThrowError
2       VAR_INPUT
3           ThrownError : PW_ERROR;
4       END_VAR
5
```

```
1       THIS^.PW_Error.PW_ErrorID := ThrownError.PW_ErrorID;
2       THIS^.PW_Error.sErrorCauseMessage := ThrownError.sErrorCauseMessage;
3       THIS^.PW_Error.sErrorSolutionMessage := ThrownError.
        sErrorSolutionMessage;
4
```

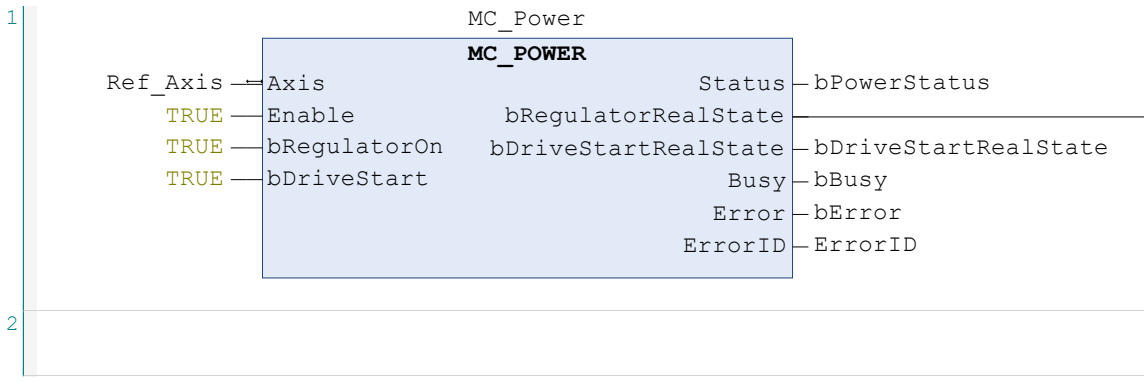
1.1.1.1.1.3.8 POU: CM_PalletRotator

```
1       FUNCTION_BLOCK PUBLIC CM_PalletRotator
2       VAR_INPUT
3           Ref_RotatorAxis : REFERENCE TO AXIS_REF_SM3;
4       END_VAR
5       VAR_OUTPUT
6           PW_Error : PW_ERROR;
7       END_VAR
8
```

1.1.1.1.1.3.8.1 Folder: Private Methods

1.1.1.1.1.3.8.1.1 Method: EnablePower

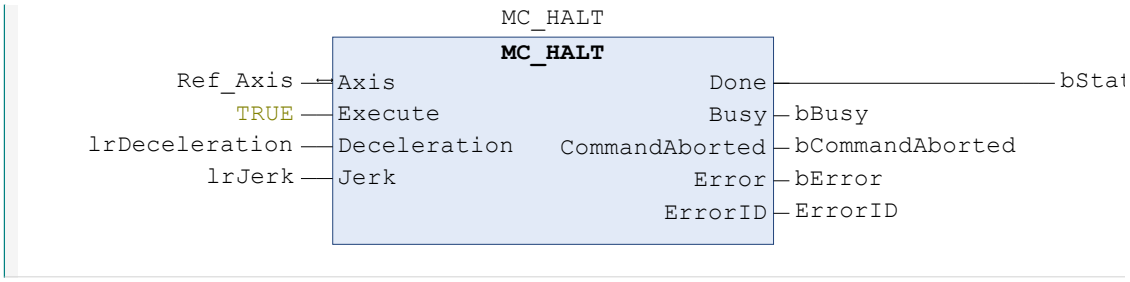
```
1       METHOD PRIVATE EnablePower
2       VAR_INPUT
3           Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       END_VAR
5       VAR
6           MC_Power : MC_Power;
7       END_VAR
8       VAR_OUTPUT
9           bPowerStatus : BOOL;
10          bRegulatorRealState : BOOL;
11          bDriveStartRealState : BOOL;
12          bBusy : BOOL;
13          bError : BOOL;
14          ErrorID : SMC_ERROR;
15      END_VAR
16
```


1
MC_Power
MC_POWER
Ref_Axis
Axis
Status
bPowerStatus
TRUE
Enable
bRegulatorRealState
TRUE
bRegulatorOn
bDriveStartRealState
bDriveStartRealState
TRUE
bDriveStart
Busy
bBusy
Error
bError
ErrorID
ErrorID
2

```
-bRegulatorRealState
```

1.1.1.1.1.3.8.1.2 Method: ExecuteHalt

```
1   METHOD PRIVATE ExecuteHalt
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrDeceleration : LREAL;
5       lrJerk : LREAL;
6   END_VAR
7   VAR
8       MC_HALT : MC_HALT;
9   END_VAR
10  VAR_OUTPUT
11      bStatus : BOOL;
12      bBusy : BOOL;
13      bCommandAborted : BOOL;
14      bError : BOOL;
15      ErrorID : SMC_ERROR;
16  END_VAR
17
```

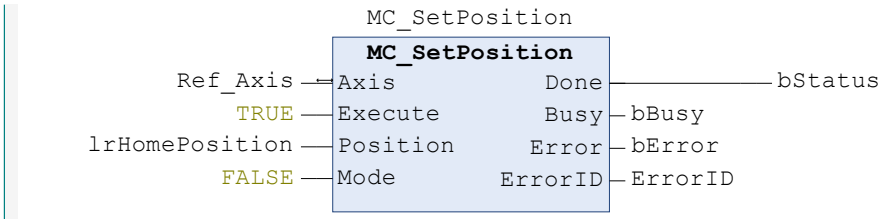


tus

1.1.1.1.1.3.8.1.3 Method: ExecuteHome

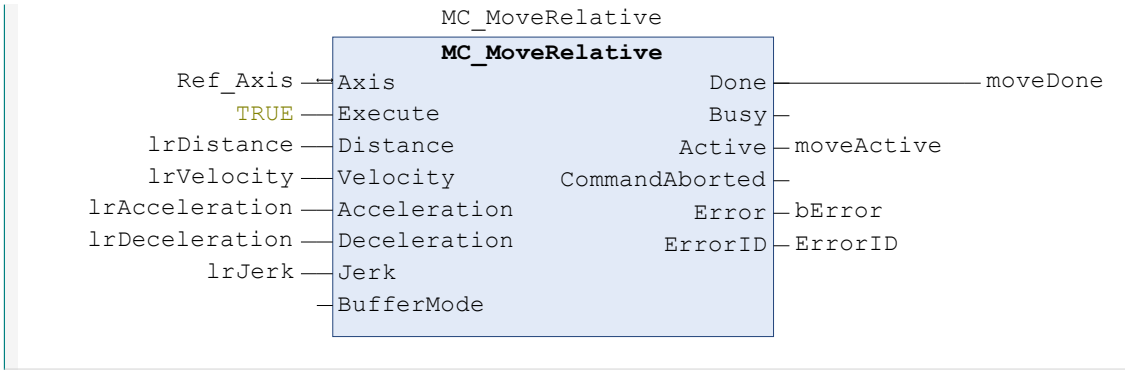
```
1   METHOD PRIVATE ExecuteHome
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrHomePosition : LREAL;
5   END_VAR
6   VAR
7       MC_SetPosition : MC_SetPosition;
8   END_VAR
9   VAR_OUTPUT
10      bStatus : BOOL;
11      bBusy : BOOL;
12      bError : BOOL;
13      ErrorID : SMC_ERROR;
14  END_VAR
15
```

```
1
```



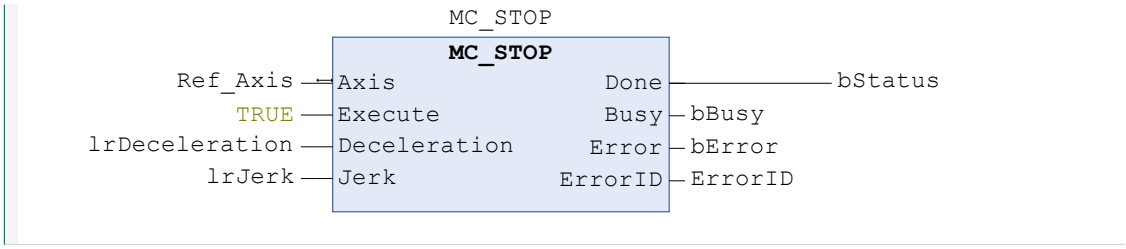
1.1.1.1.1.3.8.1.4 Method: ExecuteMoveRel

```
1   METHOD PRIVATE ExecuteMoveRel
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrDistance : LREAL;
5       lrVelocity : LREAL;
6       lrAcceleration : LREAL;
7       lrDeceleration : LREAL;
8       lrJerk : LREAL;
9   END_VAR
10  VAR
11      MC_MoveRelative : MC_MoveRelative;
12
13  END_VAR
14  VAR_OUTPUT
15      moveDone : BOOL;
16      moveActive : BOOL;
17      bError : BOOL;
18      ErrorID : SMC_ERROR;
19  END_VAR
20
```



1.1.1.1.1.3.8.1.5 Method: ExecuteStop

```
1   METHOD PRIVATE ExecuteStop
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrDeceleration : LREAL;
5       lrJerk : LREAL;
6   END_VAR
7   VAR
8       MC_STOP : MC_STOP;
9   END_VAR
10  VAR_OUTPUT
11      bStatus : BOOL;
12      bBusy : BOOL;
13      bError : BOOL;
14      ErrorID : SMC_ERROR;
15  END_VAR
16
```



1.1.1.1.1.3.8.2 Method: EnableDrivePower

```
1   METHOD PUBLIC EnableDrivePower
2   VAR
3       bErrorPRA : BOOL;
4       ErrorIDPRA : SMC_ERROR;
5   END_VAR
6   VAR_OUTPUT
7       bPowerStatusPRA : BOOL;
8       bRegulatorRealStatePRA : BOOL;
9       bDriveStartRealStatePRA : BOOL;
10  END_VAR
11
```

```
1   CASE THIS^.Ref_RotatorAxis.nAxisState OF
2       0,1,2 : EnablePower(Ref_Axis := THIS^.Ref_RotatorAxis,
3                           bPowerStatus => bPowerStatusPRA,
4                           bRegulatorRealState => bRegulatorRealStatePRA,
5                           bDriveStartRealState => bDriveStartRealStatePRA,
6                           bError => bErrorPRA,
7                           ErrorID => ErrorIDPRA);
8   ELSE
9       bPowerStatusPRA := TRUE;
10  END_CASE
11  IF bErrorPRA AND ErrorIDPRA <> 0 THEN
12      THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ROTATOR_POWER_ERROR;
13      THIS^.PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId :=
    ErrorIDPRA, Language := 0);
14      THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
    PW_ERROR_ID.PW_ROTATOR_POWER_ERROR);
15  END_IF
16
17
```

1.1.1.1.1.3.8.3 Method: ExecuteDriveHome

```
1   METHOD PUBLIC ExecuteDriveHome
2   VAR
3       bErrorPRA : BOOL;
4       ErrorIDPRA : SMC_ERROR;
5   END_VAR
6   VAR_OUTPUT
7       bHomeDone : BOOL;
8   END_VAR
9
```

```
1   ExecuteHome(Ref_Axis := THIS^.Ref_RotatorAxis,
2                   lrHomePosition := RotatorParameters.LR_HOME_POSITION,
3                   bStatus => bHomeDone,
4                   bError => bErrorPRA,
5                   ErrorID => ErrorIDPRA);
6
7   IF bErrorPRA AND ErrorIDPRA <> 0 THEN
```

```
8           THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ROTATOR_HOME_ERROR;
9           THIS^.PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId :=
        ErrorIDPRA, Language := 0);
10          THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
        PW_ERROR_ID.PW_ROTATOR_HOME_ERROR);
11      END_IF
12
```

1.1.1.1.1.3.8.4 Method: HaltDrive

```
1       METHOD PUBLIC HaltDrive
2       VAR
3           bError : BOOL;
4           ErrorID : SMC_ERROR;
5       END_VAR
6       VAR_OUTPUT
7           bHaltDone : BOOL;
8           bBusy : BOOL;
9       END_VAR
10
```

```
1       ExecuteHalt(Ref_Axis := THIS^.Ref_RotatorAxis,
2                   lrDeceleration := RotatorParameters.LR_DECELERATION,
3                   lrJerk := RotatorParameters.LR_JERK,
4                   bStatus => bHaltDone,
5                   bBusy => bBusy,
6                   bError => bError,
7                   ErrorID => ErrorID);
8
9       IF bError AND ErrorID <> 0 THEN
10          THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ROTATOR_HALT_ERROR;
11          THIS^.PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId :=
        ErrorID, Language := 0);
12          THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(
        PW_ERROR_ID.PW_ROTATOR_HALT_ERROR);
13      END_IF
14
```

1.1.1.1.1.3.8.5 Method: MoveRelative

```
1       METHOD PUBLIC MoveRelative
2       VAR
3           bError : BOOL;
4           ErrorID : SMC_ERROR;
5       END_VAR
6       VAR_INPUT
7           lrDistance : LREAL;
8       END_VAR
9       VAR_OUTPUT
10          mMoveDone : BOOL;
11      END_VAR
12
```

```
1       ExecuteMoveRel(Ref_Axis := THIS^.Ref_RotatorAxis,
2                      lrDistance := lrDistance,
```

```
                lrVelocity := RotatorParameters.LR_VELOCITY,
                lrAcceleration := RotatorParameters.LR_ACCELERATION,
                lrDeceleration := RotatorParameters.LR_DECELERATION,
                lrJerk := RotatorParameters.LR_JERK,
                moveDone => mMoveDone,
                bError => bError,
                ErrorID => ErrorID);

IF bError AND ErrorID <> 0 THEN
    THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ROTATOR_MOVE_ERROR;
    THIS^.PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorID, Language := 0);
    THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ROTATOR_MOVE_ERROR);
END_IF
```

1.1.1.1.1.3.8.6 Method: ResetFaults

```
METHOD PUBLIC ResetFaults
VAR
    bErrorCancelled : BOOL := TRUE;   // HMI confiramtion
    bErrorOccured : BOOL := FALSE;
    sErrorCause : STRING;
    sErrorSolution : STRING;
END_VAR
VAR_OUTPUT
    bResetFaultDone : BOOL := FALSE;
END_VAR
```

```
IF THIS^.PW_Error.PW_ErrorID <> PW_ERROR_ID.PW_NO_ERROR THEN
    bErrorOccured := TRUE;
    sErrorCause := THIS^.PW_Error.sErrorCauseMessage;
    sErrorSolution := THIS^.PW_Error.sErrorSolutionMessage;
    IF bErrorCancelled THEN
        THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_NO_ERROR;
        THIS^.PW_Error.sErrorCauseMessage := '';
        THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_NO_ERROR);
    bResetFaultDone := TRUE;
    END_IF
    bResetFaultDone := FALSE;
ELSE
    bResetFaultDone := TRUE;
END_IF
```

1.1.1.1.1.3.8.7 Method: StopDrive

```
METHOD PUBLIC StopDrive
VAR
    bErrorPRA : BOOL;
    ErrorIDPRA : SMC_ERROR;
END_VAR
VAR_OUTPUT
    bStopDonePRA : BOOL;
END_VAR

```

```
CASE THIS^.Ref_RotatorAxis.nAxisState OF
    4,5,6,7 : ExecuteStop(Ref_Axis := THIS^.Ref_RotatorAxis,
                lrDeceleration := RotatorParameters.LR_DECELERATION,
                lrJerk := RotatorParameters.LR_JERK,
                bStatus => bStopDonePRA,
                bError => bErrorPRA,
                ErrorID => ErrorIDPRA);

    IF bErrorPRA AND ErrorIDPRA <> 0 THEN
        THIS^.PW_Error.PW_ErrorID := PW_ERROR_ID.PW_ROTATOR_STOP_ERROR;
        THIS^.PW_Error.sErrorCauseMessage := SMC_ErrorString(ErrorId := ErrorIDPRA, Language := 0);
        THIS^.PW_Error.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ROTATOR_STOP_ERROR);
    END_IF
END_CASE

```

1.1.1.1.1.3.8.8 Method: ThrowError

```
METHOD PUBLIC ThrowError
VAR_INPUT
    ThrownError : PW_ERROR;
END_VAR

```

```
THIS^.PW_Error.PW_ErrorID := ThrownError.PW_ErrorID;
THIS^.PW_Error.sErrorCauseMessage := ThrownError.sErrorCauseMessage;
THIS^.PW_Error.sErrorSolutionMessage := ThrownError.sErrorSolutionMessage;

```

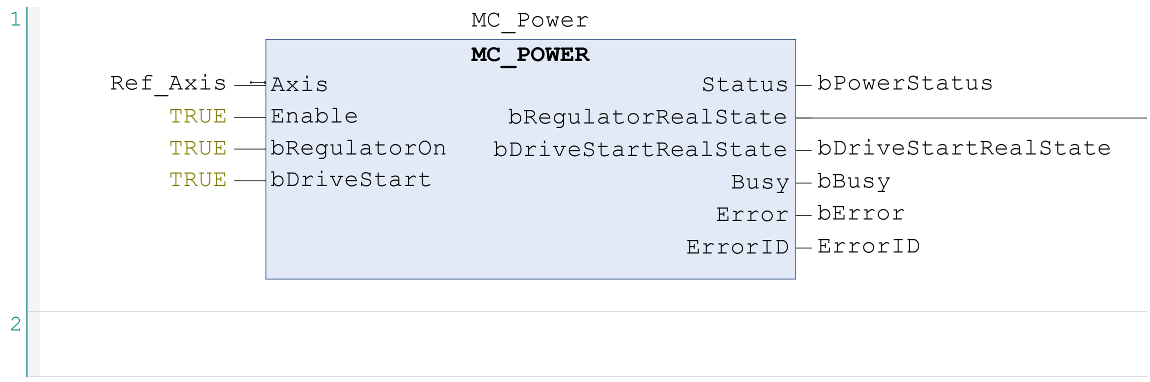
1.1.1.1.1.3.9 POU: ConveyorAxis

```
FUNCTION_BLOCK ConveyorAxis


```

1.1.1.1.1.3.9.1 Folder: Private Methods

1.1.1.1.1.3.9.1.1 Method: EnablePower

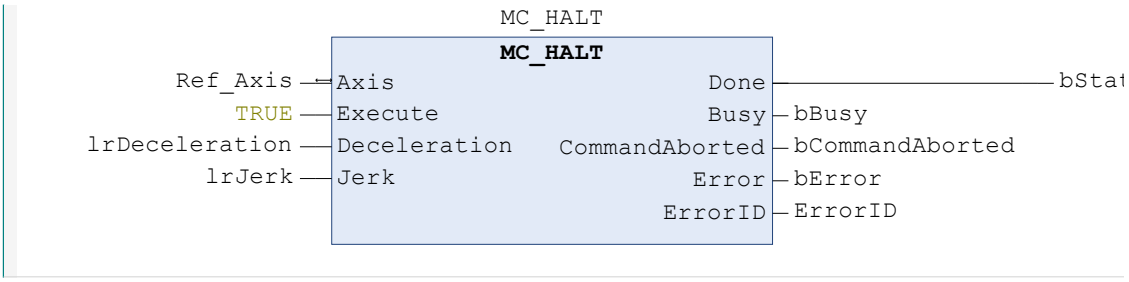
```
METHOD PROTECTED EnablePower
VAR_INPUT
    Ref_Axis : REFERENCE TO AXIS_REF_SM3 ;
END_VAR
VAR
    MC_Power : MC_Power ;
END_VAR
VAR_OUTPUT
    bPowerStatus : BOOL ;
    bRegulatorRealState : BOOL ;
    bDriveStartRealState : BOOL ;
    bBusy : BOOL ;
    bError : BOOL ;
    ErrorID : SMC_ERROR ;
END_VAR

```



```
-bRegulatorRealState
```

1.1.1.1.1.3.9.1.2 Method: ExecuteHalt

```
1   METHOD PROTECTED ExecuteHalt
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrDeceleration : LREAL;
5       lrJerk : LREAL;
6   END_VAR
7   VAR
8       MC_HALT : MC_HALT;
9   END_VAR
10  VAR_OUTPUT
11      bStatus : BOOL;
12      bBusy : BOOL;
13      bCommandAborted : BOOL;
14      bError : BOOL;
15      ErrorID : SMC_ERROR;
16  END_VAR
17
```

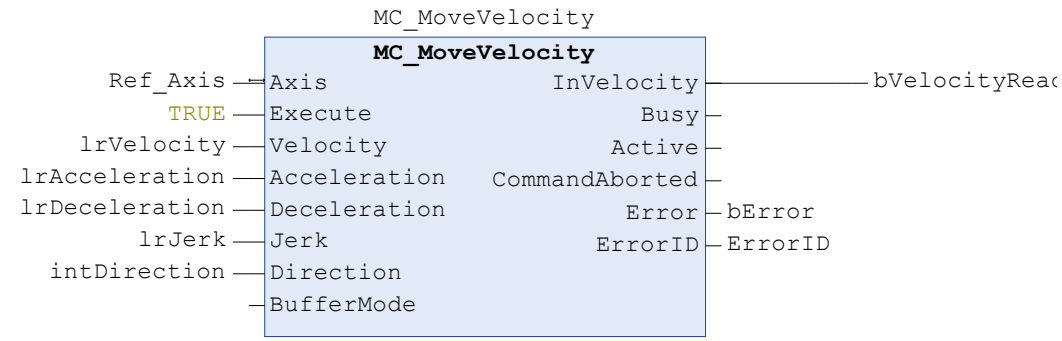


tus

1.1.1.1.1.3.9.1.3 Method: ExecuteMoveVel

```
1   METHOD PROTECTED ExecuteMoveVel : BOOL
2   VAR_INPUT
3       Ref_Axis : REFERENCE TO AXIS_REF_SM3;
4       lrVelocity : LREAL;
5       lrAcceleration : LREAL;
6       lrDeceleration : LREAL;
7       lrJerk : LREAL;
8       intDirection : INT;
9   END_VAR
10  VAR
11      MC_MoveVelocity : MC_MoveVelocity;
12  END_VAR
13  VAR_OUTPUT
14      bVelocityReached : BOOL;
15      bError : BOOL;
16      ErrorID : SMC_ERROR;
17  END_VAR
18
```

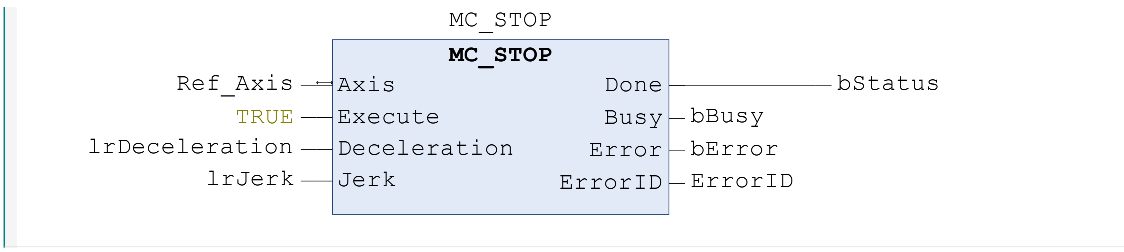
```
1
```



ched

1.1.1.1.1.3.9.1.4 Method: ExecuteStop

```
METHOD PROTECTED ExecuteStop
VAR_INPUT
    Ref_Axis : REFERENCE TO AXIS_REF_SM3 ;
    lrDeceleration : LREAL ;
    lrJerk : LREAL ;
END_VAR
VAR
    MC_STOP : MC_STOP ;
END_VAR
VAR_OUTPUT
    bStatus : BOOL ;
    bBusy : BOOL ;
    bError : BOOL ;
    ErrorID : SMC_ERROR ;
END_VAR
```



1.1.1.1.1.3.10 POU: EM_CenterPallet

```
FUNCTION_BLOCK PUBLIC EM_CenterPallet
VAR_INPUT
    Ref_CenterConveyor : REFERENCE TO CM_CenterConveyor ;
END_VAR
VAR
    bDiffuseEntrySensor : BOOL := FALSE ;          // Io-Link mapping
    bDiffuseExitSensor : BOOL := FALSE ;               // Io-Link mapping
END_VAR

```

1.1.1.1.1.3.10.1 Method: CyclePreparation

```
METHOD PUBLIC CyclePreparation : BOOL
VAR
    mPowerEnable : BOOL ;
END_VAR

```

```
THIS^.Ref_CenterConveyor.EnableDrivePower(bPowerStatus => mPowerEnable)
;
CyclePreparation := mPowerEnable ;

```

1.1.1.1.1.3.10.2 Method: EmergencyStop

```
METHOD PUBLIC EmergencyStop : BOOL
VAR
    mConveyorStopped : BOOL ;
END_VAR

```

```
THIS^.Ref_CenterConveyor.StopDrive(bStopDone => mConveyorStopped) ;
EmergencyStop := mConveyorStopped ;

```

1.1.1.1.1.3.10.3 Method: Halt

```
METHOD PUBLIC Halt : BOOL
VAR
    mConveyorHalted : BOOL ;
END_VAR

```

```
THIS^.Ref_CenterConveyor.HaltDrive(bHaltDone => mConveyorHalted) ;
Halt := mConveyorHalted ;


```

1.1.1.1.1.3.10.4 Method: ProcessStage1

```
METHOD PUBLIC ProcessStage1
VAR
    bHaltDone : BOOL;
END_VAR
VAR_IN_OUT
    mStage1Busy : BOOL;
    mStage1Done : BOOL;
END_VAR


```

```
IF NOT bDiffuseExitSensor THEN
    THIS^.Ref_CenterConveyor.MoveVelocity(intDirection := MC_DIRECTION.
positive);
    mStage1Busy := TRUE;
ELSE
    THIS^.Ref_CenterConveyor.HaltDrive(bHaltDone => bHaltDone);
    IF bHaltDone THEN
        mStage1Done := TRUE; mStage1Busy := FALSE;
    END_IF
END_IF

```

1.1.1.1.1.3.10.5 Method: ProcessStage2

```
METHOD PUBLIC ProcessStage2
VAR
    bHaltDone : BOOL;
END_VAR
VAR_IN_OUT
    mStage2Busy : BOOL;
    mStage2Done : BOOL;
END_VAR


```

```
IF bDiffuseExitSensor THEN
    THIS^.Ref_CenterConveyor.MoveVelocity(intDirection := MC_DIRECTION.
positive);
    mStage2Busy := TRUE;
ELSE
    Ref_CenterConveyor.HaltDrive(bHaltDone => bHaltDone);
    IF bHaltDone THEN
        mStage2Done := TRUE; mStage2Busy := FALSE;
    END_IF
END_IF

```

1.1.1.1.1.3.10.6 Method: ResetFaults

```
METHOD PUBLIC ResetFaults : BOOL
VAR
    mCenterConveyorResetFaultDone : BOOL := FALSE ;
END_VAR

```

```
THIS^.Ref_CenterConveyor.ResetFaults(bResetFaultDone => mCenterConveyorResetFaultDone);
ResetFaults := mCenterConveyorResetFaultDone;

```

1.1.1.1.1.3.11 POU: EM_EnterPallet

```
FUNCTION_BLOCK PUBLIC EM_EnterPallet
VAR_INPUT
    Ref_EntryConveyor : REFERENCE TO CM_EnterConveyor;
END_VAR
VAR
    bDiffuseEntrySensor : BOOL := FALSE;          // Io-Link mapping
    bDiffuseExitSensor : BOOL := FALSE;               // Io-Link mapping
END_VAR

```

1.1.1.1.1.3.11.1 Method: CyclePreparation

```
METHOD PUBLIC CyclePreparation : BOOL
VAR
    mPowerEnable : BOOL;
END_VAR

```

```
THIS^.Ref_EntryConveyor.EnableDrivePower(bPowerStatus => mPowerEnable);
CyclePreparation := mPowerEnable;

```

1.1.1.1.1.3.11.2 Method: EmergencyStop

```
METHOD PUBLIC EmergencyStop : BOOL
VAR
    mConveyorStopped : BOOL;
END_VAR

```

```
THIS^.Ref_EntryConveyor.StopDrive(bStopDone => mConveyorStopped);
EmergencyStop := mConveyorStopped;

```

1.1.1.1.1.3.11.3 Method: Halt

```
1   METHOD PUBLIC Halt : BOOL
2   VAR
3       mConveyorHalted : BOOL;
4   END_VAR
5
```

```
1   THIS^.Ref_EntryConveyor.HaltDrive(bHaltDone => mConveyorHalted);
2   Halt := mConveyorHalted;
3
4
5
```

1.1.1.1.1.3.11.4 Method: ProcessStage1

```
1   METHOD PUBLIC ProcessStage1
2   VAR_IN_OUT
3       mStage1Busy : BOOL;
4       mStage1Done : BOOL;
5   END_VAR
6
7
```

```
1
2
3   IF bDiffuseEntrySensor OR mStage1Busy THEN
4       IF NOT bDiffuseExitSensor THEN
5           THIS^.Ref_EntryConveyor.MoveVelocity(intDirection :=
    MC_DIRECTION.positive);
6           mStage1Busy := TRUE;
7       ELSE
8           mStage1Done := TRUE; mStage1Busy := FALSE;
9       END_IF
10  END_IF
11
12
```

1.1.1.1.1.3.11.5 Method: ProcessStage2

```
1   METHOD PUBLIC ProcessStage2
2   VAR
3       bHaltDone : BOOL;
4   END_VAR
5   VAR_IN_OUT
6       mStage2Busy : BOOL;
7       mStage2Done : BOOL;
8   END_VAR
9
10
```

```
1   IF bDiffuseExitSensor THEN
2       THIS^.Ref_EntryConveyor.MoveVelocity(intDirection := MC_DIRECTION.
```

```
        positive ) ;
3           mStage2Busy  :=  TRUE ;
4       ELSE
5           Ref_EntryConveyor . HaltDrive ( bHaltDone  =>  bHaltDone ) ;
6           IF  bHaltDone  THEN
7               mStage2Done  :=  TRUE ;  mStage2Busy  :=  FALSE ;
8           END_IF
9       END_IF
10
```

1.1.1.1.1.3.11.6 Method: ResetFaults

```
1       METHOD  PUBLIC  ResetFaults  :  BOOL
2       VAR
3           mEnterConveyorResetFaultDone  :  BOOL  :=  FALSE ;
4       END_VAR
5
```

```
1       THIS ^ . Ref_EntryConveyor . ResetFaults ( bResetFaultDone  =>
        mEnterConveyorResetFaultDone ) ;
2       ResetFaults  :=  mEnterConveyorResetFaultDone ;
3
```

1.1.1.1.1.3.11.7 Method: ThrowUsageError

```
1       METHOD  PUBLIC  ThrowUsageError
2       VAR_INPUT
3           errUsageError  :  PW_ERROR ;
4       END_VAR
5
```

```
1       THIS ^ . Ref_EntryConveyor . ThrowError ( ThrownError  :=  errUsageError ) ;
2
```

1.1.1.1.1.3.12 POU: EM_ExitPallet

```
1       FUNCTION_BLOCK  PUBLIC  EM_ExitPallet
2       VAR_INPUT
3           Ref_ExitConveyor  :  REFERENCE  TO  CM_ExitConveyor ;
4       END_VAR
5       VAR
6           bDiffuseEntrySensor  :  BOOL  :=  FALSE ;          // Io-Link mapping
7           bDiffuseExitSensor  :  BOOL  :=  FALSE ;               // Io-Link mapping
8           bRedundantExitSensor  :  BOOL  :=  FALSE ;             // Prevents the
        pallet falling out.
9       END_VAR
10
```

1.1.1.1.1.3.12.1 Method: CyclePreparation

```
METHOD PUBLIC CyclePreparation : BOOL
VAR
    mPowerEnable : BOOL;
END_VAR

```

```
THIS^.Ref_ExitConveyor.EnableDrivePower(bPowerStatus => mPowerEnable);
CyclePreparation := mPowerEnable;

```

1.1.1.1.1.3.12.2 Method: EmergencyStop

```
METHOD PUBLIC EmergencyStop : BOOL
VAR
    mConveyorStopped : BOOL;
END_VAR

```

```
THIS^.Ref_ExitConveyor.StopDrive(bStopDone => mConveyorStopped);
EmergencyStop := mConveyorStopped;

```

1.1.1.1.1.3.12.3 Method: Halt

```
METHOD PUBLIC Halt : BOOL
VAR
    mConveyorHalted : BOOL;
END_VAR

```

```
THIS^.Ref_ExitConveyor.HaltDrive(bHaltDone => mConveyorHalted);
Halt := mConveyorHalted;


```

1.1.1.1.1.3.12.4 Method: IsExitBlocked

```
METHOD PUBLIC IsExitBlocked
VAR
    bHaltDone : BOOL;
END_VAR
VAR_OUTPUT
    mStage1Blocked : BOOL;
END_VAR

```

```
IF bDiffuseExitSensor THEN
    mStage1Blocked := TRUE;
ELSE
    IF bRedundantExitSensor THEN
```

```
            mStage1Blocked := TRUE;
        ELSE mStage1Blocked := FALSE;
        END_IF
    END_IF
```

1.1.1.1.1.3.12.5 Method: ProcessStage1

```
    METHOD PUBLIC ProcessStage1
    VAR_INPUT
        mStage1Blocked : BOOL;
    END_VAR
    VAR
        bHaltDone : BOOL;
    END_VAR
    VAR_IN_OUT
        mStage1Busy : BOOL;
        mStage1Done : BOOL;

    END_VAR
```

```
    IF NOT mStage1Blocked THEN
        THIS^.Ref_ExitConveyor.MoveVelocity(intDirection := MC_DIRECTION.
    positive);
        mStage1Busy := TRUE;
    ELSE
        THIS^.Ref_ExitConveyor.HaltDrive(bHaltDone => bHaltDone);
        IF bHaltDone THEN
            mStage1Done := TRUE; mStage1Busy := FALSE;
        END_IF
    END_IF
```

1.1.1.1.1.3.12.6 Method: ResetFaults

```
    METHOD PUBLIC ResetFaults : BOOL
    VAR
        mExitConveyorResetFaultDone : BOOL := FALSE;
    END_VAR
```

```
    THIS^.Ref_ExitConveyor.ResetFaults(bResetFaultDone =>
    mExitConveyorResetFaultDone);
    ResetFaults := mExitConveyorResetFaultDone;
```

1.1.1.1.1.3.13 POU: EM_PalletWrapper

```
FUNCTION_BLOCK EM_PalletWrapper
VAR_INPUT
    Ref_FilmAplicator : REFERENCE TO CM_FilmAplicator;
    Ref_PalletRotator : REFERENCE TO CM_PalletRotator;
    Ref_FilmClamp : REFERENCE TO CM_FilmClamp;
    Ref_FilmSticker : REFERENCE TO CM_FilmSticker;
    Ref_PalletProtector : REFERENCE TO CM_PalletProtector;
END_VAR
VAR_OUTPUT
     CurrentAplicatorMode : AplicatorMode;
END_VAR


```

1.1.1.1.1.3.13.1 Folder: AuxillaryMethods

1.1.1.1.1.3.13.1.1 Method: ClampProcess

```
METHOD PRIVATE ClampProcess
VAR_INPUT
    usintCurrentLayerNumber : USINT;
END_VAR
VAR_IN_OUT
    mCloseClampDone : BOOL;
    mReleaseClampDone : BOOL;
END_VAR

```

```
 IF usintCurrentLayerNumber = 2 THEN
    IF NOT mReleaseClampDone THEN
        Ref_FilmClamp.ReleaseClamp(bReleaseDone => mReleaseClampDone);
    END_IF
END_IF

IF usintCurrentLayerNumber = 8 THEN
    IF NOT mCloseClampDone THEN
        Ref_FilmClamp.CloseClamp(bCloseDone => mCloseClampDone);
    END_IF
END_IF

```

1.1.1.1.1.3.13.1.2 Method: FinishTopLayer

```
METHOD PRIVATE FinishTopLayer : BOOL
VAR_INPUT
    usintCurrentLayerNumber : USINT;
    usintWrapsNumber : USINT;
    usintCurrentWrap : USINT;
END_VAR
VAR
    mRotatorMoveDone : BOOL := FALSE;
    mRotatorHaltDone : BOOL := FALSE;
END_VAR
VAR_OUTPUT
    mFinishTopLayerDone : BOOL := FALSE;
END_VAR

```

```
IF usintCurrentWrap <= usintWrapsNumber THEN
    IF NOT mRotatorMoveDone THEN
        Ref_PalletRotator.MoveRelative(lrDistance := RotatorParameters.LR_FIXED_POSITION - Ref_PalletRotator.Ref_RotatorAxis.fActPosition,
                                       mMoveDone => mRotatorMoveDone);

    END_IF
    IF mRotatorMoveDone AND (usintCurrentWrap < usintWrapsNumber) THEN
        usintCurrentWrap := usintCurrentWrap + 1;
        mRotatorMoveDone := FALSE;
    END_IF
ELSE mFinishTopLayerDone := TRUE; //when particular layer does not require wrapping.
END_IF

IF mRotatorMoveDone THEN
    mFinishTopLayerDone := TRUE;
ELSE mFinishTopLayerDone := FALSE;
END_IF


```

1.1.1.1.1.3.13.1.3 Method: ProtectorProcess

```
METHOD PRIVATE ProtectorProcess
VAR_INPUT
    usintCurrentLayerNumber : USINT;
    mCloseClampDone : BOOL;
    mStickerDetachDone : BOOL;
    mPutLayerDone : BOOL;
END_VAR
VAR_IN_OUT
    mExtendProtectorDone : BOOL;
    mReleaseProtectorDone : BOOL;
END_VAR

```

```
IF usintCurrentLayerNumber = 7 THEN
    IF mPutLayerDone AND NOT mExtendProtectorDone THEN
        Ref_PalletProtector.ExtendProtector(bExtendDone => mExtendProtectorDone);
    END_IF
END_IF

IF usintCurrentLayerNumber = 8
    AND mCloseClampDone
    AND mStickerDetachDone THEN
    IF NOT mReleaseProtectorDone THEN
        Ref_PalletProtector.ReleaseProtector(bReleaseDone => mReleaseProtectorDone);
    END_IF
END_IF

```

1.1.1.1.1.3.13.1.4 Method: PutLayer

```
METHOD PRIVATE PutLayer
VAR_INPUT
    usintWrapsNumber : USINT;
    lrAplicatorAimPosition : LREAL;
END_VAR
VAR
    mAplicatorMoveDone : BOOL;
    mRotatorMoveDone : BOOL;
END_VAR
VAR_IN_OUT
    usintCurrentWrap : USINT;
END_VAR
VAR_OUTPUT
    mPutLayerDone : BOOL;
END_VAR

```

```
// Aplicator movement
IF NOT mAplicatorMoveDone THEN
    Ref_FilmAplicator.MoveAbsolute(lrPosition := lrAplicatorAimPosition,
                                   bMoveDone => mAplicatorMoveDone);
END_IF
// Rotator movement
IF usintCurrentWrap <= usintWrapsNumber THEN
    IF NOT mRotatorMoveDone THEN
    Ref_PalletRotator.MoveRelative(lrDistance := RotatorParameters.LR_FIXED_POSITION - Ref_PalletRotator.Ref_RotatorAxis.fActPosition,
                                   mMoveDone => mRotatorMoveDone);
    END_IF
    IF mRotatorMoveDone AND (usintCurrentWrap < usintWrapsNumber) THEN
        usintCurrentWrap := usintCurrentWrap + 1;
        mRotatorMoveDone := FALSE;
    END_IF
ELSE mRotatorMoveDone := TRUE;    //when particular layer does not require wrapping.
END_IF

```

```
19      IF mAplicatorMoveDone AND mRotatorMoveDone THEN
20          mPutLayerDone := TRUE;
21      ELSE mPutLayerDone := FALSE;
22      END_IF
23
24
25
```

1.1.1.1.1.3.13.1.5 Method: SetNewTopLayer

```
1       METHOD PRIVATE SetNewTopLayer
2       VAR_INPUT
3           usintCurrentLayerNumber : USINT;
4       END_VAR
5       VAR
6           bHaltDone : BOOL;
7       END_VAR
8       VAR_OUTPUT
9           bMoveDone : BOOL;
10      END_VAR
11      VAR_IN_OUT
12          lrPalletHeight : LREAL;
13      END_VAR
14
```

```
1       IF NOT bMoveDone THEN
2           Ref_FilmAplicator.PositionToPalletHeight(usintCurrentLayerNumber := usintCurrentLayerNumber,
3                                                    lrPalletHeight := lrPalletHeight,
4                                                    bMoveDone => bMoveDone);
5       END_IF
6       (*
7       IF NOT bMoveDone AND THIS^.Ref_FilmAplicator.HeightSensor THEN
8              Ref_FilmAplicator.HaltDrive(bHaltDone => bHaltDone);
9              IF bHaltDone THEN
10                    lrPalletHeight := Ref_FilmAplicator.Ref_AplicatorAxis.fActPosition;
11                    bMoveDone := TRUE;
12             END_IF
13      END_IF

14      IF bMoveDone THEN
15             IF NOT THIS^.Ref_FilmAplicator.HeightSensor THEN
16
        Ref_FilmAplicator.PositionToPalletHeight(usintCurrentLayerNumber := 8,
17
        lrPalletHeight := lrPalletHeight,
18                                                                          bMoveDone => bMoveDone);
19             ELSE
20                    Ref_FilmAplicator.HaltDrive(bHaltDone => bHaltDone);
21                    IF bHaltDone THEN
22                           lrPalletHeight := Ref_FilmAplicator.Ref_AplicatorAxis.fActPosition;
23                           bMoveDone := TRUE;
```

```
                END_IF
            END_IF
        END_IF*}
```

1.1.1.1.1.3.13.1.6 Method: StickerProcess

```
METHOD PRIVATE StickerProcess
VAR_INPUT
    usintCurrentLayerNumber : usint;
    mCloseClampDone : BOOL;
END_VAR
VAR_IN_OUT
    mStickerAttachDone : BOOL;
    mStickerDetachDone : BOOL;
END_VAR
```

```
IF usintCurrentLayerNumber = 8 AND mCloseClampDone THEN
    IF NOT mStickerAttachDone THEN
            Ref_FilmSticker.AttachSticker(bAttachDone => mStickerAttachDone);
        RETURN;
    ELSE
        IF NOT mStickerDetachDone THEN
            Ref_FilmSticker.DetachSticker(bDetachDone => mStickerDetachDone);
        END_IF
    END_IF
END_IF
```

1.1.1.1.1.3.13.2 Method: ChangeWrappingMode

```
METHOD PUBLIC ChangeWrappingMode
VAR_INPUT
    newAplicatorModeName : STRING;
END_VAR
```

```
IF newAplicatorModeName = AplicatorModeNames.arrModeNames[0] THEN
    THIS^.CurrentAplicatorMode.sModeName := AplicatorModeNames.arrModeNames[0];
    THIS^.CurrentAplicatorMode.arrLayeringSystem := LayerSystems.arrEcoSystem;
END_IF
IF newAplicatorModeName = AplicatorModeNames.arrModeNames[1] THEN
    THIS^.CurrentAplicatorMode.sModeName := AplicatorModeNames.arrModeNames[1];
    THIS^.CurrentAplicatorMode.arrLayeringSystem := LayerSystems.arrNormalSystem;
END_IF
IF newAplicatorModeName = AplicatorModeNames.arrModeNames[2] THEN
    THIS^.CurrentAplicatorMode.sModeName := AplicatorModeNames.
```

```
        arrModeNames [ 2 ] ;
11          THIS ^ . CurrentAplicatorMode . arrLayeringSystem  :=  LayerSystems .
        arrHighPerformanceSystem ;
12      END_IF
13
14
15
16
```

1.1.1.1.1.3.13.3 Method: CyclePreparation

```
1       METHOD  PUBLIC  CyclePreparation  :  BOOL
2       VAR
3           mAplicatorPowerEnable  :  BOOL ;
4           mAplicatorHome  :  BOOL ;
5           mRotatorPowerEnable  :  BOOL ;
6           mRotatorHome  :  BOOL ;
7           mClampInit  :  BOOL ;
8           mProtectorInit  :  BOOL ;
9           mStickerInit  :  BOOL ;
10      END_VAR
11
```

```
1       Ref_FilmAplicator . EnableDrivePower ( bPowerStatus  =>  mAplicatorPowerEnable )
        ;
2       IF  mAplicatorPowerEnable  THEN
3           Ref_FilmAplicator . ExecuteDriveHome ( bHomeDone  =>  mAplicatorHome ) ;
4       END_IF
5
6       Ref_PalletRotator . EnableDrivePower ( bPowerStatusPRA  =>  mRotatorPowerEnable
        ) ;
7       IF  mRotatorPowerEnable  THEN
8           Ref_PalletRotator . ExecuteDriveHome ( bHomeDone  =>  mRotatorHome ) ;
9       END_IF
10
11      Ref_FilmClamp . InitializeClamp ( bInitializationDone  =>  mClampInit ) ;
12      Ref_PalletProtector . InitializeProtector ( bInitializationDone  =>
        mProtectorInit ) ;
13      Ref_FilmSticker . InitializeSticker ( bInitializationDone  =>  mStickerInit ) ;
14
15      CyclePreparation  :=  mAplicatorHome
16                           AND  mRotatorHome
17                           AND  mClampInit
18                           AND  mProtectorInit
19                           AND  mStickerInit ;
20
```

1.1.1.1.1.3.13.4 Method: EmergencyStop

```
METHOD PUBLIC EmergencyStop : BOOL
VAR
    mAplicatorStoped : BOOL;
    mRotatorStoped : BOOL;
    mClampReleased : BOOL;
END_VAR
```

```
Ref_FilmAplicator.StopDrive(bStopDone => mAplicatorStoped);
Ref_PalletRotator.StopDrive(bStopDonePRA => mRotatorStoped);
Ref_FilmClamp.ReleaseClamp(bReleaseDone => mClampReleased);

EmergencyStop := mAplicatorStoped AND mRotatorStoped AND mClampReleased;
```

1.1.1.1.1.3.13.5 Method: Halt

```
METHOD PUBLIC Halt : BOOL
VAR
    mAplicatorHalted : BOOL;
    mRotatorHalted : BOOL;
    mClampInit : BOOL;
END_VAR
```

```
Ref_FilmAplicator.HaltDrive(bHaltDone => mAplicatorHalted);
Ref_PalletRotator.HaltDrive(bHaltDone => mRotatorHalted);
Ref_FilmClamp.InitializeClamp(bInitializationDone => mClampInit);
Halt := mAplicatorHalted AND mRotatorHalted AND mClampInit;
```

1.1.1.1.1.3.13.6 Method: ProcessStage1

```
METHOD PUBLIC ProcessStage1
VAR_IN_OUT
    usintCurrentLayerNumber : USINT;
    usintCurrentWrap : USINT;
    mExtendProtectorDone : BOOL;
    mReleaseProtectorDone : BOOL;
    mCloseClampDone : BOOL;
    mReleaseClampDone : BOOL;
    mStickerAttachDone : BOOL;
    mStickerDetachDone : BOOL;
    mPutLayerDone : BOOL;
    mStage1Done : BOOL;
    mStage1Interupted : BOOL;
    mCurrentLayerDone : BOOL;
    lrPalletHeight : LREAL;
END_VAR
VAR
```

```
18          usintWrapsNumber  : USINT ;
19          lrAplicatorAimPosition  : LREAL ;
20          mAplicatorHaltDone  : BOOL ;
21          mRotatorHaltDone  : BOOL ;
22
23      END_VAR
24
```

```
1       IF Ref_FilmAplicator . HeightSensor  THEN
2           usintWrapsNumber  := CurrentAplicatorMode . arrLayeringSystem [
        usintCurrentLayerNumber ] ;
3           lrAplicatorAimPosition  := AplicatorParameters . LRA_LAYER_POSITIONS [
        usintCurrentLayerNumber ] ;
4           IF NOT mCurrentLayerDone  THEN
5               PutLayer ( usintWrapsNumber  := usintWrapsNumber ,
6                          lrAplicatorAimPosition  := lrAplicatorAimPosition ,
7                          usintCurrentWrap  := usintCurrentWrap ,
8                          mPutLayerDone  => mPutLayerDone ) ;
9
10              ClampProcess ( usintCurrentLayerNumber  := usintCurrentLayerNumber ,
11                         mCloseClampDone  := mCloseClampDone ,
12                         mReleaseClampDone  := mReleaseClampDone ) ;
13
14              StickerProcess ( usintCurrentLayerNumber  := usintCurrentLayerNumber ,
15                         mCloseClampDone  := mCloseClampDone ,
16                         mStickerAttachDone  := mStickerAttachDone ,
17                         mStickerDetachDone  := mStickerDetachDone ) ;
18
19              ProtectorProcess ( usintCurrentLayerNumber  :=
        usintCurrentLayerNumber ,
20                         mCloseClampDone  := mCloseClampDone ,
21                         mPutLayerDone  := mPutLayerDone ,
22                         mStickerDetachDone  := mStickerDetachDone ,
23                         mExtendProtectorDone  := mExtendProtectorDone ,
24                         mReleaseProtectorDone  := mReleaseProtectorDone ) ;
25              CASE usintCurrentLayerNumber  OF
26                  1 , 3 , 4  : IF mPutLayerDone  THEN mCurrentLayerDone  := TRUE ;
        END_IF
27                  2  : IF mPutLayerDone  AND mReleaseClampDone  THEN
        mCurrentLayerDone  := TRUE ;  END_IF
28              END_CASE
29          ELSE
30              usintCurrentWrap  := 1 ;
31              mCurrentLayerDone  := FALSE ;
32              mPutLayerDone  := FALSE ;
33              CASE usintCurrentLayerNumber  OF
34                  1 , 2 , 3  : usintCurrentLayerNumber  := usintCurrentLayerNumber +
        1 ;
35
36                  4  : usintCurrentLayerNumber  := 5 ; mStage1Done  := TRUE ;
37
38              END_CASE
39          END_IF
40      ELSE
41          lrPalletHeight  := Ref_FilmAplicator . Ref_AplicatorAxis . fActPosition ;
```

```
42          Ref_FilmAplicator . HaltDrive ( bHaltDone  =>  mAplicatorHaltDone ) ;
43          Ref_PalletRotator . HaltDrive ( bHaltDone  =>  mRotatorHaltDone ) ;
44          IF  mRotatorHaltDone  AND  mAplicatorHaltDone  THEN
45              mStage1Interupted  :=  TRUE ;
46          END_IF
47      END_IF
48
```

1.1.1.1.1.3.13.7 Method: ProcessStage2

```
1       METHOD  PUBLIC  ProcessStage2
2       VAR_INPUT
3           usintCurrentLayerNumber  :  USINT ;
4           usintCurrentWrap  :  USINT ;
5       END_VAR
6       VAR_OUTPUT
7           mStage2Done  :  BOOL ;
8       END_VAR
9
```

```
1       FinishTopLayer ( usintCurrentLayerNumber  :=  usintCurrentLayerNumber ,
2                   usintWrapsNumber  :=  CurrentAplicatorMode . arrLayeringSystem [
        usintCurrentLayerNumber ] ,
3                   usintCurrentWrap  :=  usintCurrentWrap ,
4                   mFinishTopLayerDone  =>  mStage2Done ) ;
5
```

1.1.1.1.1.3.13.8 Method: ProcessStage3

```
1       METHOD  PUBLIC  ProcessStage3
2       VAR_IN_OUT
3           usintCurrentLayerNumber  :  USINT ;
4           mStage3Done  :  BOOL ;
5       END_VAR
6       VAR_IN_OUT
7           lrPalletHeight  :  LREAL ;
8       END_VAR
9
```

```
1       usintCurrentLayerNumber  :=  5 ;
2       SetNewTopLayer ( usintCurrentLayerNumber  :=  usintCurrentLayerNumber ,
3                   lrPalletHeight  :=  lrPalletHeight ,
4                   bMoveDone  =>  mStage3Done ) ;
5
6
7
```

1.1.1.1.1.3.13.9 Method: ProcessStage4

```
METHOD PUBLIC ProcessStage4
VAR_INPUT
    lrPalletHeight : LREAL;
    mStage1Interupted : BOOL;
END_VAR
VAR_IN_OUT
    usintCurrentLayerNumber : USINT;
    usintCurrentWrap : USINT;
    mExtendProtectorDone : BOOL;
    mReleaseProtectorDone : BOOL;
    mCloseClampDone : BOOL;
    mReleaseClampDone : BOOL;
    mStickerAttachDone : BOOL;
    mStickerDetachDone : BOOL;
    mPutLayerDone : BOOL;
    mStage4Done : BOOL;
    mCurrentLayerDone : BOOL;
END_VAR
VAR
    usintWrapsNumber : USINT;
    lrAplicatorAimPosition : LREAL;
    bHaltDone : BOOL;
    ThrownError : PW_ERROR;
END_VAR

```

```
IF Ref_FilmAplicator.HeightSensor THEN
    IF mStage1Interupted AND usintCurrentLayerNumber = 5 THEN
        IF lrPalletHeight < AplicatorParameters.LRA_LAYER_POSITIONS[5]
THEN
            IF lrPalletHeight > AplicatorParameters.LRA_LAYER_POSITIONS[
6] THEN
                usintCurrentLayerNumber := 6;
            END_IF
            IF lrPalletHeight < AplicatorParameters.LRA_LAYER_POSITIONS[6
]
                AND lrPalletHeight > AplicatorParameters.
LRA_LAYER_POSITIONS[7] THEN
                usintCurrentLayerNumber := 7;
            END_IF
            IF lrPalletHeight < AplicatorParameters.LRA_LAYER_POSITIONS[
7]
                AND lrPalletHeight > AplicatorParameters.
LRA_LAYER_POSITIONS[8] THEN
                    usintCurrentLayerNumber := 8;
            END_IF
        END_IF
        usintWrapsNumber := CurrentAplicatorMode.arrLayeringSystem[
usintCurrentLayerNumber];
        lrAplicatorAimPosition := lrPalletHeight;
    ELSE
        usintWrapsNumber := CurrentAplicatorMode.arrLayeringSystem[
```

```
usintCurrentLayerNumber ] ;
20          lrAplicatorAimPosition  := AplicatorParameters . LRA_LAYER_POSITIONS [
    usintCurrentLayerNumber ] ;
21      END_IF
22
23      IF NOT mCurrentLayerDone THEN
24          PutLayer ( usintWrapsNumber  := usintWrapsNumber ,
25                     lrAplicatorAimPosition  := lrAplicatorAimPosition ,
26                     usintCurrentWrap  := usintCurrentWrap ,
27                     mPutLayerDone  => mPutLayerDone ) ;
28
29          ClampProcess ( usintCurrentLayerNumber  := usintCurrentLayerNumber ,
30                     mCloseClampDone  := mCloseClampDone ,
31                     mReleaseClampDone  := mReleaseClampDone ) ;
32
33          StickerProcess ( usintCurrentLayerNumber  := usintCurrentLayerNumber ,
34                     mCloseClampDone  := mCloseClampDone ,
35                     mStickerAttachDone  := mStickerAttachDone ,
36                     mStickerDetachDone  := mStickerDetachDone ) ;
37
38          ProtectorProcess ( usintCurrentLayerNumber  :=
    usintCurrentLayerNumber ,
39                     mCloseClampDone  := mCloseClampDone ,
40                     mPutLayerDone  := mPutLayerDone ,
41                     mStickerDetachDone  := mStickerDetachDone ,
42                     mExtendProtectorDone  := mExtendProtectorDone ,
43                     mReleaseProtectorDone  := mReleaseProtectorDone ) ;
44          CASE usintCurrentLayerNumber OF
45              5 , 6 : IF mPutLayerDone THEN mCurrentLayerDone  := TRUE ;
    END_IF
46              7 : IF mPutLayerDone AND mExtendProtectorDone THEN
    mCurrentLayerDone  := TRUE ; END_IF
47              8 : IF mPutLayerDone AND mCloseClampDone AND
    mStickerDetachDone AND mReleaseProtectorDone THEN mCurrentLayerDone  :=
    TRUE ; END_IF
48          END_CASE
49      ELSE
50          usintCurrentWrap  := 1 ;
51          mCurrentLayerDone  := FALSE ;
52          mPutLayerDone  := FALSE ;
53
54          CASE usintCurrentLayerNumber OF
55              5 , 6 , 7 : usintCurrentLayerNumber  := usintCurrentLayerNumber +
    1 ;
56              8 : mStage4Done  := TRUE ;
57          END_CASE
58      END_IF
59      {*ELSE
60          Ref_FilmAplicator.HaltDrive(bHaltDone => bHaltDone);
61               IF bHaltDone THEN
62                     ThrownError.PW_ErrorID        :=
    PW_ERROR_ID.PW_APLICATOR_PALLET_SENSOR_ERROR;
63                     ThrownError.sErrorCauseMessage := ' ';
64                     ThrownError.sErrorSolutionMessage:=
    PW_getErrorSolution(PW_ERROR_ID.PW_APLICATOR_PALLET_SENSOR_ERROR);
65
66                     Ref_FilmAplicator.ThrowError(ThrownError := ThrownError);
```

```
                    mStage4Done := TRUE;
                END_IF*}
    END_IF
```

1.1.1.1.1.3.13.10 Method: ProvideFilm

```
    METHOD PUBLIC ProvideFilm : BOOL
    VAR
        mFilmProvided : BOOL;
    END_VAR
```

```
    Ref_FilmClamp.ProvideFilm(bFilmProvided => mFilmProvided);
    ProvideFilm := mFilmProvided;
```

1.1.1.1.1.3.13.11 Method: ResetFaults

```
    METHOD PUBLIC ResetFaults : BOOL
    VAR
        mAplicatorResetFaultDone : BOOL := FALSE;
        mRotatorResetFaultDone : BOOL := FALSE;
        mClampResetFaultDone : BOOL := FALSE;
        mStickerResetFaultDone : BOOL := FALSE;
        mProtectorResetFaultDone : BOOL := FALSE;

    END_VAR
```

```
    THIS^.Ref_FilmAplicator.ResetFaults(bResetFaultDone =>
    mAplicatorResetFaultDone);
    THIS^.Ref_PalletRotator.ResetFaults(bResetFaultDone =>
    mRotatorResetFaultDone);
    THIS^.Ref_FilmClamp.ResetFaults(bResetFaultDone => mClampResetFaultDone
    );
    THIS^.Ref_FilmSticker.ResetFaults(bResetFaultDone =>
    mStickerResetFaultDone);
    THIS^.Ref_PalletProtector.ResetFaults(bResetFaultDone =>
    mProtectorResetFaultDone);


    ResetFaults := mAplicatorResetFaultDone
                   AND mRotatorResetFaultDone
                   AND mClampResetFaultDone
                   AND mStickerResetFaultDone
                   AND mProtectorResetFaultDone;

```

1.1.1.1.2 POU: prgCenterPallet

```
PROGRAM prgCenterPallet
VAR_INPUT
    Ref_CenterPallet : REFERENCE TO EM_CenterPallet;
    mCenterReadyToMoveIn : BOOL;
    mCenterReadyToMoveOut : BOOL;
    mWrapPalletProcessDone : BOOL;
END_VAR
VAR
    mStage1Done : BOOL;
    mStage1Busy : BOOL;
    mStage2Done : BOOL;
    mStage2Busy : BOOL;
    mProcessInit : BOOL := FALSE;
END_VAR
VAR_OUTPUT
    mEntryReadyToMoveIn : BOOL;
    mExitReadyToMoveIn : BOOL;
    mWrapperReadyToProcess : BOOL;
END_VAR


```

```
IF NOT mProcessInit THEN
    mStage1Done := FALSE; mStage1Busy := FALSE;
    mStage2Done := FALSE; mStage2Busy := FALSE;
    mEntryReadyToMoveIn := FALSE;
    mExitReadyToMoveIn := FALSE;
    mWrapperReadyToProcess := FALSE;
    mProcessInit := TRUE;
    RETURN;
ELSE
    // STAGE 1
    IF (mCenterReadyToMoveIn OR mStage1Busy) AND NOT mStage1Done THEN
        Ref_CenterPallet.ProcessStage1(mStage1Busy := mStage1Busy,
mStage1Done := mStage1Done);
    END_IF
    // STAGE 2
    IF mStage1Done AND (mStage2Busy OR (mCenterReadyToMoveOut AND
mWrapPalletProcessDone)) THEN
        Ref_CenterPallet.ProcessStage2(mStage2Busy := mStage2Busy,
mStage2Done := mStage2Done);
    END_IF

    IF NOT mStage1Done AND NOT mStage2Done THEN
        mEntryReadyToMoveIn := TRUE;
    ELSE mEntryReadyToMoveIn := FALSE;
    END_IF

    IF mStage1Done AND NOT mStage2Done AND NOT mStage2Busy THEN
        mWrapperReadyToProcess := TRUE;
    ELSE mWrapperReadyToProcess := FALSE;
    END_IF
```

```
28
29          IF mStage1Done AND mWrapPalletProcessDone THEN
30              mExitReadyToMoveIn := TRUE;
31          ELSE mExitReadyToMoveIn := FALSE;
32          END_IF
33
34          IF mStage1Done AND mStage2Done THEN
35              mProcessInit := FALSE;
36          END_IF
37      END_IF
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
```

1.1.1.1.3 POU: prgEnterPallet

```
1       PROGRAM prgEnterPallet
2       VAR_INPUT
3           Ref_EnterPallet : REFERENCE TO EM_EnterPallet;
4           mEntryReadyToMoveIn : BOOL;
5       END_VAR
6       VAR
7           errUsageError : PW_ERROR;
8           mStage1Done : BOOL;
9           mStage1Busy : BOOL;
10          mStage2Done : BOOL;
11          mStage2Busy : BOOL;
12          mProcessInit : BOOL := FALSE;
13      END_VAR
14      VAR_OUTPUT
15           mCenterReadyToMoveIn : BOOL;
16      END_VAR
17
```

```
1       IF NOT mProcessInit THEN
2           mStage1Done := FALSE; mStage1Busy := FALSE;
3           mStage2Done := FALSE; mStage2Busy := FALSE;
4           mCenterReadyToMoveIn := FALSE;
```

```
        mProcessInit := TRUE;
    ELSE
        // Machine used in inapropriate way. Throw an error.
        IF NOT mEntryReadyToMoveIn AND mStage1Busy THEN
            errUsageError.PW_ErrorID := PW_ERROR_ID.PW_ENTER_CONVEYOR_USAGE_ERROR;
            errUsageError.sErrorCauseMessage := '';
            errUsageError.sErrorSolutionMessage := PW_getErrorSolution(PW_ERROR_ID.PW_ENTER_CONVEYOR_USAGE_ERROR);
            Ref_EnterPallet.ThrowUsageError(errUsageError := errUsageError);
        END_IF

        // STAGE 1
        IF mEntryReadyToMoveIn AND NOT mStage1Done THEN
            Ref_EnterPallet.ProcessStage1(mStage1Busy := mStage1Busy, mStage1Done := mStage1Done);
        END_IF
        // STAGE 2
        IF mEntryReadyToMoveIn AND mStage1Done AND NOT mStage2Done THEN
            Ref_EnterPallet.ProcessStage2(mStage2Busy := mStage2Busy, mStage2Done := mStage2Done);
        END_IF

        IF mStage1Done AND mStage2Busy THEN
            mCenterReadyToMoveIn := TRUE;
        ELSE mCenterReadyToMoveIn := FALSE;
        END_IF
        IF mStage1Done AND mStage2Done THEN
            mProcessInit := FALSE;
        END_IF
    END_IF

```

1.1.1.1.4 POU: prgExitPallet

```
PROGRAM prgExitPallet
VAR_INPUT
    Ref_ExitPallet : REFERENCE TO EM_ExitPallet;
    mExitReadyToMoveIn : BOOL;
END_VAR
VAR
    mStage1Done : BOOL;
    mStage1Blocked : BOOL;
    mStage1Busy : BOOL;
    mProcessInit : BOOL := FALSE;
END_VAR
VAR_OUTPUT
    mCenterReadyToMoveOut : BOOL := FALSE;
END_VAR

```

```
IF NOT mProcessInit THEN
    mStage1Done := FALSE;
    mStage1Busy := FALSE;
    mStage1Blocked := FALSE;
```

```
5           mProcessInit  :=  TRUE ;
6       ELSE
7           // Stage 1
8           Ref_ExitPallet . IsExitBlocked ( mStage1Blocked  =>  mStage1Blocked ) ;
9           IF  ( mExitReadyToMoveIn  OR  mStage1Busy )  AND  NOT  mStage1Done  THEN
10          Ref_ExitPallet . ProcessStage1 ( mStage1Busy  :=  mStage1Busy ,
11                                           mStage1Done  :=  mStage1Done ,
12                                           mStage1Blocked  :=  mStage1Blocked ) ;
13          END_IF
14
15          IF  NOT  mStage1Blocked  AND  NOT  mStage1Done  THEN
16              mCenterReadyToMoveOut  :=  TRUE ;
17          ELSE  mCenterReadyToMoveOut  :=  FALSE ;
18          END_IF
19
20          IF  mStage1Done  AND  NOT  mStage1Blocked  THEN
21              mProcessInit  :=  FALSE ;
22          END_IF
23      END_IF
24
25
26
27
28
29
```

1.1.1.1.5 POU: prgMain

```
1       PROGRAM  prgMain
2       VAR
3           // Machine objects initialization.
4           EM_WrapPallet  :  EM_PalletWrapper ;
5           EM_EnterPallet  :  EM_EnterPallet ;
6           EM_CenterPallet  :  EM_CenterPallet ;
7           EM_ExitPallet  :  EM_ExitPallet ;
8
9           Ref_WrapPallet  :  REFERENCE  TO  EM_PalletWrapper ;
10          Ref_EnterPallet  :  REFERENCE  TO  EM_EnterPallet ;
11          Ref_ExitPallet  :  REFERENCE  TO  EM_ExitPallet ;
12          Ref_CenterPallet  :  REFERENCE  TO  EM_CenterPallet ;
13
14          CM_FilmClamp  :  CM_FilmClamp ;
15          CM_FilmSticker  :  CM_FilmSticker ;
16          CM_PalletProtector  :  CM_PalletProtector ;
17          CM_FilmAplicator  :  CM_FilmAplicator ;
18          CM_PalletRotator  :  CM_PalletRotator ;
19          CM_EnterConveyor  :  CM_EnterConveyor ;
20          CM_ExitConveyor  :  CM_ExitConveyor ;
21          CM_CenterConveyor  :  CM_CenterConveyor ;
22
23          Ref_FilmAplicator  :  REFERENCE  TO  CM_FilmAplicator ;
24          Ref_PalletRotator  :  REFERENCE  TO  CM_PalletRotator ;
25          Ref_FilmClamp  :  REFERENCE  TO  CM_FilmClamp ;
26          Ref_FilmSticker  :  REFERENCE  TO  CM_FilmSticker ;
27          Ref_PalletProtector  :  REFERENCE  TO  CM_PalletProtector ;
28          Ref_EnterConveyor  :  REFERENCE  TO  CM_EnterConveyor ;
```

```
29          Ref_ExitConveyor  : REFERENCE TO CM_ExitConveyor ;
30          Ref_CenterConveyor  : REFERENCE TO CM_CenterConveyor ;
31
32          // Initialization flags.
33          mMachineInit  : BOOL := FALSE ;
34          mUnitModesInit  : BOOL := FALSE ;
35
36          // PackML.
37          ProductionMode  : PACK_ML . UnitMode ;
38          ManualMode  : PACK_ML . UnitMode ;
39          unitManager  : PACK_ML . UnitModeManager ;
40
41          // Visualization.
42          bStartProcessButton  : BOOL := FALSE ;
43          bStopProcessButton  : BOOL := FALSE ;
44          sSelectedUnitModeName  : STRING ;
45          intDesiredAplicatorModeNameIndex  : INT := 1 ;
46          intDesiredUnitModeNameIndex  : INT := 1 ;
47          bChangeModesNotAllowed  : BOOL := TRUE ;
48
49          // Business logic markers.
50          mCenterReadyToMoveIn  : BOOL := FALSE ;
51          mCenterReadyToMoveOut  : BOOL := FALSE ;
52          mExitReadyToMoveIn  : BOOL := FALSE ;
53          mEntryReadyToMoveIn  : BOOL := FALSE ;
54          mWrapperReadyToProcess  : BOOL := FALSE ;
55          mWrapPalletProcessDone  : BOOL := FALSE ;
56      END_VAR
57
```

```
1       IF NOT mMachineInit THEN
2           // Control modules.
3           CM_FilmSticker ( ) ;
4           CM_FilmClamp ( ) ;
5           CM_PalletProtector ( ) ;
6           CM_FilmAplicator ( Ref_AplicatorAxis := IoConfig_Globals .
        FilmAplicatorAxis ) ;
7           CM_PalletRotator ( Ref_RotatorAxis := IoConfig_Globals .
        PalletRotatorAxis ) ;
8           CM_EnterConveyor ( Ref_EnterConveyorAxis := IoConfig_Globals .
        EnterPalletAxis ) ;
9           CM_ExitConveyor ( Ref_ExitConveyorAxis := IoConfig_Globals .
        ExitPalletAxis ) ;
10          CM_CenterConveyor ( Ref_CenterConveyorAxis := IoConfig_Globals .
        CenterPalletAxis ) ;
11
12          // Control modules references.
13          Ref_FilmClamp REF= CM_FilmClamp ;
14          Ref_FilmSticker REF= CM_FilmSticker ;
15          Ref_PalletProtector REF= CM_PalletProtector ;
16          Ref_FilmAplicator REF= CM_FilmAplicator ;
17          Ref_PalletRotator REF= CM_PalletRotator ;
18          Ref_EnterConveyor REF= CM_EnterConveyor ;
19          Ref_ExitConveyor REF= CM_ExitConveyor ;
```

```
20          Ref_CenterConveyor  REF=  CM_CenterConveyor ;
21
22          // Equipment modules references
23          EM_WrapPallet ( Ref_FilmAplicator  :=  Ref_FilmAplicator ,
24                      Ref_PalletRotator  :=  Ref_PalletRotator ,
25                      Ref_FilmClamp  :=  Ref_FilmClamp ,
26                      Ref_FilmSticker  :=  Ref_FilmSticker ,
27                      Ref_PalletProtector  :=  Ref_PalletProtector ) ;
28          EM_EnterPallet ( Ref_EntryConveyor  :=  Ref_EnterConveyor ) ;
29          EM_ExitPallet ( Ref_ExitConveyor  :=  Ref_ExitConveyor ) ;
30          EM_CenterPallet ( Ref_CenterConveyor  :=  Ref_CenterConveyor ) ;
31
32          // Equipment modules references
33          Ref_WrapPallet  REF=  EM_WrapPallet ;
34          Ref_EnterPallet  REF=  EM_EnterPallet ;
35          Ref_ExitPallet  REF=  EM_ExitPallet ;
36          Ref_CenterPallet  REF=  EM_CenterPallet ;
37
38          // Set default wrapping mode to 'Normal'.
39          Ref_WrapPallet . ChangeWrappingMode ( newAplicatorModeName  :=
        AplicatorModeNames . arrModeNames [ intDesiredAplicatorModeNameIndex ] ) ;
40          mMachineInit  :=  TRUE ;
41          RETURN ;
42      END_IF
43
44      // Initialize Unit Modes PackMl.
45      IF  NOT  mUnitModesInit  THEN
46          ProductionMode (
47                          sName  :=  UnitModeNames . arrUnitModeNames [ 0 ] ,
48                          dwSupportedStates  :=  PACK_ML . State . All ,
49                          dwAllowsLeavingFromStates  :=  PACK_ML . State . Idle  OR
        PACK_ML . State . Stopped  OR  PACK_ML . State . Resetting ,
50                          dwAllowsEnteringIntoStates  :=  PACK_ML . State . Idle  OR
        PACK_ML . State . Stopped  OR  PACK_ML . State . Resetting ,
51                          eInitialState  :=  PACK_ML . State . Idle ,
52                          xActive  :=  TRUE
53                         ) ;
54          ManualMode (
55                          sName  :=   UnitModeNames . arrUnitModeNames [ 1 ] ,
56                          dwSupportedStates  :=  PACK_ML . State . All  -  PACK_ML .
        State . Holding  -  PACK_ML . State . UnHolding  -  PACK_ML . State . Held  -
        PACK_ML . State . Suspending  -  PACK_ML . State . UnSuspending  -  PACK_ML . State
        . Suspended  -  PACK_ML . State . Completing  -  PACK_ML . State . Complete ,
57                          dwAllowsLeavingFromStates  :=  PACK_ML . State . Idle  OR
        PACK_ML . State . Stopped  OR  PACK_ML . State . Resetting ,
58                          dwAllowsEnteringIntoStates  :=  PACK_ML . State . Idle  OR
        PACK_ML . State . Stopped  OR  PACK_ML . State . Resetting ,
59                          eInitialState  :=  PACK_ML . State . Idle ,
60                          xActive  :=  FALSE
61                     ) ;
62          unitManager . Register ( pum  :=  ProductionMode ) ;
63          unitManager . Register ( pum  :=  ManualMode ) ;
64          mUnitModesInit  :=  TRUE ;
65          RETURN ;
66      END_IF
67
```

```
68      // Errors catching
69      IF EM_WrapPallet . Ref_FilmAplicator . PW_Error . PW_ErrorID <> PW_ERROR_ID .
        PW_NO_ERROR
70          OR EM_WrapPallet . Ref_PalletRotator . PW_Error . PW_ErrorID <>
        PW_ERROR_ID . PW_NO_ERROR
71          OR ( EM_WrapPallet . Ref_FilmClamp . PW_Error . PW_ErrorID <> PW_ERROR_ID .
        PW_NO_ERROR
72              AND EM_WrapPallet . Ref_FilmClamp . PW_Error . PW_ErrorID <>
        PW_ERROR_ID . PW_FILM_NOT_DETECTED_ERROR )
73          OR EM_WrapPallet . Ref_FilmSticker . PW_Error . PW_ErrorID <> PW_ERROR_ID
        . PW_NO_ERROR
74          OR EM_WrapPallet . Ref_PalletProtector . PW_Error . PW_ErrorID <>
        PW_ERROR_ID . PW_NO_ERROR
75          OR EM_EnterPallet . Ref_EntryConveyor . PW_Error . PW_ErrorID <>
        PW_ERROR_ID . PW_NO_ERROR
76          OR EM_ExitPallet . Ref_ExitConveyor . PW_Error . PW_ErrorID <> PW_ERROR_ID
        . PW_NO_ERROR
77          OR EM_CenterPallet . Ref_CenterConveyor . PW_Error . PW_ErrorID <>
        PW_ERROR_ID . PW_NO_ERROR THEN
78              unitManager . ActiveUnitMode . Abort ( ) ;

79      END_IF
80
81      IF mMachineInit AND mUnitModesInit THEN
82      // Catch stop button.
83      IF bStopProcessButton THEN
84              bStartProcessButton := FALSE ;
85              unitManager . ActiveUnitMode . Stop ( ) ;

86      END_IF
87
88      // Allow unit mode and wrapping mode changes.
89      IF unitManager . ActiveUnitMode . AllowsLeaving THEN
90          bChangeModesNotAllowed := FALSE ;
91      ELSE bChangeModesNotAllowed := TRUE ;
92      END_IF
93
94      IF NOT bChangeModesNotAllowed THEN
95          // Change Unit Mode
96          IF UnitModeNames . arrUnitModeNames [ intDesiredUnitModeNameIndex ] <>
        unitManager . ActiveUnitMode . Name THEN
97              IF UnitModeNames . arrUnitModeNames [ intDesiredUnitModeNameIndex ] =
        ProductionMode . Name THEN
98                  unitManager . SwitchUnitMode ( UnitModeNames . arrUnitModeNames [ 0 ]
        ) ;
99              ELSIF UnitModeNames . arrUnitModeNames [ intDesiredUnitModeNameIndex ]
         = ManualMode . Name THEN
100                 unitManager . SwitchUnitMode ( UnitModeNames . arrUnitModeNames [ 1 ]
        ) ;
101             END_IF
102         END_IF
103         // Change wrapping mode
104         IF AplicatorModeNames . arrModeNames [ intDesiredAplicatorModeNameIndex ]
        <> EM_WrapPallet . CurrentAplicatorMode . sModeName THEN
105             EM_WrapPallet . ChangeWrappingMode ( newAplicatorModeName :=
        AplicatorModeNames . arrModeNames [ intDesiredAplicatorModeNameIndex ] ) ;
```

```
106         END_IF
107     END_IF
108
109     // PackMl sequence
110     CASE  unitManager . ActiveUnitMode . CurrentState  OF
111         PACK_ML . State . Stopped :
112             IF  NOT  bStopProcessButton  AND  bStartProcessButton  THEN
113                 unitManager . ActiveUnitMode . Reset ( ) ;
114             END_IF
115
116         PACK_ML . State . Resetting :
117             IF  EM_WrapPallet . ResetFaults ( )
118                 AND  EM_EnterPallet . ResetFaults ( )
119                 AND  EM_CenterPallet . ResetFaults ( )
120                 AND  EM_ExitPallet . ResetFaults ( )  THEN
121                 bStopProcessButton  :=  FALSE ;
122                 unitManager . ActiveUnitMode . ActingStateCompleted ( ) ;
123             ELSE  RETURN ;
124             END_IF
125
126         PACK_ML . State . Idle :
127             IF  bStartProcessButton  THEN
128                 IF  EM_EnterPallet . CyclePreparation ( )
129                     AND  EM_CenterPallet . CyclePreparation ( )
130                     AND  EM_ExitPallet . CyclePreparation ( )
131                     AND  EM_WrapPallet . CyclePreparation ( )  THEN
132                         unitManager . ActiveUnitMode . Start ( ) ;
133                 ELSE  RETURN ;
134                 END_IF
135             END_IF
136
137         PACK_ML . State . Starting :
138             IF  EM_WrapPallet . Ref_FilmClamp . PW_Error . PW_ErrorID  =  PW_ERROR_ID
    . PW_FILM_NOT_DETECTED_ERROR  THEN
139                 unitManager . ActiveUnitMode . Hold ( ) ;
140             ELSE
141                 unitManager . ActiveUnitMode . ActingStateCompleted ( ) ;
142             END_IF
143
144         PACK_ML . State . Execute :
145             // The wrapping film run-out.
146             IF  EM_WrapPallet . Ref_FilmClamp . PW_Error . PW_ErrorID  =  PW_ERROR_ID
    . PW_FILM_NOT_DETECTED_ERROR  THEN
147                 unitManager . ActiveUnitMode . Hold ( ) ;
148             END_IF
149             // Business logic
150             prgCenterPallet ( Ref_CenterPallet  :=  Ref_CenterPallet ,
151                               mCenterReadyToMoveIn  :=  mCenterReadyToMoveIn ,
152                               mCenterReadyToMoveOut  :=  mCenterReadyToMoveOut ,
153                               mWrapPalletProcessDone  :=  mWrapPalletProcessDone ,
154                               mEntryReadyToMoveIn  =>  mEntryReadyToMoveIn ,
155                               mExitReadyToMoveIn  =>  mExitReadyToMoveIn ,
156                               mWrapperReadyToProcess  =>  mWrapperReadyToProcess ) ;
```

```
157
158             prgWrapPallet ( Ref_WrapPallet  :=  Ref_WrapPallet ,
159                             mWrapperReadyToProcess  :=  mWrapperReadyToProcess ,
160                             mWrapPalletProcessDone  =>  mWrapPalletProcessDone ) ;
161
162             prgExitPallet ( Ref_ExitPallet  :=  Ref_ExitPallet ,
163                             mExitReadyToMoveIn  :=  mExitReadyToMoveIn ,
164                             mCenterReadyToMoveOut  =>  mCenterReadyToMoveOut ) ;
165
166             prgEnterPallet ( Ref_EnterPallet  :=  Ref_EnterPallet ,
167                              mEntryReadyToMoveIn  :=  mEntryReadyToMoveIn ,
168                              mCenterReadyToMoveIn  =>  mCenterReadyToMoveIn ) ;
169
170         PACK_ML . State . Completing :
171             unitManager . ActiveUnitMode . ActingStateCompleted ( ) ;
172
173         PACK_ML . State . Complete :
174             unitManager . ActiveUnitMode . Reset ( ) ;
175
176         PACK_ML . State . Holding :
177             IF  EM_WrapPallet . Halt ( )
178                 AND  EM_EnterPallet . Halt ( )
179                 AND  EM_CenterPallet . Halt ( )
180                 AND  EM_ExitPallet . Halt ( )  THEN
181                 unitManager . ActiveUnitMode . ActingStateCompleted ( ) ;
182             ELSE  RETURN ;
183             END_IF
184
185         PACK_ML . State . Held :
186             IF  EM_WrapPallet . ProvideFilm ( )  THEN
187                 unitManager . ActiveUnitMode . Unhold ( ) ;
188             ELSE  RETURN ;
189             END_IF
190
191         PACK_ML . State . UnHolding :
192             unitManager . ActiveUnitMode . ActingStateCompleted ( ) ;
193
194         {* Functionality NOT implemented yet
195            PACK_ML.State.Suspending:
196                  unitManager.ActiveUnitMode.ActingStateCompleted();
197            *}
198         {* Functionality NOT implemented yet
199            PACK_ML.State.Suspended:
200                  unitManager.ActiveUnitMode.UnSuspend();
201            *}
202         {* Functionality NOT implemented yet
203            PACK_ML.State.UnSuspending:
204                  unitManager.ActiveUnitMode.ActingStateCompleted();
205            *}
206
207         PACK_ML . State . Aborting :
208             IF  EM_WrapPallet . EmergencyStop ( )
209                 AND  EM_EnterPallet . EmergencyStop ( )
210                 AND  EM_ExitPallet . EmergencyStop ( )
211                 AND  EM_CenterPallet . EmergencyStop ( )  THEN
212                 unitManager . ActiveUnitMode . ActingStateCompleted ( ) ;
```

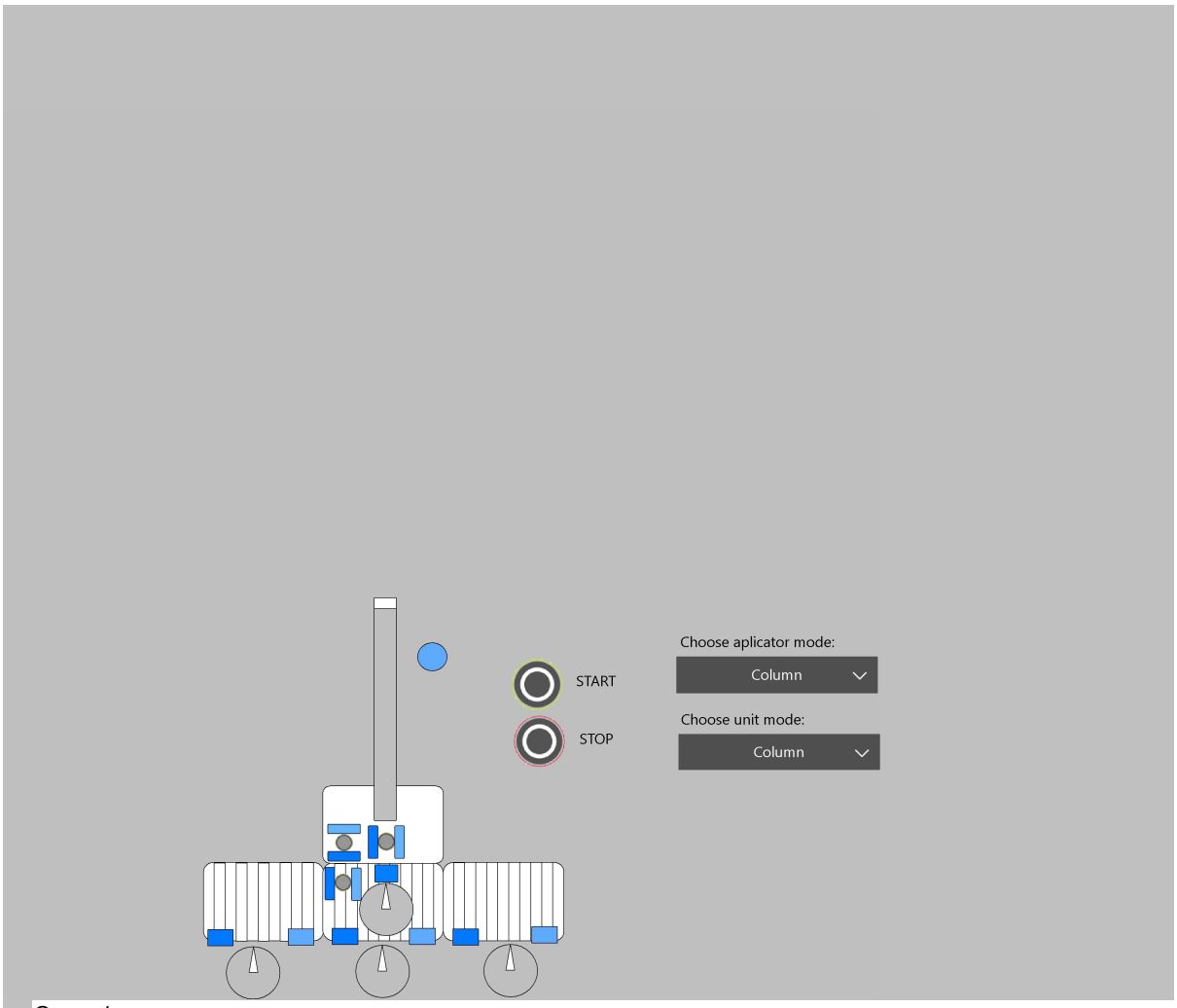
```
213             ELSE  RETURN ;
214             END_IF
215
216         PACK_ML . State . Aborted :
217             unitManager . ActiveUnitMode . Clear ( ) ;
218
219         PACK_ML . State . Clearing :
220             unitManager . ActiveUnitMode . ActingStateCompleted ( ) ;
221
222         PACK_ML . State . Stopping :
223             IF  EM_WrapPallet . Halt ( )
224                 AND  EM_EnterPallet . Halt ( )
225                 AND  EM_CenterPallet . Halt ( )
226                 AND  EM_ExitPallet . Halt ( )  THEN
227                 unitManager . ActiveUnitMode . ActingStateCompleted ( ) ;
228             ELSE  RETURN ;
229             END_IF
230     END_CASE
231     END_IF
232
233
```

1.1.1.1.6 POU: prgWrapPallet

```
1       PROGRAM  prgWrapPallet
2       VAR_INPUT
3           Ref_WrapPallet  :  REFERENCE  TO  EM_PalletWrapper ;
4           mWrapperReadyToProcess  :  BOOL ;
5       END_VAR
6       VAR
7           mProcessInit  :  BOOL  :=  FALSE ;
8           mStage1Done  :  BOOL ;
9           mStage1Interupted  :  BOOL ;
10          mStage2Done  :  BOOL ;
11          mStage3Done  :  BOOL ;
12          mStage4Done  :  BOOL ;
13
14          lrPalletHeight  :  LREAL ;
15          usintCurrentLayerNumber  :  USINT ;
16          usintCurrentWrap  :  USINT ;
17
18          mExtendProtectorDone  :  BOOL ;
19          mReleaseProtectorDone  :  BOOL ;
20          mCloseClampDone  :  BOOL ;
21          mReleaseClampDone  :  BOOL ;
22          mStickerAttachDone  :  BOOL ;
23          mStickerDetachDone  :  BOOL ;
24          mPutLayerDone  :  BOOL ;
25          mCurrentLayerDone  :  BOOL ;
26
27      END_VAR
28      VAR_OUTPUT
29          mWrapPalletProcessDone  :  BOOL ;
30      END_VAR
31
```

```
1       IF NOT mProcessInit THEN
2           mWrapPalletProcessDone := FALSE;
3           mStage1Done := FALSE;
4           mStage1Interupted := FALSE;
5           mStage2Done := FALSE;
6           mStage3Done := FALSE;
7           mStage4Done := FALSE;
8           lrPalletHeight := 1;
9           usintCurrentWrap := 1;
10          usintCurrentLayerNumber := 1;
11          mExtendProtectorDone := FALSE;
12          mReleaseProtectorDone := FALSE;
13          mCloseClampDone := FALSE;
14          mReleaseClampDone := FALSE;
15          mStickerAttachDone := FALSE;
16          mStickerDetachDone := FALSE;
17          mPutLayerDone := FALSE;
18          mCurrentLayerDone := FALSE;
19          mStage1Interupted := FALSE;
20
21          mProcessInit := TRUE;
22      ELSE
23          IF NOT mWrapperReadyToProcess AND mWrapPalletProcessDone THEN
24              mProcessInit := FALSE;
25          END_IF
26
27          IF (mStage1Done AND mStage4Done AND NOT mStage2Done AND NOT
        mStage3Done)
28              XOR (mStage1Interupted AND mStage2Done AND mStage3Done AND
        mStage4Done) THEN
29                  mWrapPalletProcessDone := TRUE;
30          ELSE mWrapPalletProcessDone := FALSE;
31          END_IF
32
33          //STAGE 1
34          IF mWrapperReadyToProcess
35              AND NOT mStage1Done
36              AND NOT mStage1Interupted
37              AND NOT mStage2Done
38              AND NOT mStage3Done
39              AND NOT mStage4Done THEN
40              Ref_WrapPallet.ProcessStage1(usintCurrentWrap := usintCurrentWrap
        ,
41                                           usintCurrentLayerNumber :=
        usintCurrentLayerNumber,
42                                           mExtendProtectorDone :=
        mExtendProtectorDone,
43                                           mReleaseProtectorDone :=
        mReleaseProtectorDone,
44                                           mCloseClampDone := mCloseClampDone,
45                                           mReleaseClampDone := mReleaseClampDone,
46                                           mStickerAttachDone :=
        mStickerAttachDone,
47                                           mStickerDetachDone :=
        mStickerDetachDone,
48                                           mPutLayerDone := mPutLayerDone,
49                                           mStage1Done := mStage1Done,
```

```
50                                      mStage1Interupted  :=  mStage1Interupted ,
51                                      mCurrentLayerDone  :=  mCurrentLayerDone ,
52                                      lrPalletHeight  :=  lrPalletHeight ) ;
53          END_IF
54          //STAGE 2
55          IF  mWrapperReadyToProcess
56              AND  mStage1Interupted
57              AND  NOT  mStage1Done
58              AND  NOT  mStage2Done
59              AND  NOT  mStage3Done
60              AND  NOT  mStage4Done  THEN
61              Ref_WrapPallet . ProcessStage2 ( usintCurrentLayerNumber  :=
    usintCurrentLayerNumber ,
62                                       usintCurrentWrap  :=  usintCurrentWrap ,
63                                       mStage2Done  =>  mStage2Done ) ;
64          END_IF
65          //STAGE 3
66          IF  mWrapperReadyToProcess
67              AND  mStage2Done
68              AND  NOT  mStage1Done
69              AND  NOT  mStage3Done
70              AND  NOT  mStage4Done  THEN
71              Ref_WrapPallet . ProcessStage3 ( usintCurrentLayerNumber  :=
    usintCurrentLayerNumber ,
72                                      mStage3Done  :=  mStage3Done ,
73                                      lrPalletHeight  :=  lrPalletHeight ) ;
74          END_IF
75          //STAGE 4
76          IF  mWrapperReadyToProcess
77              AND  NOT  mStage4Done
78              AND  ( ( mStage2Done  AND  mStage3Done ) XOR ( NOT  mStage1Interupted  AND
     mStage1Done ) )  THEN
79              Ref_WrapPallet . ProcessStage4 ( lrPalletHeight  :=  lrPalletHeight ,
80                                      mStage1Interupted  :=  mStage1Interupted ,
81                                      usintCurrentWrap  :=  usintCurrentWrap ,
82                                      usintCurrentLayerNumber  :=
    usintCurrentLayerNumber ,
83                                      mExtendProtectorDone  :=
    mExtendProtectorDone ,
84                                      mReleaseProtectorDone  :=
    mReleaseProtectorDone ,
85                                      mCloseClampDone  :=  mCloseClampDone ,
86                                      mReleaseClampDone  :=  mReleaseClampDone ,
87                                      mStickerAttachDone  :=
    mStickerAttachDone ,
88                                      mStickerDetachDone  :=
    mStickerDetachDone ,
89                                      mPutLayerDone  :=  mPutLayerDone ,
90                                      mStage4Done  :=  mStage4Done ,
91                                      mCurrentLayerDone  :=  mCurrentLayerDone )
    ;
92          END_IF
93      END_IF
94
```



General
Visualization size algorithm version: Respecting scrollbar location
Background
Use background color: True
Background color: -4144960
Interface
vAR_IN_OUT

END_VAR
Visualization Element List
Rounded Rectangle Id: 0
Element name: GenElemInst_2
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 272
Y: 1025
Width: 164

Height: 106
Angle: 0
Radius setting
Radius: Relative to the element size
Value: 5
Center
X: 354
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False

- Rounded Rectangle Id: 3
  - Element name: GenElemInst_3
  - Tab Order: default
  - Static optimized: True
  - Type of element: Rounded Rectangle
  - Position
    - X: 435
    - Y: 1025
    - Width: 164
    - Height: 106
    - Angle: 0
  - Radius setting
    - Radius: Explicit
    - Value: 15
  - Center
    - X: 517
    - Y: 1078
  - Colors
    - Normal state
      - Frame color
        - Transparency: 255
      - Fill color
        - Transparency: 255
    - Alarm state
      - Frame color
        - Transparency: 255
      - Fill color
        - Transparency: 255
  - Use gradient color: False
  - Gradient setting
    - Gradient
      - Color1: Color [Black]
      - Color2: Color [White]
      - Brightnesscolor: Color [Black]
  - Appearance
    - Line width: 1
    - Fill attributes: Filled
    - Line style: Solid
  - Text properties
    - Horizontal alignment: Centered
    - Vertical alignment: Centered
    - Text format: Default
    - Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
    - Font color
      - Transparency: 255
  - Absolute movement
    - Use REAL values: False
  - Font variables
    - Size: , <pt>
  - Color variables
    - Toggle color: <toggle/tap variable>
  - Animation duration: 0
  - Input configuration
    - OnDialogClosed: Configure...

OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 5
Element name: GenElemInst_4
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 599
Y: 1025
Width: 164
Height: 106
Angle: 0
Radius setting
Radius: Relative to the element size
Value: 5
Center
X: 681
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255

Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Frame Id: 21
Element name: GenElemInst_15
Type of element: Frame -> SM3_Basic.RotDrive
Tab Order: default
Static optimized: True
Clipping: False
Show frame: No frame
Scaling type: Anisotropic
Deactivate the background drawing: False
Swiping behavior: Not swipable
Swiping preview: True
References
SM3_Basic.RotDrive
m_Input_AXISREF: EnterPalletAxis
Position
X: 304
Y: 1139
Width: 73
Height: 71
Center
X: 340
Y: 1174
Colors
Color
Transparency: 255
Alarm color
Transparency: 255
Appearance
Line width: 1
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color

Transparency: 255
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Frame Id: 23
Element name: GenElemInst_16
Type of element: Frame -> SM3_Basic.RotDrive
Tab Order: default
Static optimized: True
Clipping: False
Show frame: No frame
Scaling type: Anisotropic
Deactivate the background drawing: False
Swiping behavior: Not swipable
Swiping preview: True
References
SM3_Basic.RotDrive
m_Input_AXISREF: CenterPalletAxis
Position
X: 480
Y: 1137
Width: 73
Height: 71
Center
X: 516
Y: 1172
Colors
Color
Transparency: 255
Alarm color
Transparency: 255
Appearance
Line width: 1
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255

Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Frame Id: 25
Element name: GenElemInst_17
Type of element: Frame -> SM3_Basic.RotDrive
Tab Order: default
Static optimized: True
Clipping: False
Show frame: No frame
Scaling type: Anisotropic
Deactivate the background drawing: False
Swiping behavior: Not swipable
Swiping preview: True
References
SM3_Basic.RotDrive
m_Input_AXISREF: ExitPalletAxis
Position
X: 654
Y: 1136
Width: 73
Height: 71
Center
X: 690
Y: 1171
Colors
Color
Transparency: 255
Alarm color
Transparency: 255
Appearance
Line width: 1
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Font variables

Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 29
Element name: GenElemInst_19
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 435
Y: 920
Width: 164
Height: 106
Angle: 0
Radius setting
Radius: Explicit
Value: 10
Center
X: 517
Y: 973
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties

Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Frame Id: 37
Element name: GenElemInst_24
Type of element: Frame -> SM3_Basic.LinDrive_V
Tab Order: default
Static optimized: False
Clipping: False
Show frame: No frame
Scaling type: Anisotropic
Deactivate the background drawing: False
Swiping behavior: Not swipable
Swiping preview: True
References
SM3_Basic.LinDrive_V
m_Input_AXISREF: FilmAplicatorAxis
Position
X: 505
Y: 664
Width: 30
Height: 304
Center
X: 520
Y: 817
Colors
Color
Transparency: 255
Alarm color
Transparency: 255
Appearance
Line width: 1
Line style: Solid

Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Interior rotation: 180
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Lamp1 Id: 38
Element name: GenElemInst_25
Type of element: Lamp1
Tab Order: default
Static optimized: False
Position
X: 509
Y: 984
Width: 26
Height: 26
Variable: prgMain.EM_WrapPallet.Ref_FilmClamp.bExtensionValve
Image settings
Transparent: False
Transparent color: Black
Isotropic type: Isotropic
Horizontal alignment: Left
Vertical alignment: Top
Center
X: 522
Y: 988
Animation duration: 0
Background
Image: Green
Lamp1 Id: 40
Element name: GenElemInst_26
Type of element: Lamp1
Tab Order: default
Static optimized: False
Position

X: 452
Y: 986
Width: 25
Height: 27
Variable: prgMain.EM_WrapPallet.Ref_FilmSticker.bExtensionValve
Image settings
Transparent: False
Transparent color: Black
Isotropic type: Isotropic
Horizontal alignment: Left
Vertical alignment: Top
Center
X: 172
Y: -147
Animation duration: 0
Background
Image: Green
Rounded Rectangle Id: 48
Element name: GenElemInst_50
Tab Order: default
Static optimized: False
Type of element: Rounded Rectangle
Position
X: 442
Y: 1011
Width: 44
Height: 13
Angle: 179
Radius setting
Radius: Relative to the element size
Value: 5
Center
X: 464
Y: 1017
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled

Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_WrapPallet.Ref_FilmSticker.bRetractionSensor
Tap
Tap FALSE: False
Rounded Rectangle Id: 50
Element name: GenElemInst_49
Tab Order: default
Static optimized: False
Type of element: Rounded Rectangle
Position
X: 442
Y: 973
Width: 44
Height: 13
Angle: 0
Radius setting
Radius: Relative to the element size
Value: 5
Center
X: 464
Y: 979
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255

Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_WrapPallet.Ref_FilmSticker.bExtensionSensor
Tap
Tap FALSE: False
Rounded Rectangle Id: 54
Element name: GenElemInst_52
Tab Order: default
Static optimized: False
Type of element: Rounded Rectangle
Position
X: 517
Y: 991
Width: 44
Height: 13
Angle: 270
Radius setting
Radius: Relative to the element size
Value: 5
Center

X: 539
Y: 997
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_WrapPallet.Ref_FilmClamp.bExtensionSensor
Tap
Tap FALSE: False
Rounded Rectangle Id: 62
Element name: GenElemInst_51
Tab Order: default
Static optimized: False

Type of element: Rounded Rectangle
Position
  X: 481
  Y: 991
  Width: 44
  Height: 13
  Angle: 270
Radius setting
  Radius: Relative to the element size
  Value: 5
Center
  X: 503
  Y: 997
Colors
  Normal state
    Frame color
      Transparency: 255
    Fill color
      Transparency: 255
  Alarm state
    Frame color
      Transparency: 255
    Fill color
      Transparency: 255
Use gradient color: False
Gradient setting
  Gradient
    Color1: Color [Black]
    Color2: Color [White]
    Brightnesscolor: Color [Black]
Appearance
  Line width: 1
  Fill attributes: Filled
  Line style: Solid
Text properties
  Horizontal alignment: Centered
  Vertical alignment: Centered
  Text format: Default
  Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
  Font color
    Transparency: 255
Absolute movement
  Use REAL values: False
Font variables
  Size: , <pt>
Color variables
  Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
  OnDialogClosed: Configure...
  OnMouseClick: Configure...
  OnMouseDown: Configure...
  OnMouseEnter: Configure...
  OnMouseLeave: Configure...

OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_WrapPallet.Ref_FilmClamp.bRetractionSensor
Tap
Tap FALSE: False
Ellipse Id: 78
Element name: GenElemInst_107
Tab Order: default
Static optimized: False
Type of element: Ellipse
Position
X: 564
Y: 725
Width: 40
Height: 38
Angle: 0
Radius setting
Radius: From style
Value: 5
Center
X: 584
Y: 744
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False

Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_WrapPallet.Ref_FilmAplicator.HeightSensor
Tap
Tap FALSE: False
Rounded Rectangle Id: 80
Element name: GenElemInst_108
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 241
Y: 1071
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 294
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1

Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rectangle Id: 65
Element name: GenElemInst_40
Tab Order: default
Static optimized: False
Type of element: Rectangle
Position
X: 278
Y: 1116
Width: 35
Height: 22
Angle: 0
Radius setting
Radius: From style
Value: 5
Center
X: 295
Y: 1127
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color

Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_EnterPallet.bDiffuseEntrySensor
Tap
Tap FALSE: False
Rounded Rectangle Id: 82
Element name: GenElemInst_53
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 301
Y: 1071
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 354

Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 84
Element name: GenElemInst_54
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 271

Y: 1071
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 324
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...

Tap
Tap FALSE: False
Rounded Rectangle Id: 86
Element name: GenElemInst_55
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 361
Y: 1071
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 414
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0

Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 88
Element name: GenElemInst_56
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 331
Y: 1071
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 384
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]

Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 90
Element name: GenElemInst_57
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 525
Y: 1072
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 578
Y: 1079
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]

Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 92
Element name: GenElemInst_58
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 405
Y: 1072
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 458
Y: 1079
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color

Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 94
Element name: GenElemInst_59
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 466
Y: 1072
Width: 105
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 518

Y: 1079
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 96
Element name: GenElemInst_60
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 437

Y: 1072
Width: 105
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 489
Y: 1079
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...

Tap
Tap FALSE: False
Rounded Rectangle Id: 98
Element name: GenElemInst_61
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 496
Y: 1072
Width: 105
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 548
Y: 1079
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0

Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rectangle Id: 67
Element name: GenElemInst_41
Tab Order: default
Static optimized: False
Type of element: Rectangle
Position
X: 388
Y: 1115
Width: 35
Height: 22
Angle: 0
Radius setting
Radius: From style
Value: 5
Center
X: 405
Y: 1126
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]

Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_EnterPallet.bDiffuseExitSensor
Tap
Tap FALSE: False
Rounded Rectangle Id: 120
Element name: GenElemInst_72
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 567
Y: 1071
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 620
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]

Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 122
Element name: GenElemInst_73
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 687
Y: 1071
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 740
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255

Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 124
Element name: GenElemInst_74
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 627
Y: 1071
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5

Center
X: 680
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 126
Element name: GenElemInst_75
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle

Position
  X: 597
  Y: 1071
  Width: 106
  Height: 15
  Angle: 270
Radius setting
  Radius: From style
  Value: 5
Center
  X: 650
  Y: 1078
Colors
  Normal state
    Frame color
      Transparency: 255
    Fill color
      Transparency: 255
  Alarm state
    Frame color
      Transparency: 255
    Fill color
      Transparency: 255
Use gradient color: False
Gradient setting
  Gradient
    Color1: Color [Black]
    Color2: Color [White]
    Brightnesscolor: Color [Black]
Appearance
  Line width: 1
  Fill attributes: Filled
  Line style: Solid
Text properties
  Horizontal alignment: Centered
  Vertical alignment: Centered
  Text format: Default
  Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
  Font color
    Transparency: 255
Absolute movement
  Use REAL values: False
Font variables
  Size: , <pt>
Color variables
  Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
  OnDialogClosed: Configure...
  OnMouseClick: Configure...
  OnMouseDown: Configure...
  OnMouseEnter: Configure...
  OnMouseLeave: Configure...
  OnMouseMove: Configure...

OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 128
Element name: GenElemInst_76
Tab Order: default
Static optimized: True
Type of element: Rounded Rectangle
Position
X: 657
Y: 1071
Width: 106
Height: 15
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 710
Y: 1078
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables

Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rectangle Id: 71
Element name: GenElemInst_43
Tab Order: default
Static optimized: False
Type of element: Rectangle
Position
X: 553
Y: 1114
Width: 35
Height: 22
Angle: 0
Radius setting
Radius: From style
Value: 5
Center
X: 570
Y: 1125
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered

Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_CenterPallet.bDiffuseExitSensor
Tap
Tap FALSE: False
Lamp1 Id: 43
Element name: GenElemInst_28
Type of element: Lamp1
Tab Order: default
Static optimized: False
Position
X: 451
Y: 1040
Width: 25
Height: 29
Variable: prgMain.EM_WrapPallet.Ref_PalletProtector.bExtensionValve
Image settings
Transparent: False
Transparent color: Black
Isotropic type: Isotropic
Horizontal alignment: Left
Vertical alignment: Top
Center
X: 470
Y: 1058
Animation duration: 0
Background
Image: Green
Rounded Rectangle Id: 56
Element name: GenElemInst_47
Tab Order: default
Static optimized: False
Type of element: Rounded Rectangle
Position
X: 458

Y: 1048
Width: 44
Height: 13
Angle: 270
Radius setting
Radius: Relative to the element size
Value: 5
Center
X: 480
Y: 1054
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...

Toggle
Variable: prgMain.EM_WrapPallet.Ref_PalletProtector.bExtensionSensor
Tap
Tap FALSE: False
Rounded Rectangle Id: 58
Element name: GenElemInst_46
Tab Order: default
Static optimized: False
Type of element: Rounded Rectangle
Position
X: 422
Y: 1047
Width: 44
Height: 13
Angle: 270
Radius setting
Radius: Relative to the element size
Value: 5
Center
X: 444
Y: 1053
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables

Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_WrapPallet.Ref_PalletProtector.bRetractionSensor
Tap
Tap FALSE: False
Frame Id: 60
Element name: GenElemInst_38
Type of element: Frame -> SM3_Basic.RotDrive
Tab Order: default
Static optimized: True
Clipping: False
Show frame: No frame
Scaling type: Anisotropic
Deactivate the background drawing: False
Swiping behavior: Not swipable
Swiping preview: True
References
SM3_Basic.RotDrive
m_Input_AXISREF: PalletRotatorAxis
Position
X: 485
Y: 1053
Width: 73
Height: 71
Center
X: 521
Y: 1088
Colors
Color
Transparency: 255
Alarm color
Transparency: 255
Appearance
Line width: 1
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Font variables
Size: , <pt>

Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Tap
Tap FALSE: False
Rounded Rectangle Id: 64
Element name: GenElemInst_48
Tab Order: default
Static optimized: False
Type of element: Rounded Rectangle
Position
X: 510
Y: 1025
Width: 23
Height: 31
Angle: 270
Radius setting
Radius: From style
Value: 5
Center
X: 521
Y: 1040
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered

Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_WrapPallet.Ref_FilmClamp.bPresenceSensor
Tap
Tap FALSE: False
Rectangle Id: 69
Element name: GenElemInst_42
Tab Order: default
Static optimized: False
Type of element: Rectangle
Position
X: 448
Y: 1114
Width: 35
Height: 22
Angle: 0
Radius setting
Radius: From style
Value: 5
Center
X: 465
Y: 1125
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False

Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_CenterPallet.bDiffuseEntrySensor
Tap
Tap FALSE: False
Rectangle Id: 73
Element name: GenElemInst_44
Tab Order: default
Static optimized: False
Type of element: Rectangle
Position
X: 612
Y: 1115
Width: 35
Height: 22
Angle: 0
Radius setting
Radius: From style
Value: 5
Center
X: 629
Y: 1126
Colors

Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...
Toggle
Variable: prgMain.EM_ExitPallet.bDiffuseEntrySensor
Tap
Tap FALSE: False
Rectangle Id: 75
Element name: GenElemInst_45
Tab Order: default
Static optimized: False
Type of element: Rectangle
Position
X: 719

Y: 1112
Width: 35
Height: 22
Angle: 0
Radius setting
Radius: From style
Value: 5
Center
X: 736
Y: 1123
Colors
Normal state
Frame color
Transparency: 255
Fill color
Transparency: 255
Alarm state
Frame color
Transparency: 255
Fill color
Transparency: 255
Use gradient color: False
Gradient setting
Gradient
Color1: Color [Black]
Color2: Color [White]
Brightnesscolor: Color [Black]
Appearance
Line width: 1
Fill attributes: Filled
Line style: Solid
Text properties
Horizontal alignment: Centered
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Absolute movement
Use REAL values: False
Font variables
Size: , <pt>
Color variables
Toggle color: <toggle/tap variable>
Animation duration: 0
Input configuration
OnDialogClosed: Configure...
OnMouseClick: Configure...
OnMouseDown: Configure...
OnMouseEnter: Configure...
OnMouseLeave: Configure...
OnMouseMove: Configure...
OnMouseUp: Configure...
OnValueChanged: Configure...

Toggle
Variable: prgMain.EM_ExitPallet.bDiffuseExitSensor
Tap
Tap FALSE: False
Label Id: 137
Element name: GenElemInst_84
Type of element: Label
Tab Order: default
Static optimized: True
Position
X: 773
Y: 712
Width: 150
Height: 30
Text properties
Horizontal alignment: Left
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;12;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Center
X: 848
Y: 727
Animation duration: 0
Label Id: 151
Element name: GenElemInst_93
Type of element: Label
Tab Order: default
Static optimized: True
Text ID: 723
Texts
Text: START
Position
X: 776
Y: 762
Width: 94
Height: 30
Text properties
Horizontal alignment: Left
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;20;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Center
X: 823
Y: 777
Animation duration: 0
PushSwitch Id: 155
Element name: GenElemInst_98
Type of element: PushSwitch
Tab Order: default
Static optimized: False

Position
X: 692
Y: 747
Width: 70
Height: 70
Variable: prgMain.bStartProcessButton
Image settings
Transparent: False
Transparent color: Black
Isotropic type: Isotropic
Horizontal alignment: Left
Vertical alignment: Top
Element behavior: Image toggler
Tap FALSE: False
Center
X: 727
Y: 782
Animation duration: 0
Input configuration
OnValueChanged: Configure...
Background
Image: Green
PushSwitch Id: 157
Element name: GenElemInst_99
Type of element: PushSwitch
Tab Order: default
Static optimized: False
Position
X: 695
Y: 824
Width: 70
Height: 70
Variable: prgMain.bStopProcessButton
Image settings
Transparent: False
Transparent color: Black
Isotropic type: Isotropic
Horizontal alignment: Left
Vertical alignment: Top
Element behavior: Image toggler
Tap FALSE: False
Center
X: 730
Y: 859
Animation duration: 0
Input configuration
OnValueChanged: Configure...
Background
Image: Red
Label Id: 159
Element name: GenElemInst_100
Type of element: Label
Tab Order: default
Static optimized: True

Text ID: 652
Texts
Text: STOP
Position
X: 781
Y: 840
Width: 94
Height: 30
Text properties
Horizontal alignment: Left
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;20;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Center
X: 828
Y: 855
Animation duration: 0
ComboBoxTable Id: 195
Element name: GenElemInst_127
Type of element: ComboBoxTable
Tab Order: default
Static optimized: False
Tooltip ID: 309
Position
X: 917
Y: 744
Width: 274
Height: 50
Variable: prgMain.intDesiredAplicatorModeNameIndex
Data array: AplicatorModeNames.arrModeNames
Columns
Column
[0]
Visible: True
Width: 274
Image column: False
Image configuration
Fill mode: Centered
Transparent: True
Transparent color: White
Text alignment of column: Centered
Use template: True
Template
Type of element: Rectangle
Angle: 0
Radius setting
Radius: Relative to the element size
Value: 5
Colors
Normal state
Frame color
Transparency: 255

```
          Fill color
            Transparency: 255
        Alarm state
          Frame color
            Transparency: 255
          Fill color
            Transparency: 255
      Use gradient color: False
      Gradient setting
        Gradient
          Color1: Color [Black]
          Color2: Color [White]
          Brightnesscolor: Color [Black]
      Appearance
        Line width: 1
        Fill attributes: Filled
        Line style: Solid
      Texts
        Text: %s
      Text properties
        Horizontal alignment: Centered
        Vertical alignment: Centered
        Text format: Default
        Font: Segoe UI;20;Regular;Color [A=255, R=0, G=0, B=0]
        Font color
          Transparency: 255
      Text variables
        Text variable: AplicatorModeNames.arrModeNames[INDEX]
      Font variables
        Size: , <pt>
      Input configuration
        OnDialogClosed: Configure...
        OnMouseClick: Configure...
        OnMouseDown: Configure...
        OnMouseEnter: Configure...
        OnMouseLeave: Configure...
        OnMouseMove: Configure...
        OnMouseUp: Configure...
        OnValueChanged: Configure...
        Tap
          Tap FALSE: False
  Maximum array index: 3
  Row height: 40
  Number of visible rows: 3
  Scroll Bar size: 0
  Texts
    Tooltip: Choose aplicator mode.
  Text properties
    Usage of: Individual settings
    Individual text properties
      Font: Segoe UI;20;Regular;White, Color [A=255, R=255, G=255, B=255]
      Font color
        Transparency: 255
    Individual font variables
```

```
      Size: , <pt>
    Individual selection text properties
      Font: Segoe UI;20;Regular;White, Color [A=255, R=255, G=255, B=255]
      Font color
        Transparency: 255
    Individual selection font variables
      Size: , <pt>
  State variables
    Deactivate inputs: prgMain.bChangeModesNotAllowed
  Center
    X: 1037
    Y: 773
  Animation duration: 0
  Input configuration
    OnValueChanged: Configure...
ComboBoxTable Id: 200
  Element name: GenElemInst_131
  Type of element: ComboBoxTable
  Tab Order: default
  Static optimized: False
  Tooltip ID: 253
  Position
    X: 920
    Y: 849
    Width: 274
    Height: 50
  Variable: prgMain.intDesiredUnitModeNameIndex
  Data array: arrUnitModeNames
  Columns
    Column
      [0]
        Visible: True
        Width: 274
        Image column: False
        Image configuration
          Fill mode: Centered
          Transparent: True
          Transparent color: White
        Text alignment of column: Centered
        Use template: True
        Template
          Type of element: Rectangle
          Angle: 0
          Radius setting
            Radius: Relative to the element size
            Value: 5
          Colors
            Normal state
              Frame color
                Transparency: 255
              Fill color
                Transparency: 255
            Alarm state
              Frame color
```

```
            Transparency: 255
          Fill color
            Transparency: 255
        Use gradient color: False
        Gradient setting
          Gradient
            Color1: Color [Black]
            Color2: Color [White]
            Brightnesscolor: Color [Black]
        Appearance
          Line width: 1
          Fill attributes: Filled
          Line style: Solid
        Texts
          Text: %s
        Text properties
          Horizontal alignment: Centered
          Vertical alignment: Centered
          Text format: Default
          Font: Segoe UI;20;Regular;Color [A=255, R=0, G=0, B=0]
          Font color
            Transparency: 255
        Text variables
          Text variable: arrUnitModeNames[INDEX]
        Font variables
          Size: , <pt>
        Input configuration
          OnDialogClosed: Configure...
          OnMouseClick: Configure...
          OnMouseDown: Configure...
          OnMouseEnter: Configure...
          OnMouseLeave: Configure...
          OnMouseMove: Configure...
          OnMouseUp: Configure...
          OnValueChanged: Configure...
          Tap
            Tap FALSE: False
Maximum array index: 2
Row height: 40
Number of visible rows: 2
Scroll Bar size: 0
Texts
  Tooltip: Choose unit mode.
Text properties
  Usage of: Individual settings
  Individual text properties
    Font: Segoe UI;20;Regular;White, Color [A=255, R=255, G=255, B=255]
    Font color
      Transparency: 255
  Individual font variables
    Size: , <pt>
  Individual selection text properties
    Font: Segoe UI;20;Regular;White, Color [A=255, R=255, G=255, B=255]
    Font color
```

Transparency: 255
Individual selection font variables
Size: , <pt>
State variables
Deactivate inputs: prgMain.bChangeModesNotAllowed
Center
X: 1040
Y: 878
Animation duration: 0
Input configuration
OnValueChanged: Configure...
Label Id: 201
Element name: GenElemInst_132
Type of element: Label
Tab Order: default
Static optimized: True
Text ID: 836
Texts
Text: Choose aplicator mode:
Position
X: 919
Y: 709
Width: 250
Height: 30
Text properties
Horizontal alignment: Left
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;20;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color
Transparency: 255
Center
X: 1044
Y: 724
Animation duration: 0
Label Id: 203
Element name: GenElemInst_133
Type of element: Label
Tab Order: default
Static optimized: True
Text ID: 10
Texts
Text: Choose unit mode:
Position
X: 920
Y: 814
Width: 250
Height: 30
Text properties
Horizontal alignment: Left
Vertical alignment: Centered
Text format: Default
Font: Segoe UI;20;Regular;Font-Default-Color, Color [A=255, R=0, G=0, B=0]
Font color

Transparency: 255
Center
X: 1045
Y: 829
Animation duration: 0