Project Documentation

File: CtrlX_Module_04_2025.project

Date: 24.02.2025

Profile: ctrlX PLC 1.20.7

Table des matières

1 DUT: E_ExecuteGripper

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ExecuteGripper :
(
    Idle    := 999,
    Init    := 10,
    InOp    := 20,
    Done    := 30,
    Error   := 40
) DINT := Idle;
END_TYPE

```

2 DUT: E_InOperationBaseGripper

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_InOperationBaseGripper :
(
    Idle    := 99,
    Init    := 1,
    InOp     := 4,
    Error    := 0
) DINT := Idle;
END_TYPE

```

3 DUT: E_InOpGripper

```
{attribute 'qualified_only' := ''}
{attribute 'strict' := ''}
TYPE E_InOpGripper :
(
    IsIdle  := 999,
    IsOpen  := 10,
    IsClosed  := 20,
    PartPresent  := 30
);
END_TYPE

```

4 DUT: ST_TestFbGripperHmi

```
TYPE ST_TestFbGripperHmi :
STRUCT
    openGripper             : BOOL;
    closeGripper            : BOOL;
    executeOpenDone         : BOOL;
    executeCloseDone        : BOOL;
    gripperStateClosed      : BOOL;
    gripperStateOpen        : BOOL;
    gripperStatePartPresent : BOOL;
    gripperStateError       : BOOL;
    gripperStateInOp        : BOOL;
END_STRUCT
END_TYPE

```

5 POU: FB_CloseGripper

```
FUNCTION_BLOCK FB_CloseGripper
VAR_INPUT
    Execute : BOOL;
    thClosedMin : WORD := 950;
END_VAR
VAR_IN_OUT
    hwSensor : UA_Schunk_mms;
    hwEV : UA_Festo;
END_VAR
VAR_OUTPUT
    Done : BOOL;
    Active : BOOL;
    Error : BOOL;
END_VAR
VAR
    tnCheckDone : TON;
    rExecute : R_TRIG;
    eExecuteGripper : E_ExecuteGripper;
END_VAR

```

```
(*
      Manage Inputs
*)
rExecute(CLK := Execute);
// Timer.Q true if Execute and not threshold for more thant 1 sec.
tnCheckDone(IN := Execute AND NOT (hwSensor.Value > thClosedMin),
            PT := T#1S);

CASE eExecuteGripper OF
    E_ExecuteGripper.Idle :
        IF rExecute.Q THEN
            eExecuteGripper := E_ExecuteGripper.Init;
        END_IF
```

```
14
15          E_ExecuteGripper . Init  :
16              // No init
17              IF  tnCheckDone . Q  THEN
18                  eExecuteGripper  :=  E_ExecuteGripper . Error ;
19              ELSE
20                  eExecuteGripper  :=  E_ExecuteGripper . InOp ;
21              END_IF
22
23          E_ExecuteGripper . InOp  :
24              IF  tnCheckDone . Q  THEN
25                  eExecuteGripper  :=  E_ExecuteGripper . Error ;
26              ELSIF  ( hwSensor . Value  >  thClosedMin )  THEN
27                  eExecuteGripper  :=  E_ExecuteGripper . Done ;
28              END_IF
29
30          E_ExecuteGripper . Done  :
31              IF  NOT  Execute  THEN
32                  eExecuteGripper  :=  E_ExecuteGripper . Idle ;
33              END_IF
34
35          E_ExecuteGripper . Error  :
36              IF  NOT  Execute  THEN
37                  eExecuteGripper  :=  E_ExecuteGripper . Idle ;
38              END_IF
39      END_CASE
40
41      IF  eExecuteGripper  =  E_ExecuteGripper . InOp  THEN
42          hwEV . SetOut  :=  FALSE ;
43      END_IF
44
45      Done  :=  ( eExecuteGripper  =  E_ExecuteGripper . Done ) ;
46      Active  :=  ( eExecuteGripper  =  E_ExecuteGripper . Init )  OR  ( eExecuteGripper
         =  E_ExecuteGripper . InOp )  ;
47      Error  :=  ( eExecuteGripper  =  E_ExecuteGripper . Error ) ;
48
```

6 POU: FB_GripperState

```
1       FUNCTION_BLOCK  FB_GripperState
2       VAR_INPUT
3           /// Default Input
4           Enable :  BOOL ;
5           /// User Defined Inputs
6           thOpen :  WORD  :=  50 ;
7           thClose :  WORD  :=  950 ;
8           thPartMin :  WORD  :=  800 ;
9           thPartMax :  WORD  :=  860 ;
10      END_VAR
11      VAR_IN_OUT
12          hw :  UA_Schunk_mms ;
13      END_VAR
14      VAR_OUTPUT
15          /// Default Outputs
16          InOperation :  BOOL ;
```

```
17          Error : BOOL ;
18          /// User Outputs
19          IsOpen : BOOL ;
20          IsClosed : BOOL ;
21          PartPresent : BOOL ;
22      END_VAR
23      VAR
24          eInOperationBaseGripper : E_InOperationBaseGripper :=
        E_InOperationBaseGripper . Idle ;
25          eInOpGripper : E_InOpGripper := E_InOpGripper . IsIdle ;
26          tonIdleCondition : TON ;
27      END_VAR
28
```

```
1       (*
2           Input management. While gripper is moving, it must not be seen as an
        Idle Condition
3           Here we use a timer as Input Management.
4       *)
5
6       tonIdleCondition ( IN := ( ( ( hw . Value > thOpen ) AND ( hw . Value <
        thPartMin ) ) OR
7                                ( ( hw . Value < thClose ) AND ( hw . Value >
        thPartMax ) ) ) ,
8                           PT := T#1S ) ;
9
10      (*
11          Main State Machine
12      *)
13      CASE eInOperationBaseGripper OF
14          E_InOperationBaseGripper . Idle :
15              IF Enable THEN
16                  eInOperationBaseGripper := E_InOperationBaseGripper . Init ;
17              END_IF
18
19          E_InOperationBaseGripper . Init :
20              IF NOT Enable THEN
21                  eInOperationBaseGripper := E_InOperationBaseGripper . Idle ;
22              ELSIF tonIdleCondition . Q THEN
23                  eInOpGripper := E_InOpGripper . IsIdle ;
24                  eInOperationBaseGripper := E_InOperationBaseGripper . Error ;
25              ELSE // Init internal state machine and jump in it
26                  IF hw . Value < thOpen THEN
27                      eInOpGripper := E_InOpGripper . IsOpen ;
28                  ELSIF hw . Value > thClose THEN
29                      eInOpGripper := E_InOpGripper . IsClosed ;
30                  ELSE
31                      eInOpGripper := E_InOpGripper . PartPresent ;
32                  END_IF
33                  eInOperationBaseGripper := E_InOperationBaseGripper . InOp ;
34              END_IF
35
36          E_InOperationBaseGripper . InOp :
37              (*
38                          Sub State Machine Here
```

```
39                  *)
40              // Machine initilized in E_InOperationBase.Init
41              // Error condition again, air pressure could be removed while
    machine is running
42              IF NOT Enable THEN
43                  eInOperationBaseGripper := E_InOperationBaseGripper.Idle;
44              ELSIF tonIdleCondition.Q THEN          // Error condition
45                  eInOpGripper := E_InOpGripper.IsIdle;
46                  eInOperationBaseGripper := E_InOperationBaseGripper.Error;
47              ELSE // Init internal state machine and jump in it
48                  CASE eInOpGripper OF
49                      E_InOpGripper.IsIdle :
50                          ; // Do nothing, condition is checked in IF
51                      E_InOpGripper.IsOpen :
52                          IF hw.Value > thClose THEN
53                              eInOpGripper := E_InOpGripper.IsClosed;
54                          ELSIF (hw.Value > thPartMin) AND (hw.Value <
    thPartMax) THEN
55                              eInOpGripper := E_InOpGripper.PartPresent;
56                          END_IF
57                      E_InOpGripper.IsClosed :
58                          IF hw.Value < thOpen THEN
59                              eInOpGripper := E_InOpGripper.IsOpen;
60                          END_IF
61                      E_InOpGripper.PartPresent :
62                          IF hw.Value < thOpen THEN
63                              eInOpGripper := E_InOpGripper.IsOpen;
64                          END_IF
65                  END_CASE
66              END_IF
67          E_InOperationBaseGripper.Error :
68              IF NOT Enable THEN
69                  eInOperationBaseGripper := E_InOperationBaseGripper.Idle;
70              END_IF
71      END_CASE
72
73      (*
74              Output Management
75      *)
76      InOperation         := (eInOperationBaseGripper = E_InOperationBaseGripper
        .InOp);
77      Error               := (eInOperationBaseGripper = E_InOperationBaseGripper
        .Error);
78
79      IsOpen              := (eInOperationBaseGripper = E_InOperationBaseGripper
        .InOp) AND (eInOpGripper = E_InOpGripper.IsOpen);
80      IsClosed            := (eInOperationBaseGripper = E_InOperationBaseGripper
        .InOp) AND (eInOpGripper = E_InOpGripper.IsClosed);
81      PartPresent         := (eInOperationBaseGripper = E_InOperationBaseGripper
        .InOp) AND (eInOpGripper = E_InOpGripper.PartPresent);
82
```

7 POU: FB_OpenGripper

```
FUNCTION_BLOCK FB_OpenGripper

VAR_INPUT
    Execute : BOOL;
    thOpenMax : WORD := 50;
END_VAR

VAR_IN_OUT
    hwSensor : UA_Schunk_mms;
    hwEV : UA_Festo;
END_VAR

VAR_OUTPUT
    Done : BOOL;
    Active : BOOL;
    Error : BOOL;
END_VAR

VAR
    tnCheckDone : TON;
    rExecute : R_TRIG;
    eExecuteGripper : E_ExecuteGripper;
END_VAR
```

```
(*
      Manage Inputs
*)
rExecute(CLK := Execute);
// Timer.Q true if Execute and not threshold for more thant 1 sec.
tnCheckDone(IN := Execute AND NOT (hwSensor.Value < thOpenMax),
            PT := T#1S);

CASE eExecuteGripper OF
    E_ExecuteGripper.Idle :
        IF rExecute.Q THEN
            eExecuteGripper := E_ExecuteGripper.Init;
        END_IF

    E_ExecuteGripper.Init :
        // No init
        IF tnCheckDone.Q THEN
            eExecuteGripper := E_ExecuteGripper.Error;
        ELSE
            eExecuteGripper := E_ExecuteGripper.InOp;
        END_IF

    E_ExecuteGripper.InOp :
        IF tnCheckDone.Q THEN
            eExecuteGripper := E_ExecuteGripper.Error;
        ELSIF (hwSensor.Value < thOpenMax) THEN
```

```
27                  eExecuteGripper := E_ExecuteGripper.Done;
28              END_IF
29
30          E_ExecuteGripper.Done :
31              IF NOT Execute THEN
32                  eExecuteGripper := E_ExecuteGripper.Idle;
33              END_IF
34
35          E_ExecuteGripper.Error :
36              IF NOT Execute THEN
37                  eExecuteGripper := E_ExecuteGripper.Idle;
38              END_IF
39      END_CASE
40
41      IF eExecuteGripper = E_ExecuteGripper.InOp THEN
42          hwEV.SetOut := TRUE;
43      END_IF
44
45      Done := (eExecuteGripper = E_ExecuteGripper.Done);
46      Active := (eExecuteGripper = E_ExecuteGripper.Init) OR (eExecuteGripper
         = E_ExecuteGripper.InOp) ;
47      Error := (eExecuteGripper = E_ExecuteGripper.Error);
48
```

8 POU: PRG_Student

```
1       (*
2           Practical Work 01 / Write your first PLC Code here
3
4           www.hevs.ch
5           Institut Systemes Industriels
6           Project:        HEVS Pack 2022
7           Author:             Cedric Lenoir / Christophe Truffer
8           Date:               2025 January 29
9       *)
10      PROGRAM PRG_Student
11      VAR
12          uliLoop     : ULINT;
13
14          fbGripperState : FB_GripperState;
15          fbOpenGripper : FB_OpenGripper;
16          fbCloseGripper : FB_CloseGripper;
17
18          stTestFbGripperHmi : ST_TestFbGripperHmi;
19
20          GripperIsOpen : BOOL := TRUE;
21          GripperIsClosed : BOOL := TRUE;
22
23
24      END_VAR
25
```

```
1       uliLoop := uliLoop + 1;
2
```

```
3       // For TP4 : PRG_DeviceManager has been disabled in Task Configuration
4
5
6       // Request to open / close gripper
7       fbOpenGripper . Execute  :=  stTestFbGripperHmi . openGripper ;
8       fbCloseGripper . Execute  :=  stTestFbGripperHmi . closeGripper ;
9
10      // Gripper states
11      stTestFbGripperHmi . gripperStateClosed  :=  fbGripperState . IsClosed ;
12      stTestFbGripperHmi . gripperStateOpen  :=  fbGripperState . IsOpen ;
13      stTestFbGripperHmi . gripperStatePartPresent  :=  fbGripperState . PartPresent ;
14      stTestFbGripperHmi . gripperStateError  :=  fbGripperState . Error ;
15      stTestFbGripperHmi . gripperStateInOp  :=  fbGripperState . InOperation ;
16
17      GripperIsOpen    :=  fbOpenGripper . Done ;
18      GripperIsClosed :=  fbCloseGripper . Done ;
19
20
21
22      // Calling FBs
23      fbOpenGripper ( hwEV  :=  GVL_Abox . uaAboxInterface . uaSchunkGripper ,
24                      hwSensor  :=  GVL_Abox . uaAboxInterface . uaSchunk ,
25                      Done  =>  stTestFbGripperHmi . executeOpenDone ) ;
26
27      fbCloseGripper ( hwEV  :=  GVL_Abox . uaAboxInterface . uaSchunkGripper ,
28                      hwSensor  :=  GVL_Abox . uaAboxInterface . uaSchunk ,
29                      Done  =>  stTestFbGripperHmi . executeCloseDone ) ;
30
31      fbGripperState ( Enable  :=  TRUE ,
32                           hw  :=  GVL_Abox . uaAboxInterface . uaSchunk ) ;
33
34
35
36
37
```