

Program blocks

Main [OB1]

Main Properties

General

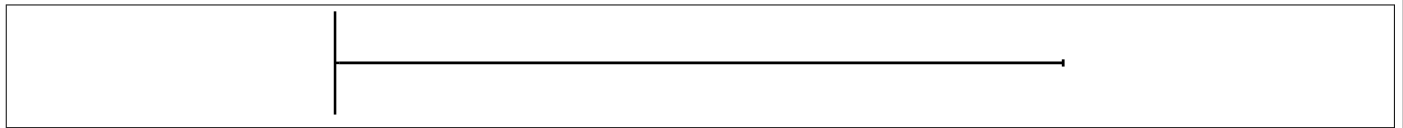
Name	Main	Number	1	Type	OB
Language	LAD	Numbering	Automatic		

Information

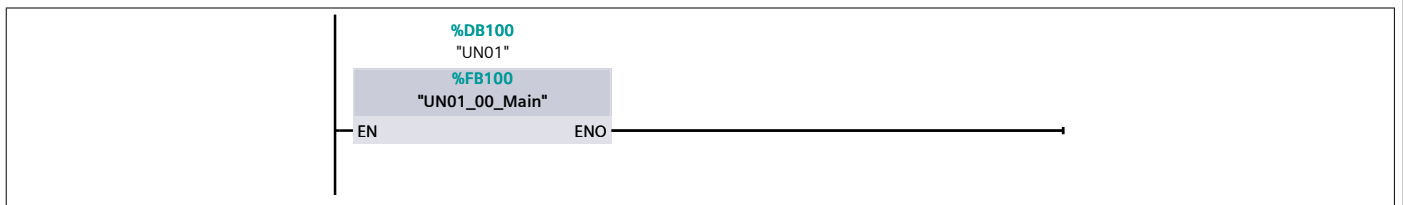
Title	"Main Program Sweep (Cycle)"	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
Remanence	Bool	
Temp		
Constant		

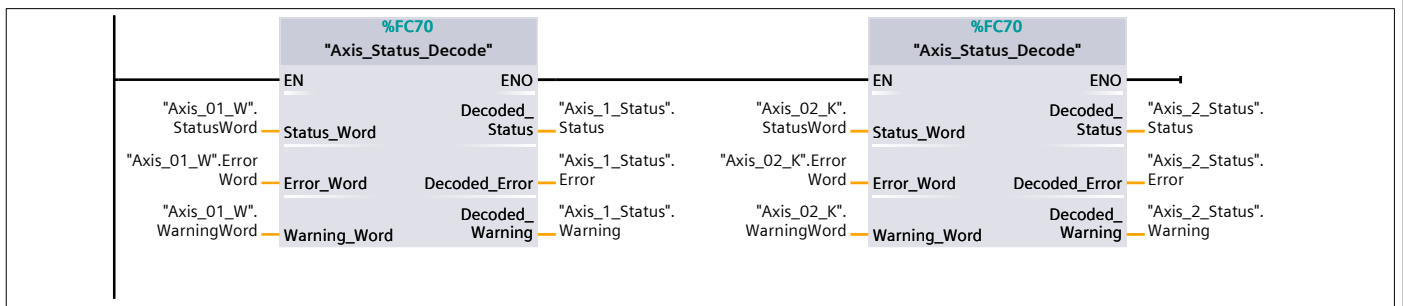
Network 1:



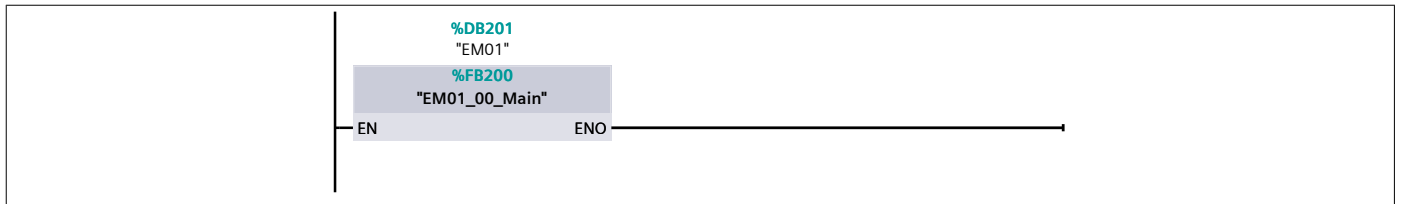
Network 2: Unit Processing



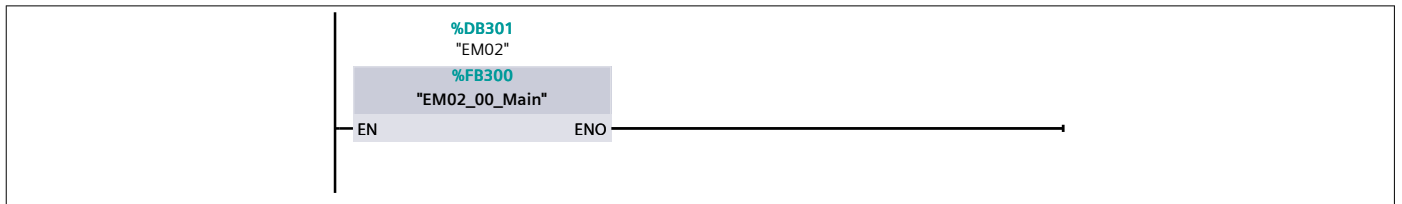
Network 3:



Network 4: EM 01 Processing



Network 5:



Program blocks

MC-Servo [OB91]

MC-Servo Properties

General

Name	MC-Servo	Number	91	Type	OB
Language	LAD	Numbering	Automatic		

Information

Title		Author		Comment	
Family		Version	1.0	User-defined ID	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
PIP_Input	Bool	
PIP_Output	Bool	
IO_System	USInt	
Event_Count	Int	
Synchronous	Bool	

Program blocks

MC-Interpolator [OB92]

MC-Interpolator Properties

General

Name	MC-Interpolator	Number	92	Type	OB
Language	LAD	Numbering	Automatic		

Information

Title		Author		Comment	
Family		Version	1.0	User-defined ID	

Name	Data type	Default value
▼ Input		
Initial_Call	Bool	
PIP_Input	Bool	
PIP_Output	Bool	
IO_System	USInt	
Event_Count	Int	
Reduction	UInt	

Program blocks

Startup [OB100]

Startup Properties

General

Name	Startup	Number	100	Type	OB
Language	LAD	Numbering	Automatic		

Information

Title	"Complete Restart"	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value
▼ Input		
LostRetentive	Bool	
LostRTC	Bool	
Temp		
Constant		

Program blocks / System blocks / Program resources

MC_ABORTMEASURINGINPUT [FB1317]

MC_ABORTMEASURINGINPUT Properties

General

Name	MC_ABORTMEASURINGINPUT	Number	1317	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
MeasuringInput	TO_MeasuringInput		
Execute	Bool	false	Non-retain
▼ Output			
Done	Bool	false	Non-retain
Busy	Bool	false	Non-retain
CommandAborted	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
InOut			
Static			

Program blocks / System blocks / Program resources

MC_MOVEJOG [FB1203]

MC_MOVEJOG Properties

General

Name	MC_MOVEJOG	Number	1203	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
Axis	TO_SpeedAxis		
JogForward	Bool	false	Non-retain
JogBackward	Bool	false	Non-retain
Velocity	LReal	100.0	Non-retain
Acceleration	LReal	-1.0	Non-retain
Deceleration	LReal	-1.0	Non-retain
Jerk	LReal	-1.0	Non-retain
PositionControlled	Bool	true	Non-retain
▼ Output			
InVelocity	Bool	false	Non-retain
Busy	Bool	false	Non-retain
CommandAborted	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
InOut			
Static			

Program blocks / System blocks / Program resources

MC_RESET [FB1207]

MC_RESET Properties

General

Name	MC_RESET	Number	1207	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
Axis	TO_Object		
Execute	Bool	false	Non-retain
Restart	Bool	false	Non-retain
▼ Output			
Done	Bool	false	Non-retain
Busy	Bool	false	Non-retain
CommandAborted	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
InOut			
Static			

Program blocks / System blocks / Program resources

MC_POWER [FB1206]

MC_POWER Properties

General

Name	MC_POWER	Number	1206	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
Axis	TO_Axis		
Enable	Bool	false	Non-retain
StartMode	DInt	1	Non-retain
StopMode	Int	0	Non-retain
▼ Output			
Status	Bool	false	Non-retain
Busy	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
InOut			
Static			

Program blocks / System blocks / Program resources

MC_HALT [FB1200]

MC_HALT Properties

General

Name	MC_HALT	Number	1200	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
Axis	TO_SpeedAxis		
Execute	Bool	false	Non-retain
Deceleration	LReal	-1.0	Non-retain
Jerk	LReal	-1.0	Non-retain
AbortAcceleration	Bool	false	Non-retain
▼ Output			
Done	Bool	false	Non-retain
Busy	Bool	false	Non-retain
CommandAborted	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
InOut			
Static			

Program blocks / System blocks / Program resources

MC_INTERPOLATECAM [FB1315]

MC_INTERPOLATECAM Properties

General

Name	MC_INTERPOLATECAM	Number	1315	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
Cam	TO_CamBase		
Execute	Bool	false	Non-retain
▼ Output			
Done	Bool	false	Non-retain
Busy	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
InOut			
Static			

Program blocks / System blocks / Program resources

MC_MEASURINGINPUT [FB1316]

MC_MEASURINGINPUT Properties

General

Name	MC_MEASURINGINPUT	Number	1316	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
MeasuringInput	TO_MeasuringInput		
Execute	Bool	false	Non-retain
Mode	DInt	0	Non-retain
MeasuringRange	Bool	false	Non-retain
StartPosition	LReal	0.0	Non-retain
EndPosition	LReal	0.0	Non-retain
▼ Output			
Done	Bool	false	Non-retain
Busy	Bool	false	Non-retain
CommandAborted	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
MeasuredValue1	LReal	0.0	Non-retain
MeasuredValue2	LReal	0.0	Non-retain
InOut			
Static			

Program blocks / System blocks / Program resources

MC_CAMIN [FB1309]

MC_CAMIN Properties

General

Name	MC_CAMIN	Number	1309	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
Master	TO_Axis		
Slave	TO_Synchronou- sAxis		
Cam	TO_CamBase		
Execute	Bool	false	Non-retain
MasterOffset	LReal	0.0	Non-retain
SlaveOffset	LReal	0.0	Non-retain
MasterScaling	LReal	1.0	Non-retain
SlaveScaling	LReal	1.0	Non-retain
MasterSyncPosition	LReal	0.0	Non-retain
SyncProfileReference	DInt	1	Non-retain
MasterStartDistance	LReal	0.0	Non-retain
Velocity	LReal	-1.0	Non-retain
Acceleration	LReal	-1.0	Non-retain
Deceleration	LReal	-1.0	Non-retain
Jerk	LReal	-1.0	Non-retain
ApplicationMode	DInt	0	Non-retain
SyncDirection	DInt	3	Non-retain
▼ Output			
StartSync	Bool	false	Non-retain
InSync	Bool	false	Non-retain
Busy	Bool	false	Non-retain
CommandAborted	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
EndOfProfile	Bool	false	Non-retain
InOut			
Static			

Program blocks / System blocks / Program resources

MC_MOVEVELOCITY [FB1205]

MC_MOVEVELOCITY Properties

General

Name	MC_MOVEVELOCITY	Number	1205	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
Axis	TO_SpeedAxis		
Execute	Bool	false	Non-retain
Velocity	LReal	100.0	Non-retain
Acceleration	LReal	-1.0	Non-retain
Deceleration	LReal	-1.0	Non-retain
Jerk	LReal	-1.0	Non-retain
Direction	Int	0	Non-retain
Current	Bool	false	Non-retain
PositionControlled	Bool	true	Non-retain
▼ Output			
InVelocity	Bool	false	Non-retain
Busy	Bool	false	Non-retain
CommandAborted	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
InOut			
Static			

Program blocks / System blocks / Program resources

MC_HOME [FB1201]

MC_HOME Properties

General

Name	MC_HOME	Number	1201	Type	FB
Language	Motion_DB	Numbering	Automatic		

Information

Title		Author	SIMATIC	Comment	
Family	MC_1500	Version	8.0	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
Axis	TO_Axis		
Execute	Bool	false	Non-retain
Position	LReal	0.0	Non-retain
Mode	Int	0	Non-retain
Sensor	DInt	0	Non-retain
▼ Output			
ReferenceMarkPosition	LReal	0.0	Non-retain
Done	Bool	false	Non-retain
Busy	Bool	false	Non-retain
CommandAborted	Bool	false	Non-retain
Error	Bool	false	Non-retain
ErrorId	Word	16#0	Non-retain
InOut			
Static			

Program blocks / 0_UN01_ExampleMachine

UN01_00_Main [FB100]

UN01_00_Main Properties

General

Name	UN01_00_Main	Number	100	Type	FB
Language	LAD	Numbering	Manual		

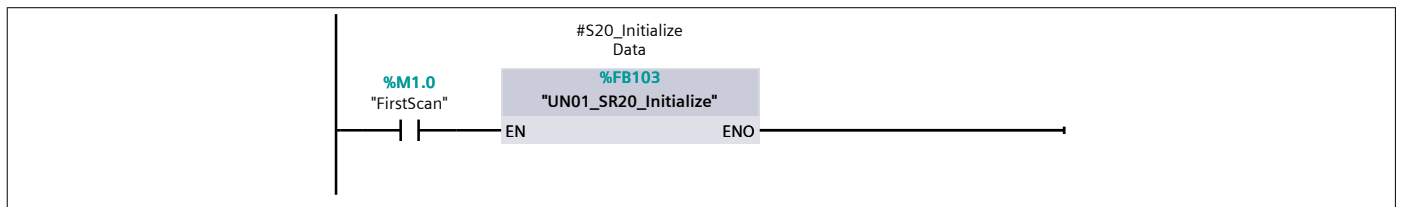
Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

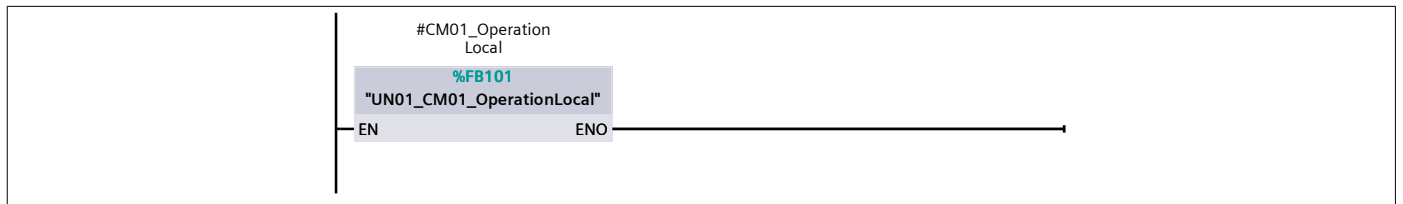
Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
HMI_O_ModeCurrentName	String[18]	"	Non-retain
HMI_O_StateCurrentName	String[16]	"	Non-retain
HMI_O_MachineSafetyFault	DInt	0	Non-retain
HMI_O_MajorFaultMessage	DInt	0	Non-retain
HMI_O_MinorFaultMessage	DInt	0	Non-retain
HMI_O_Message	String[18]	"	Non-retain
Sts_GoodProducts	DInt	0	Non-retain
Sts_MSG_MachineFault	String	'Machine Fault'	Non-retain
Sts_MSG_MachineHealthy	String	'Machine Healthy'	Non-retain
MachineFault_Major	Bool	false	Non-retain
MachineFault_Minor	Bool	false	Non-retain
MachineFaulted	Bool	false	Non-retain
UP_Exec_Step	DInt	0	Non-retain
UP_Exec_StepDn	Array[0..32] of DInt		Non-retain
UP_Exec_LastStep	Bool	false	Non-retain
UP_Exec_NoStepsActive	Bool	false	Non-retain
Inp_BatchCountSetpoint	DInt	0	Non-retain
Wrk_StartRequest	Bool	false	Non-retain
Wrk_StopRequest	Bool	false	Non-retain
Wrk_ResetRequest	Bool	false	Non-retain
Wrk_AbortRequest	Bool	false	Non-retain
Wrk_ClearRequest	Bool	false	Non-retain
Wrk_CompleteRequest	Bool	false	Non-retain
Wrk_HoldRequest	Bool	false	Non-retain
Wrk_UnHoldRequest	Bool	false	Non-retain
Wrk_SuspendRequest	Bool	false	Non-retain
Wrk_Compare_All_EM_Enabled	DInt	-1	Non-retain
Wrk_AllSafetyOK	Bool	false	Non-retain
Wrk_EstopOK	Bool	false	Non-retain
Wrk_GuardsOK	Bool	false	Non-retain
Wrk_LightCurtainsOK	Bool	false	Non-retain

Name	Data type	Default value	Retain
Wrk_NetworkCommunicationOK	Bool	false	Non-retain
Wrk_StartWarningExpired	Bool	false	Non-retain
S20_InitializeData	"UN01_SR20_Initialize"		
CM01_OperationLocal	"UN01_CM01_OperationLocal"		
CM03_FaultHandler	"UN01_CM03_FaultHandler"		
UP02_StateComplete	"UN01_UP02_StateComplete"		
UP01_PackML	"UN01_UP01_PackML"		
UP00_Procedure	"UN01_UP00_Procedure"		
Temp			
Constant			

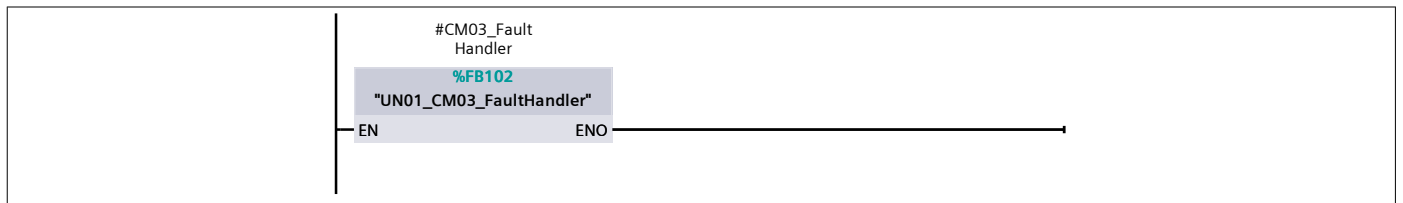
Network 1:



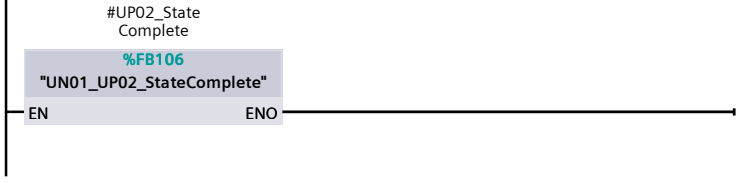
Network 2:



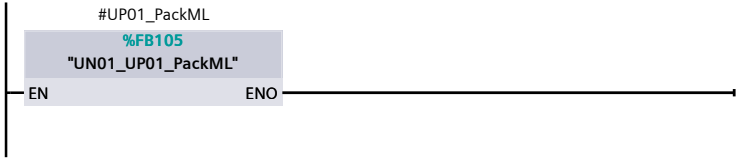
Network 3:



Network 4:



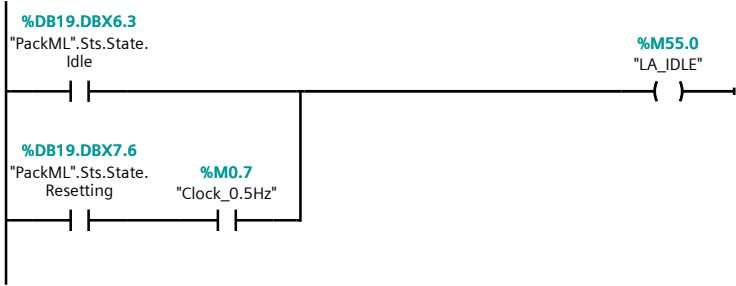
Network 5:



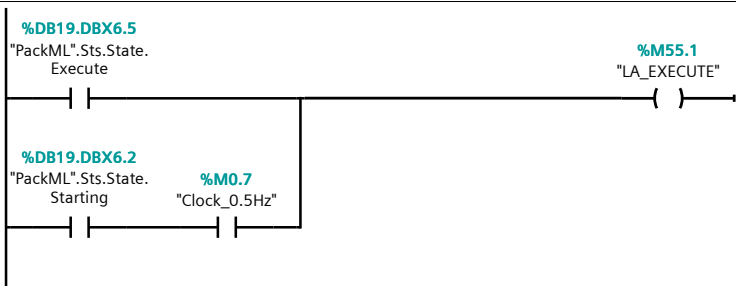
Network 6:



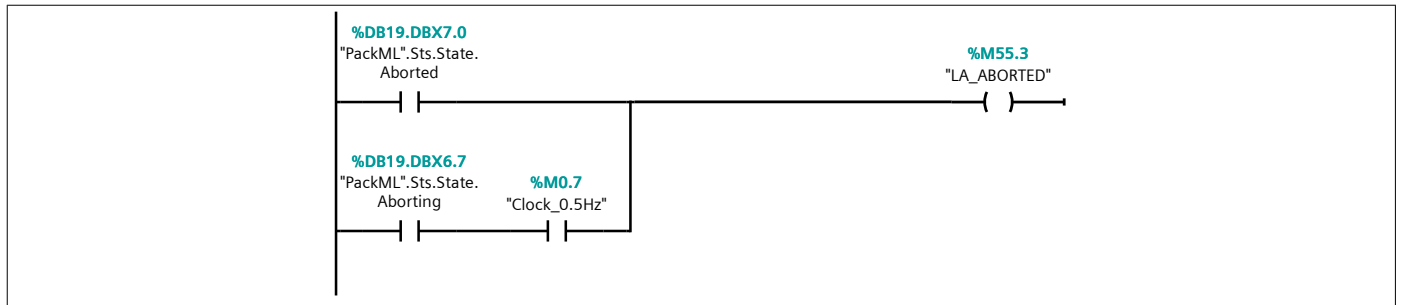
Network 7: INDICATOR LAMPS for C-More



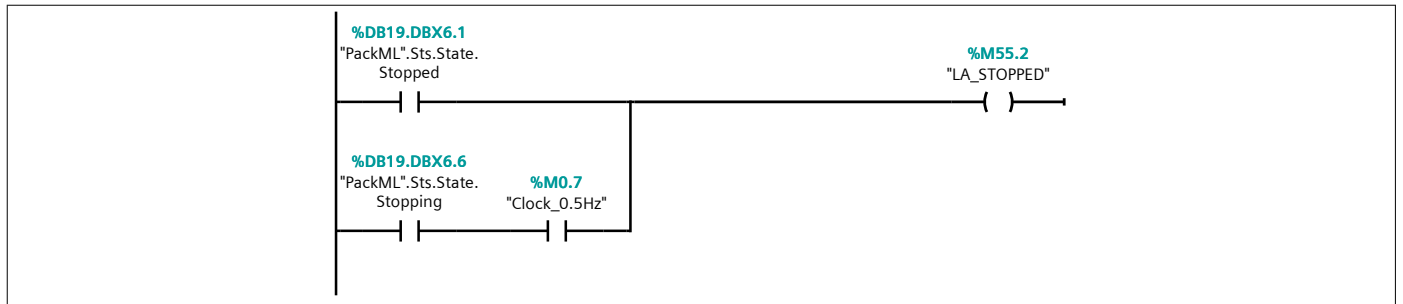
Network 8:



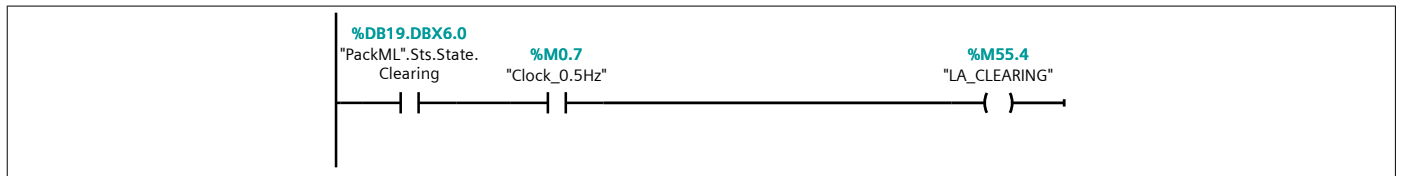
Network 9:



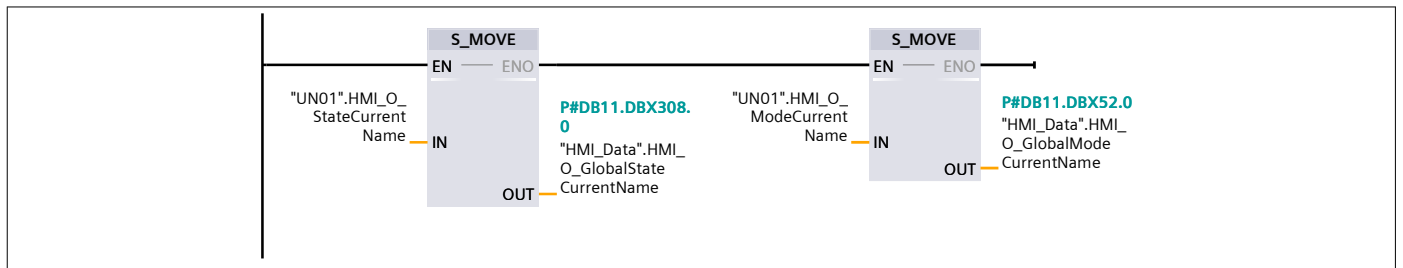
Network 10:



Network 11:



Network 12: Pass on state and mode names to global hmi



Program blocks / 0_UN01_ExampleMachine

UN01_CM01_OperationLocal [FB101]

UN01_CM01_OperationLocal Properties

General

Name	UN01_CM01_Operation-Local	Number	101	Type	FB
Language	LAD	Numbering	Manual		

Information

Title		Author		Comment	Handle commands that are requests to change to a particular state
Family		Version	0.1	User-defined ID	

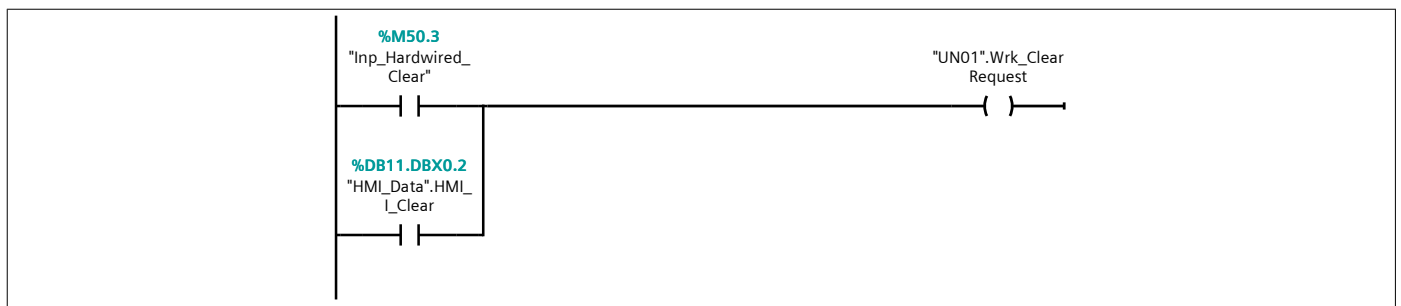
Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Wrk_StartWarning_TON	TON_TIME		Non-retain
Temp			
Constant			

Network 1: Clear faults conditions

CLEAR FAULTS CONDITIONS

The Unit Conditions Set Here Are Used to Clear Faults and Initiate a State Transition of the Current Mode Operation Procedure from the Aborted State:

- 1) To the Clearing State, If the Clearing State is Enabled
- 2) To the Stopped State, If the Clearing State is Disabled

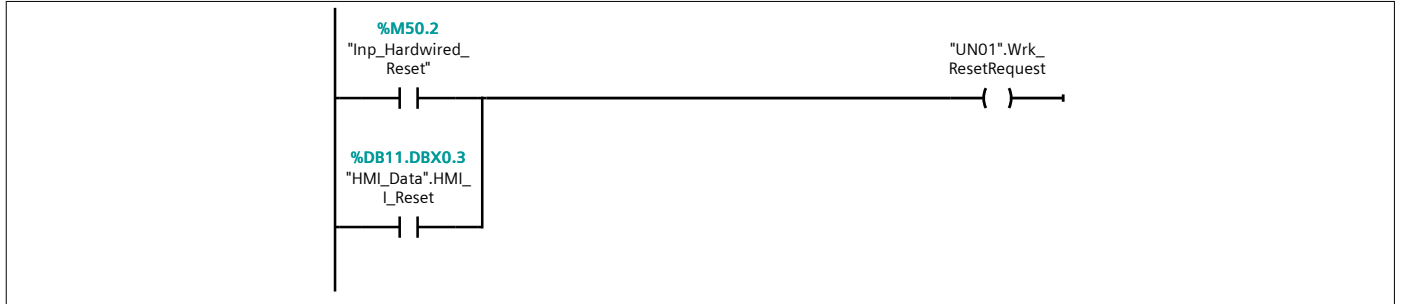


Network 2: Reset Conditions

RESET CONDITIONS

The Unit Condition Set Here Is Used to Initiate the Start Warning Cycle That Results in a State Transition of the Current Mode Operation Procedure from the Stopped State:

- 1) To the Resetting State, If the Resetting State is Enabled
- 2) To the Idle State, If the Resetting State is Disabled

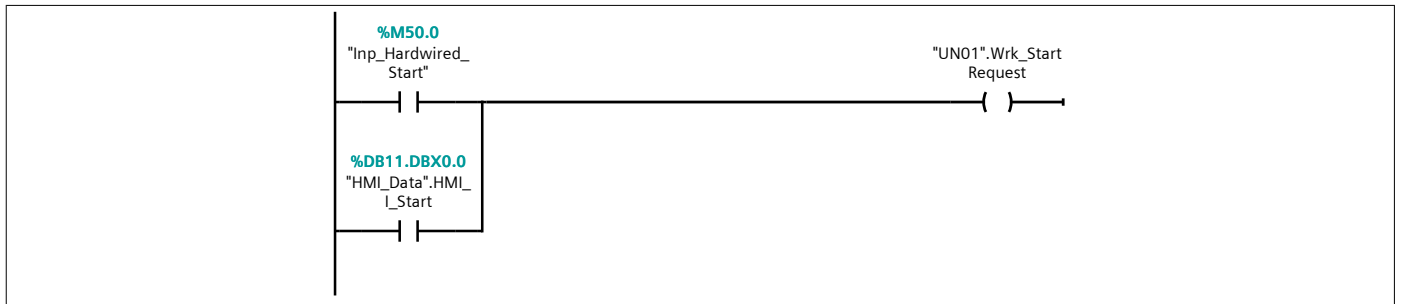


Network 3: Start conditions

START CONDITIONS

The Unit Condition Set Here is Used to

- 1) If the Idle State is Enabled, Initiate a State Transition of the Current Mode Operation Procedure:
 - a) From the Idle State to the Starting State, If the Starting State Is Enabled
 - b) From the Idle State to the Execute State, If the Starting State Is Disabled
- 2) If the Idle State is Disabled, Initiate the Start Warning Cycle That Results in a State Transition of the Current Mode Operation Procedure from the Stopped State:
 - a) To the Resetting State, If the Resetting State Is Enabled
 - b) To the Starting State, If the Resetting State is Disabled and the Starting State Is Enabled
 - c) To the Execute State, If Bothe the Resetting and Starting States Are Disabled



Network 4: Stop Conditions

***** Changed polarity of Inp_Hardwired_Stop to match it being a N.C. contact

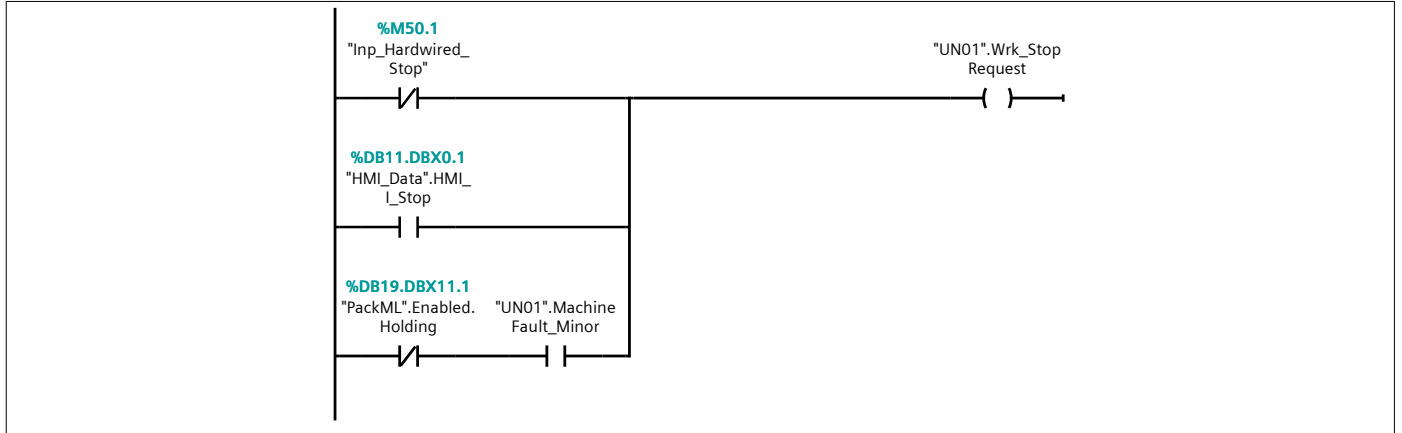
STOP CONDITIONS

The Unit Condition Set Here is Used to Initiate a State Transition of the Current Mode Operation Procedure to the:

- 1) Stopping State, If the Stopping State Is Enabled
- 2) Stopped State, If the Stopping State Is Disabled

From Any of the Following States:

Resetting, Idle, Starting, Execute, Holding, Held, UnHolding, Suspending, Suspended, UnSuspending, Completing



Network 5: Abort Conditions

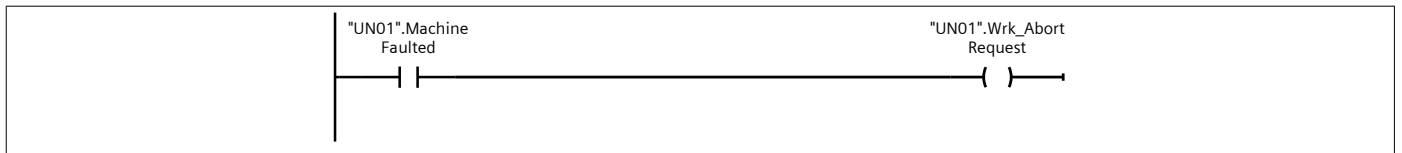
ABORT CONDITIONS

The Unit Condition Set Here is Used to Initiate a State Transition of the Current Mode Operation Procedure to the:

- 1) Aborting State, If the Aborting State Is Enabled
- 2) Aborted State, If the Aborting State Is Disabled

From Any of the Following States:

Resetting, Idle, Starting, Execute, Holding, Held, UnHolding, Suspending, Suspended, UnSuspending, Completing, Stopped, Stopping, Clearing

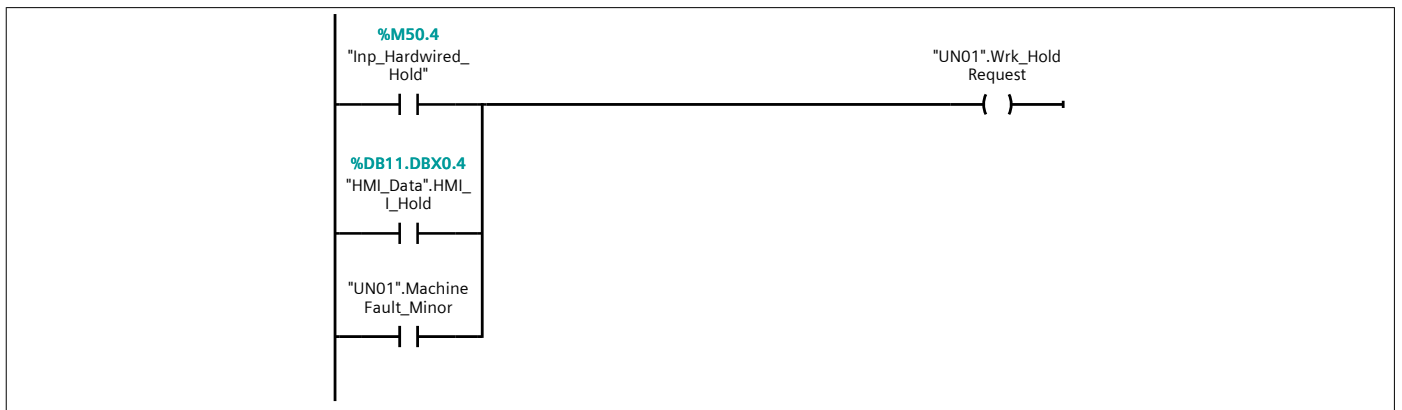


Network 6: Hold Conditions

HOLD CONDITIONS

The Unit Condition Set Here is Used to Initiate a State Transition of the Current Mode Operation Procedure from the Execute State to the:

- 1) Holding State, If the Holding State is Enabled
- 2) Held State, If the Holding State is Disabled

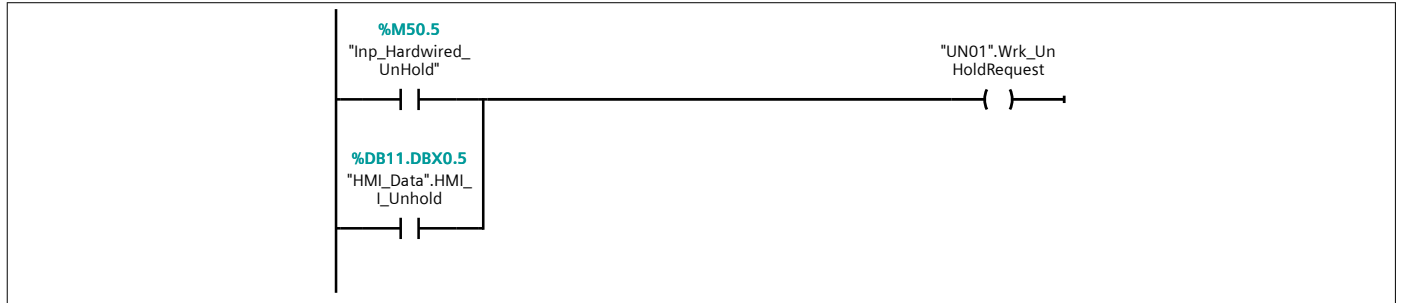


Network 7: Un-hold conditions

UN-HOLD CONDITIONS

The Unit Condition Set Here is Used to Initiate a State Transition of the Current Mode Operation Procedure from the Held State to the:

- 1) UnHolding State, If the UnHolding State is Enabled
- 2) Execute State, If the UnHolding State is Disabled



Network 8: Suspend Conditions

SUSPEND CONDITIONS

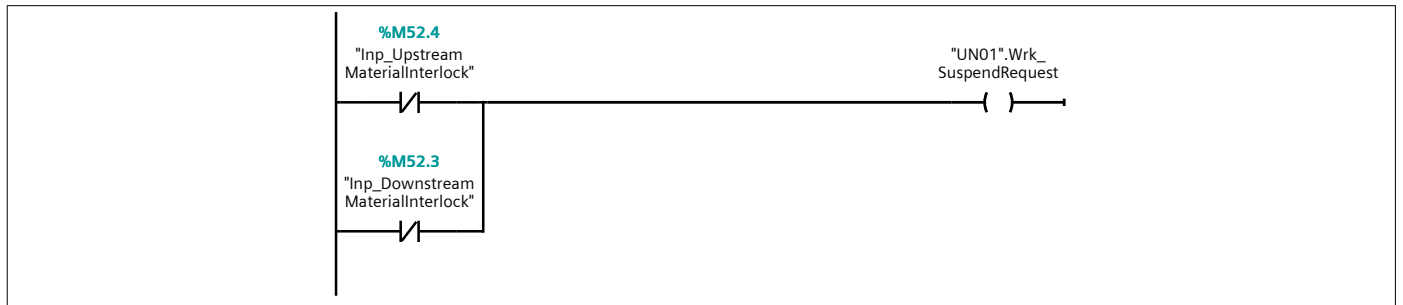
If any upstream or downstream material interlocks are not satisfied a suspend request will be made.

If a suspend request is made while the Machine is in the Execute State, Then a State Transition Is Initiated From the Execute State to the:

- 1) Suspending State, If the Suspending State Is Enabled
- 2) Suspended State, If the Suspending State Is Disabled

When ALL interlock conditions Are Satisfied, Then a State Transition Is Initiated From the Suspended State to the:

- 1) UnSuspending State, If the UnSuspending State Is Enabled
- 2) Execute State, If the UnSuspending State Is Disabled



Network 9: Complete Condition

COMPLETE CONDITION

The Unit Condition Set Here is Used to Indicate That the Unit Has Produced the Desired Quantity of Good Product. It is Used for Processes That Run Desired Batch or Job Quantities And Automatically Stop to Conserve Materials. Such Processes Take Advantage of the Completing And Complete States of the PackML Model.

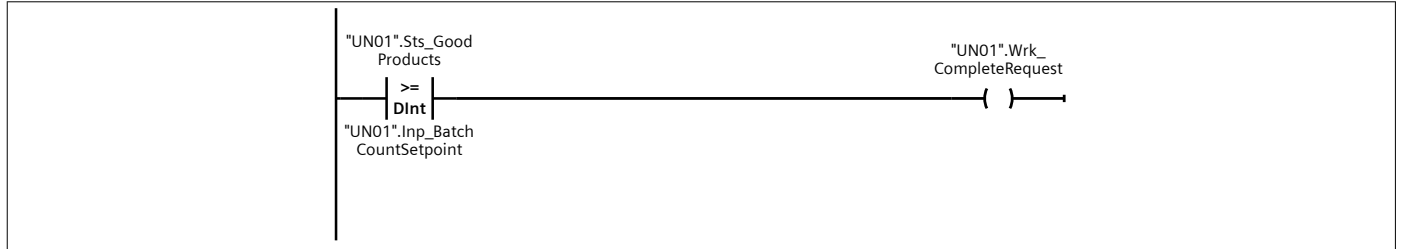
This Unit Condition is Generated By Comparing the Quantity of Good Product Counted or Calculated By the Program (In this Example, the Information from the Performance Tracking AOI is Used to Calculate Good Product *), to the Batch Count Setpoint for the Current Recipe.

The Condition Initiates a State Complete Transition from the Execute State to the:

- 1) Completing State, If the Completing State Is Enabled
- 2) Complete State, If the Completing State Is Disabled

* The Production Counting Data Used By the Performance Tracking FB Must Be Generated By the Program and Input Into the UN01 DB.

It May Be Preferred to Use the Production Counting Data Directly for Setting this Condition.

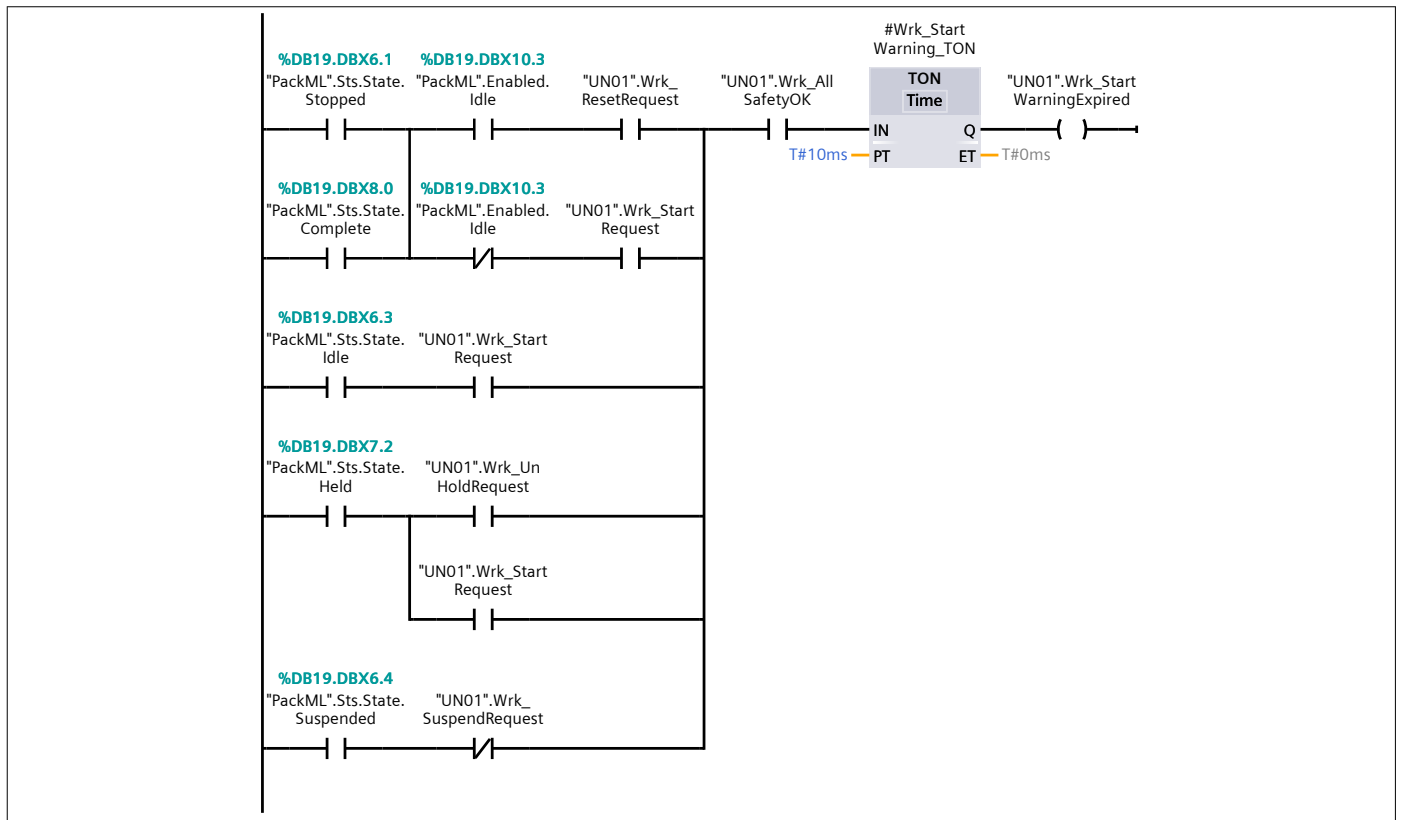


Network 10: Unit Start Logic

UNIT START LOGIC

This Unit Start Logic is Used to Provide a Warning Cycle Time Before Motion Occurs on the Machine. This Warning Cycle Should be Used Upon Initiating All State Transition Commands That Ultimately Result in Transition to the Execute State, And May be Used to Provide Audible And/Or Visual Notification to the Operator That Motion Will Occur Upon Completion.

In case of Start warning buzzer included in the logic increase the timer preset to 1 or 2 sec



Program blocks / 0_UN01_ExampleMachine

UN01_CM03_FaultHandler [FB102]

UN01_CM03_FaultHandler Properties

General

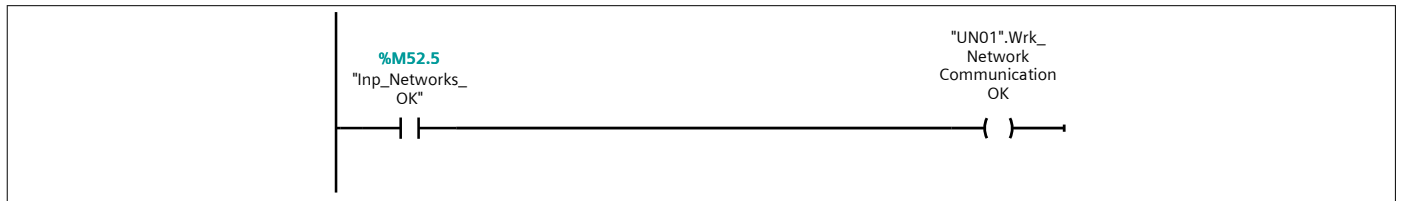
Name	UN01_CM03_FaultHandler	Number	102	Type	FB
Language	LAD	Numbering	Manual		

Information

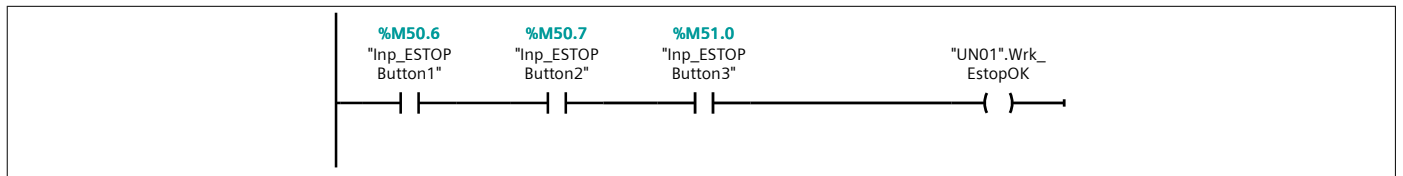
Title	Handle unit faults	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Temp_Dint	DInt	0	Non-retain
Temp_Bool	Bool	false	Non-retain
Temp			
Constant			

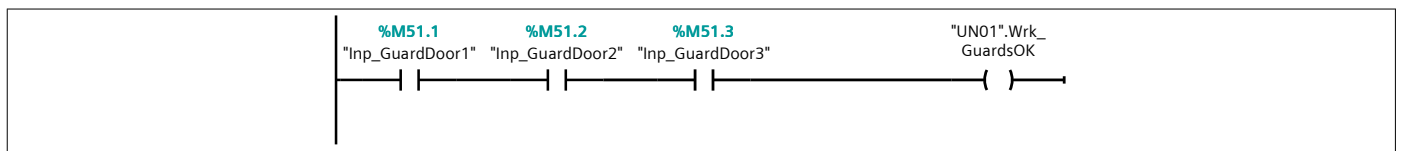
Network 1: Communication faults



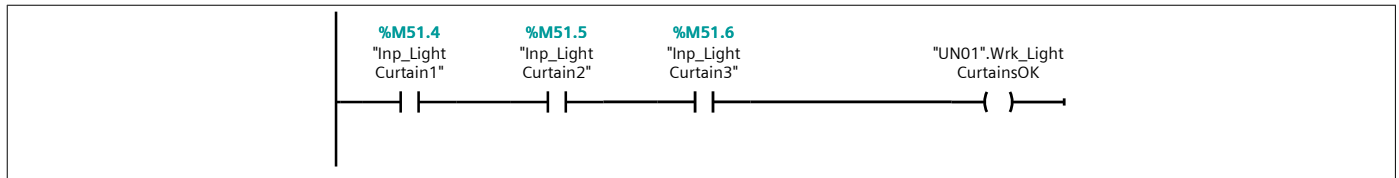
Network 2: E-Stop Collection



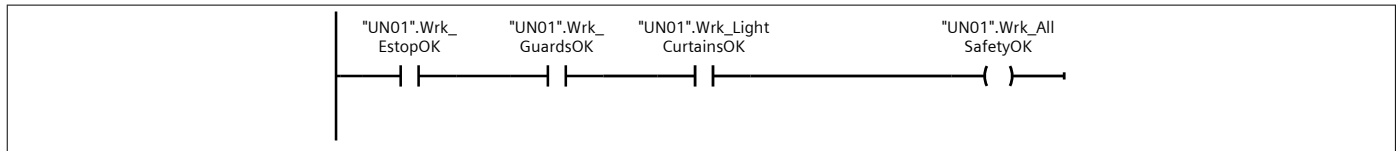
Network 3: Guard door collection



Network 4: Light curtain collection

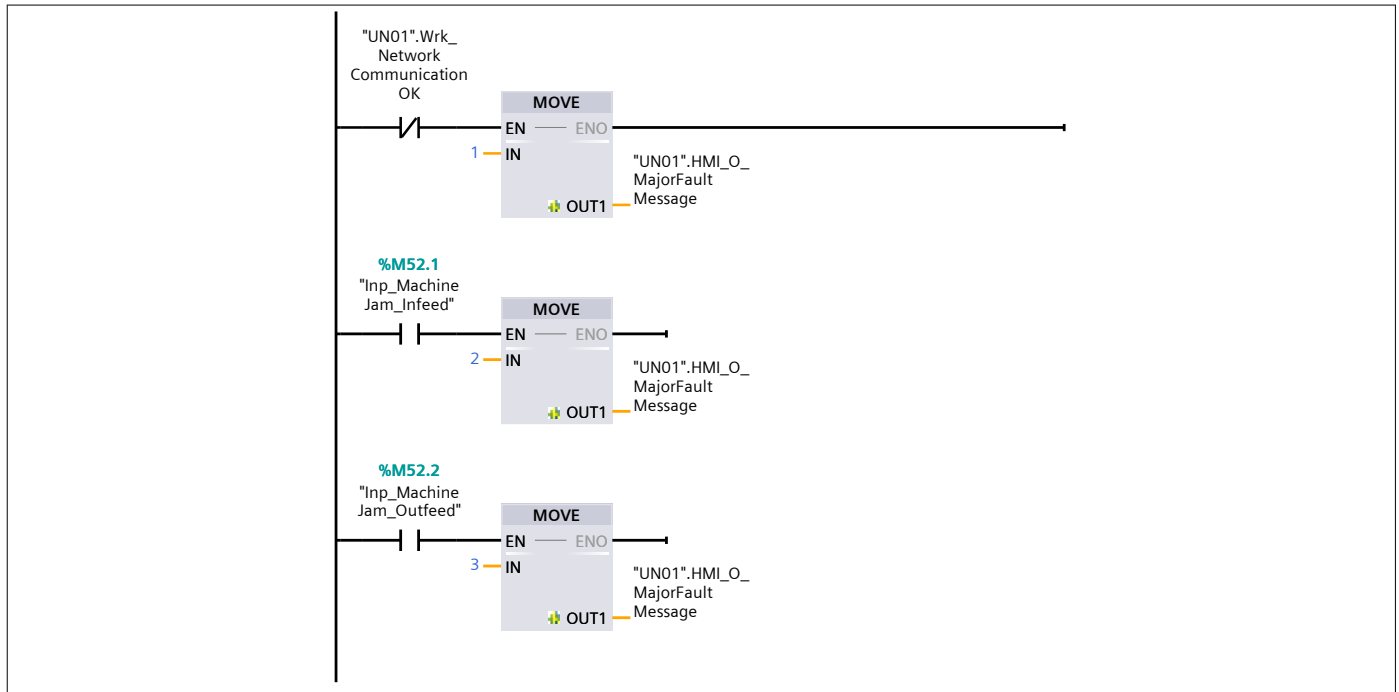


Network 5: All safety inputs OK



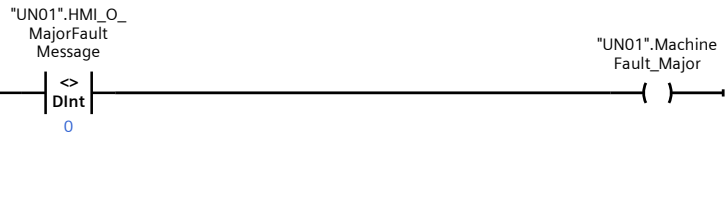
Network 6: Machine major faults

Add major machine faults that should result in an abort request to the machine

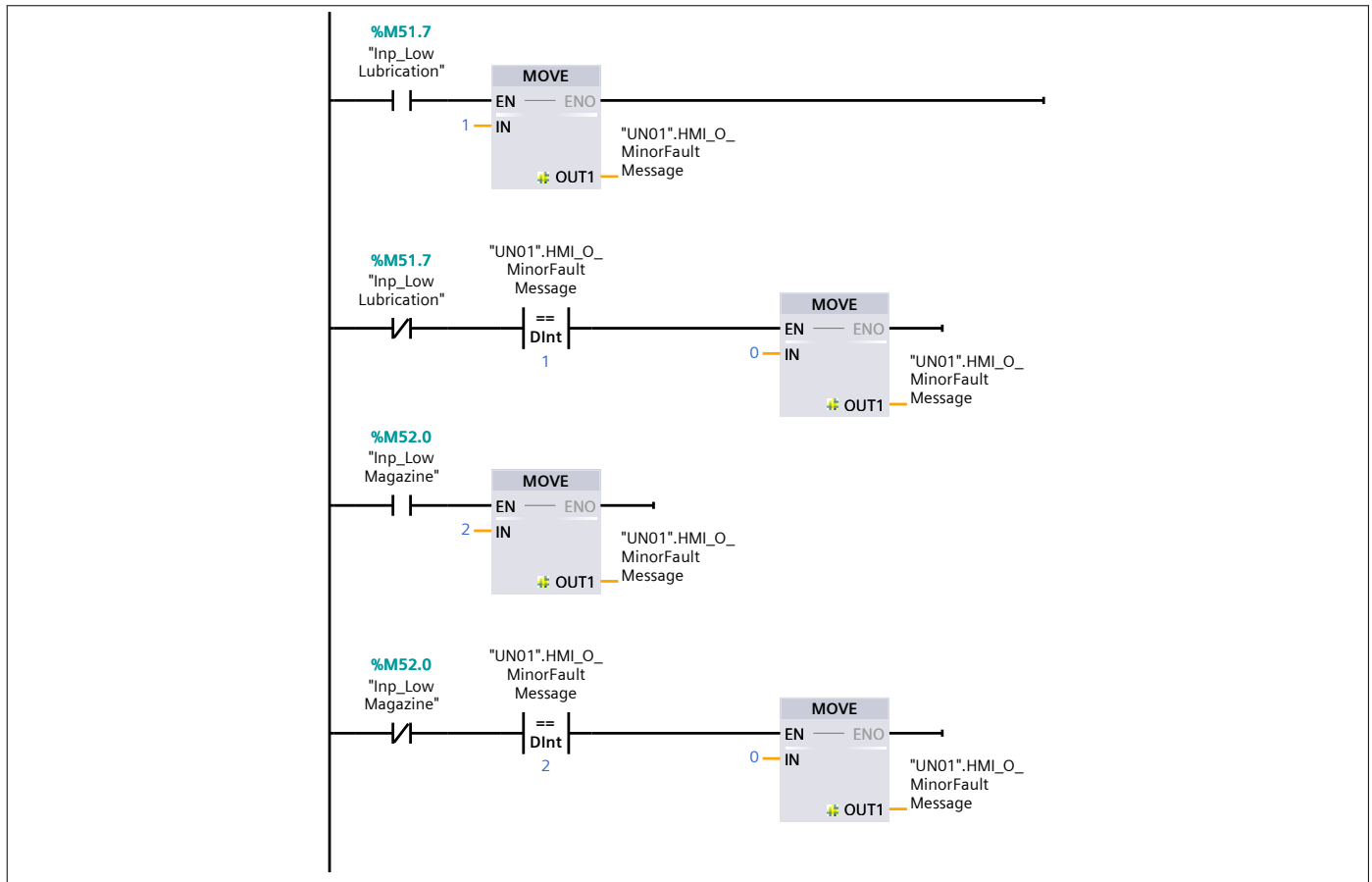


Network 7: General Unit Major Fault - Will result in machine abort request.

Add machine-specific major fault conditions as rung in conditions.

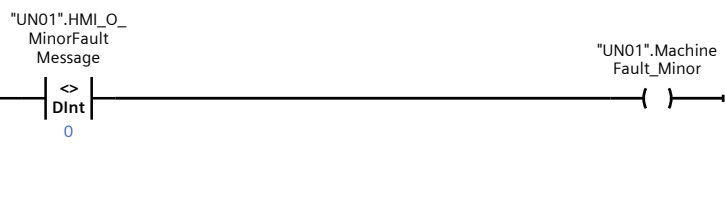


Network 8: Machine Minor faults

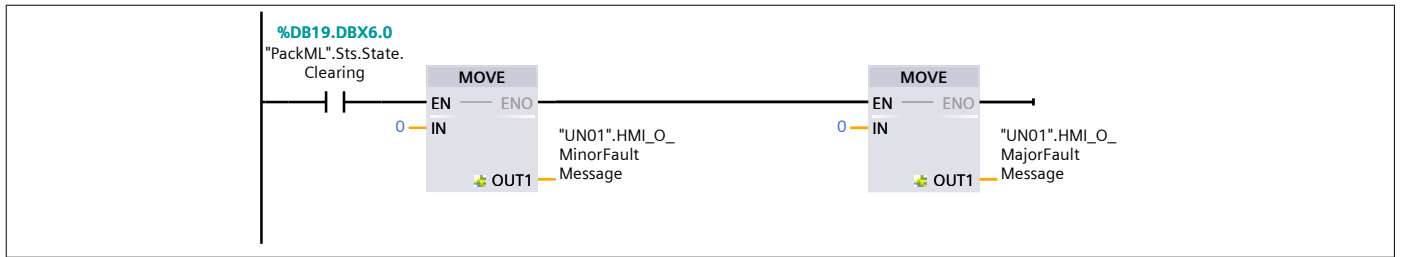


Network 9: General unit minor fault

Will result in machine hold request if holding state is enabled or stop if holding state is not enabled.

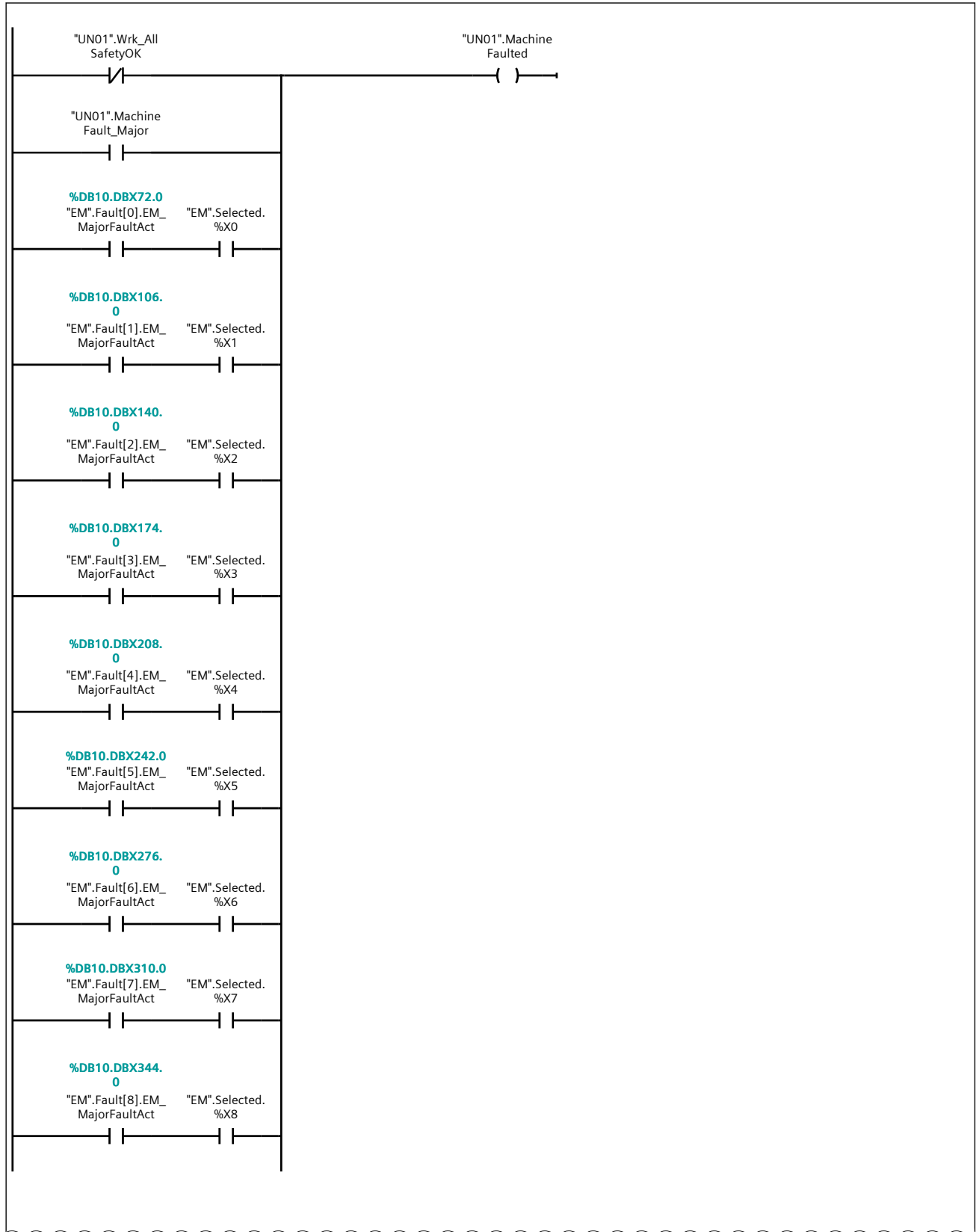


Network 10: Clear machine fault indicators



Network 11: General unit fault summary

Network 11: General unit fault summary (1.1 / 4.1)



Network 11: General unit fault summary (2.1 / 4.1)

1.1 (Page17 - 5)

%DB10.DBX378.0 "EM".Fault[9].EM_MajorFaultAct	"EM".Selected. %X9
%DB10.DBX412.0 "EM".Fault[10].EM_MajorFaultAct	"EM".Selected. %X10
%DB10.DBX446.0 "EM".Fault[11].EM_MajorFaultAct	"EM".Selected. %X11
%DB10.DBX480.0 "EM".Fault[12].EM_MajorFaultAct	"EM".Selected. %X12
%DB10.DBX514.0 "EM".Fault[13].EM_MajorFaultAct	"EM".Selected. %X13
%DB10.DBX548.0 "EM".Fault[14].EM_MajorFaultAct	"EM".Selected. %X14
%DB10.DBX582.0 "EM".Fault[15].EM_MajorFaultAct	"EM".Selected. %X15
%DB10.DBX616.0 "EM".Fault[16].EM_MajorFaultAct	"EM".Selected. %X16
%DB10.DBX650.0 "EM".Fault[17].EM_MajorFaultAct	"EM".Selected. %X17
%DB10.DBX684.0 "EM".Fault[18].EM_MajorFaultAct	"EM".Selected. %X18
%DB10.DBX718.0 "EM".Fault[19].EM_MajorFaultAct	"EM".Selected. %X19

3.1 (Page17 - 7)

Network 11: General unit fault summary (3.1 / 4.1)

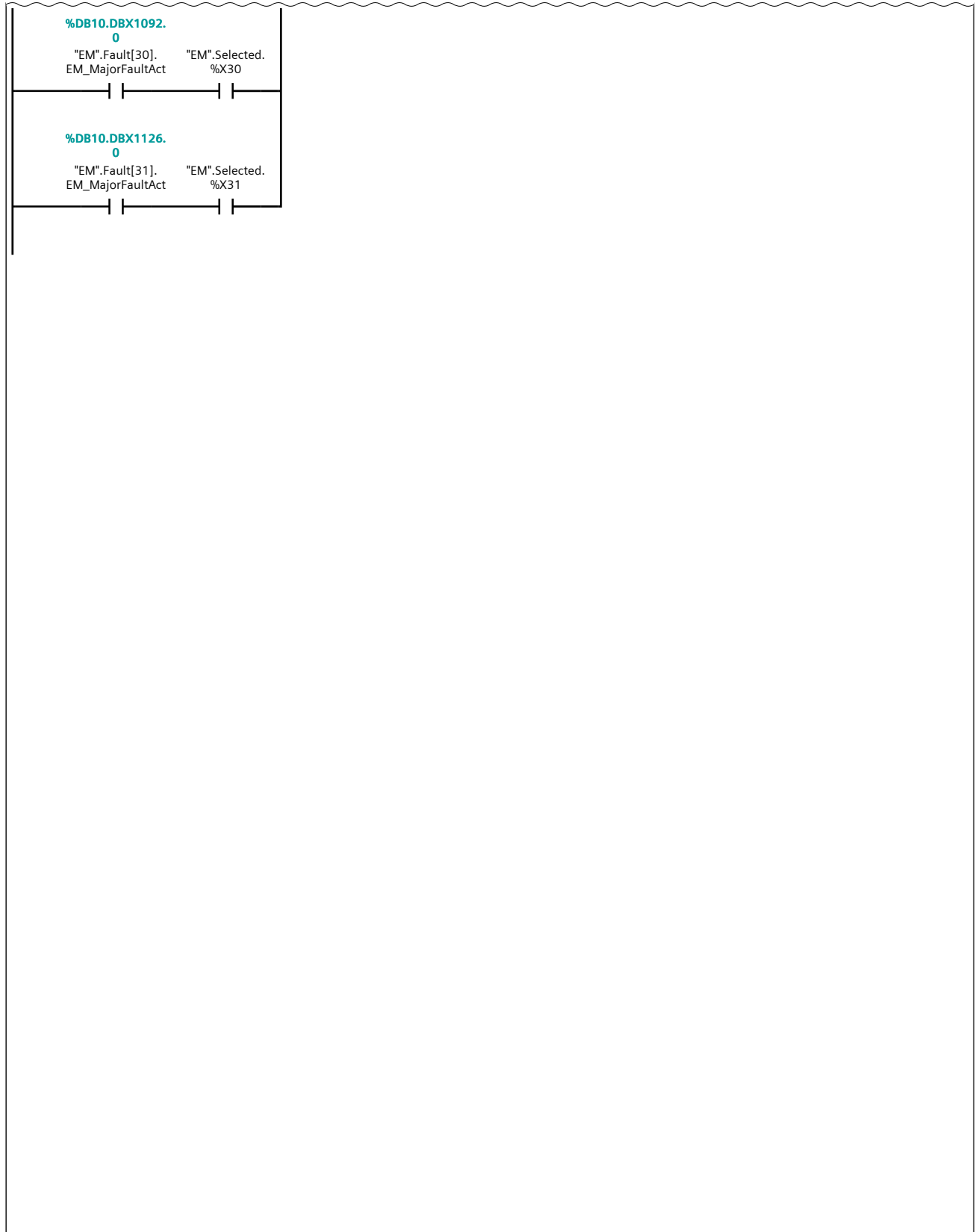
2.1 (Page17 - 6)

%DB10.DBX752.0 "EM".Fault[20]. EM_MajorFaultAct	"EM".Selected. %X20
%DB10.DBX786.0 "EM".Fault[21]. EM_MajorFaultAct	"EM".Selected. %X21
%DB10.DBX820.0 "EM".Fault[22]. EM_MajorFaultAct	"EM".Selected. %X22
%DB10.DBX854.0 "EM".Fault[23]. EM_MajorFaultAct	"EM".Selected. %X23
%DB10.DBX888.0 "EM".Fault[24]. EM_MajorFaultAct	"EM".Selected. %X24
%DB10.DBX922.0 "EM".Fault[25]. EM_MajorFaultAct	"EM".Selected. %X25
%DB10.DBX956.0 "EM".Fault[26]. EM_MajorFaultAct	"EM".Selected. %X26
%DB10.DBX990.0 "EM".Fault[27]. EM_MajorFaultAct	"EM".Selected. %X27
%DB10.DBX1024.0 "EM".Fault[28]. EM_MajorFaultAct	"EM".Selected. %X28
%DB10.DBX1058.0 "EM".Fault[29]. EM_MajorFaultAct	"EM".Selected. %X29

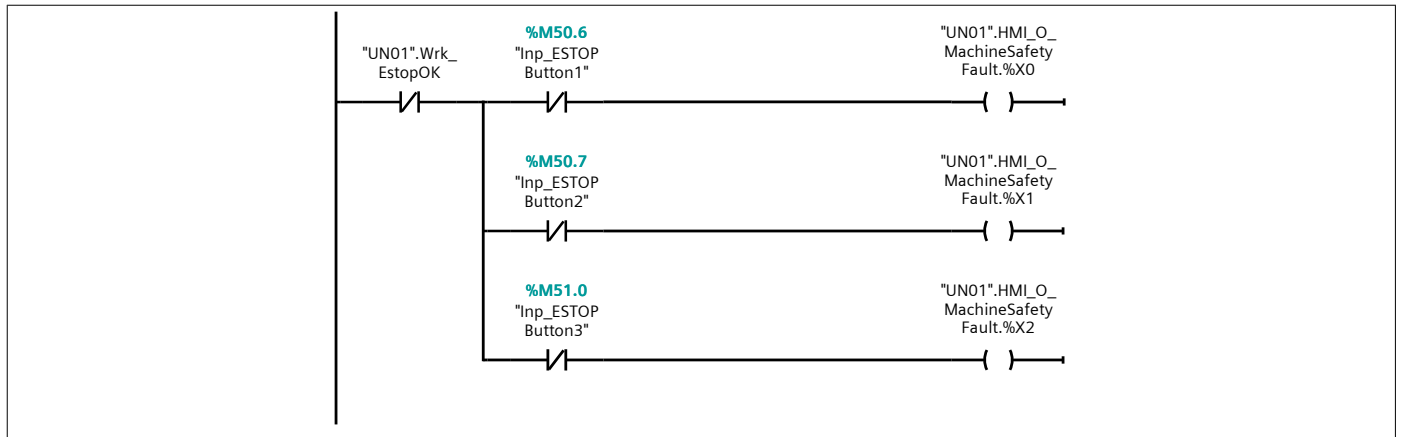
4.1 (Page17 - 8)

Network 11: General unit fault summary (4.1 / 4.1)

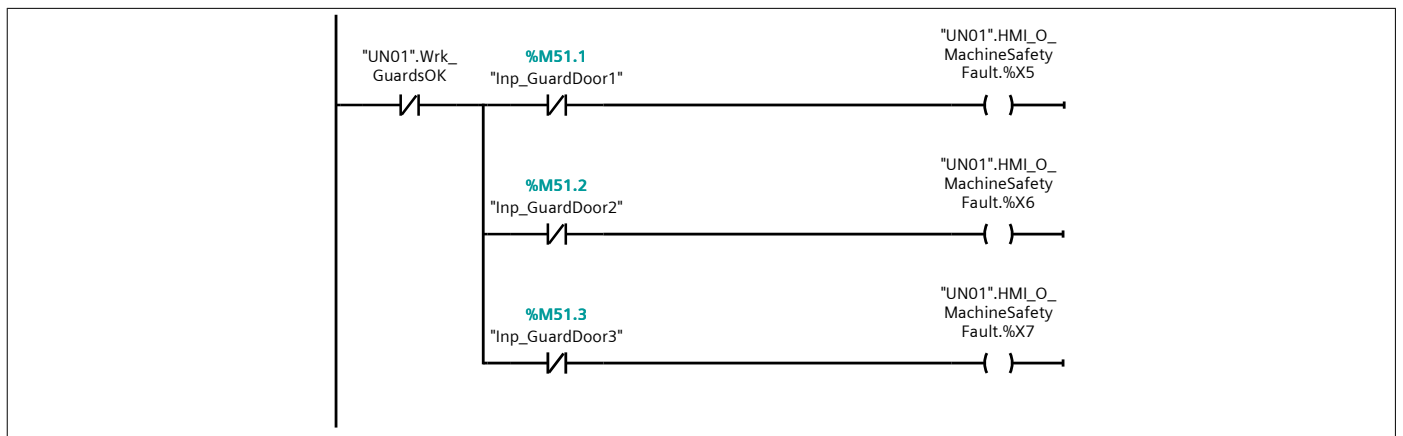
3.1 (Page17 - 7)



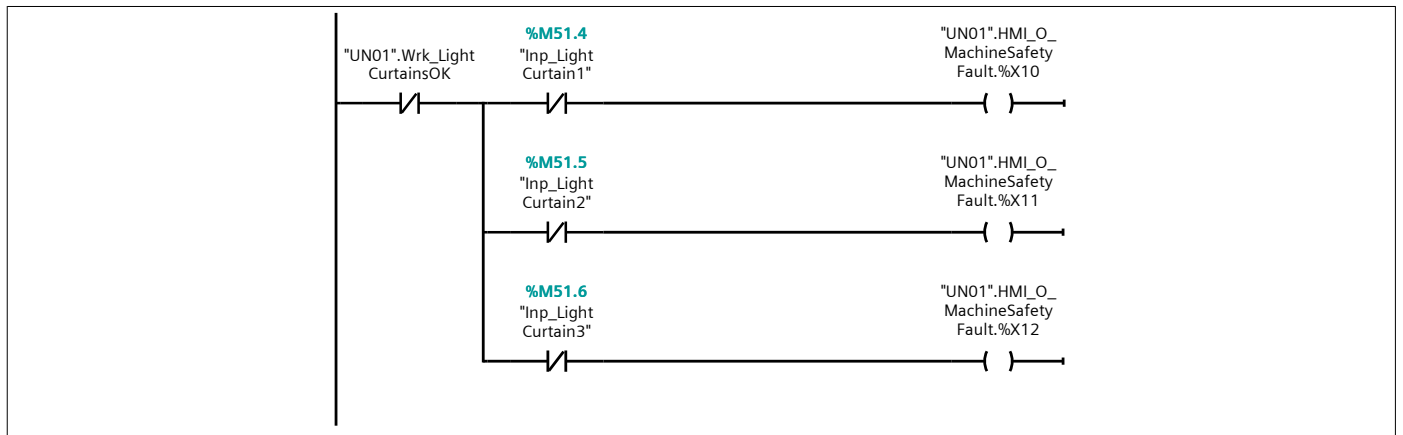
Network 12: Safety estops/guards/curtains status reporting



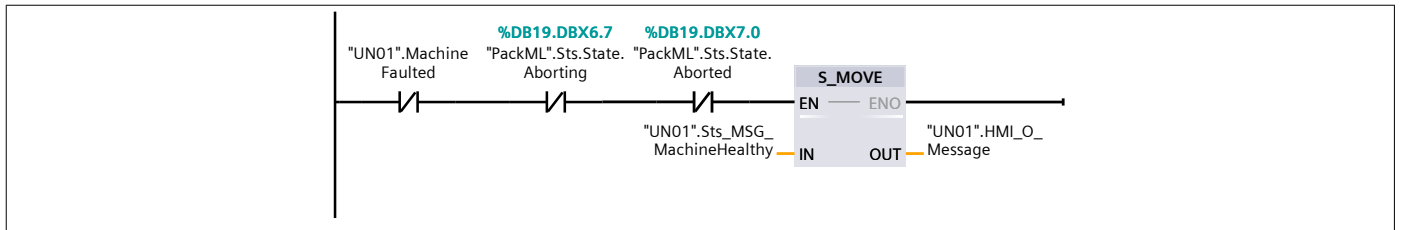
Network 13:



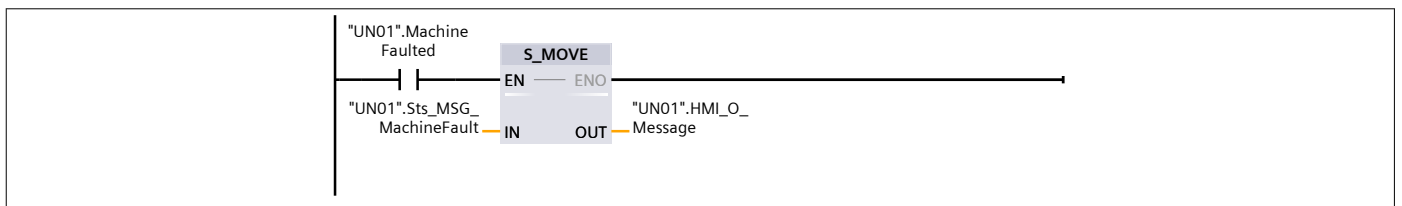
Network 14:



Network 15:



Network 16:



Program blocks / 0_UN01_ExampleMachine

UN01_SR20_Initialize [FB103]

UN01_SR20_Initialize Properties

General

Name	UN01_SR20_Initialize	Number	103	Type	FB
Language	LAD	Numbering	Manual		

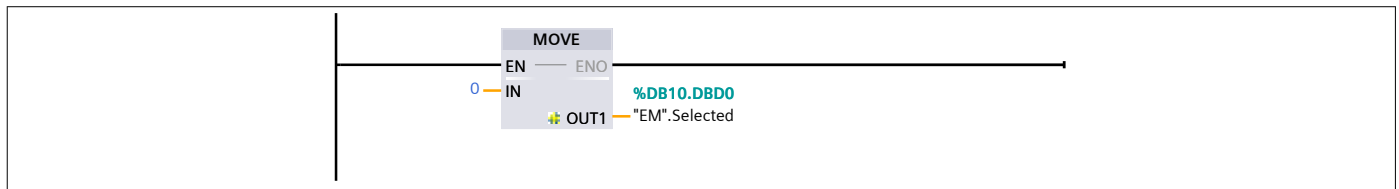
Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Temp_Dint	DInt	0	Non-retain
Temp_Bool	Bool	false	Non-retain
Temp			
Constant			

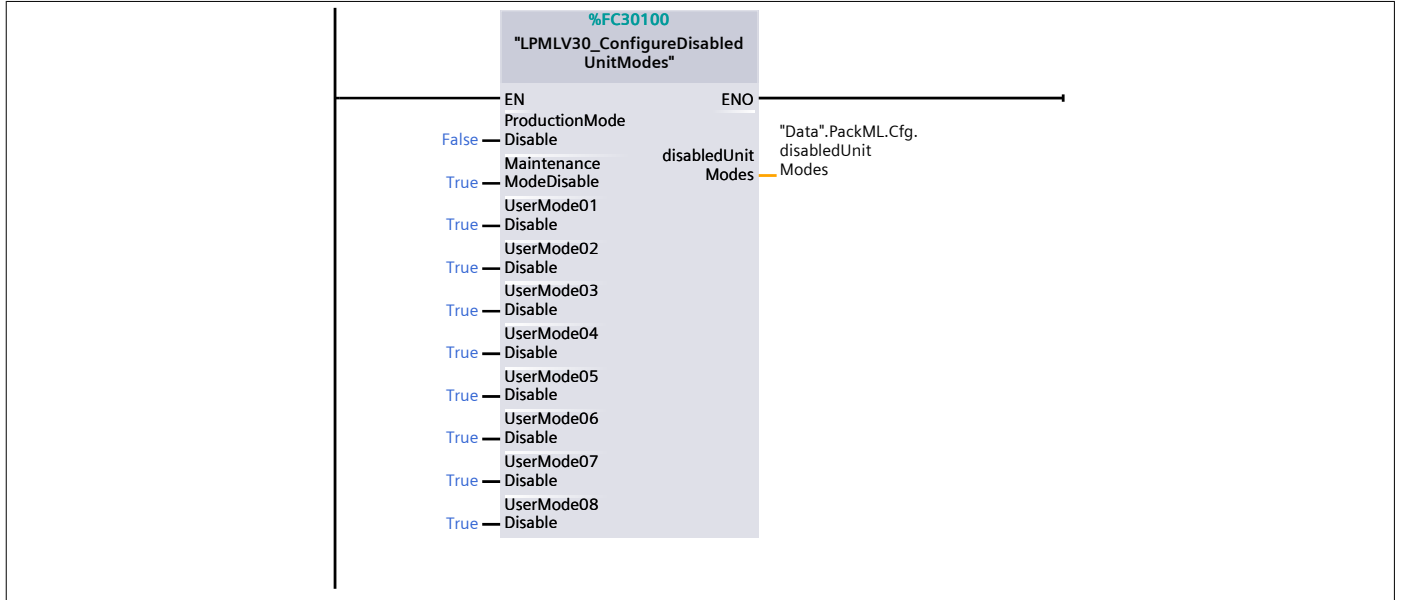
Network 1: Clear EM selected

Each EM will turn on its bit as it initializes



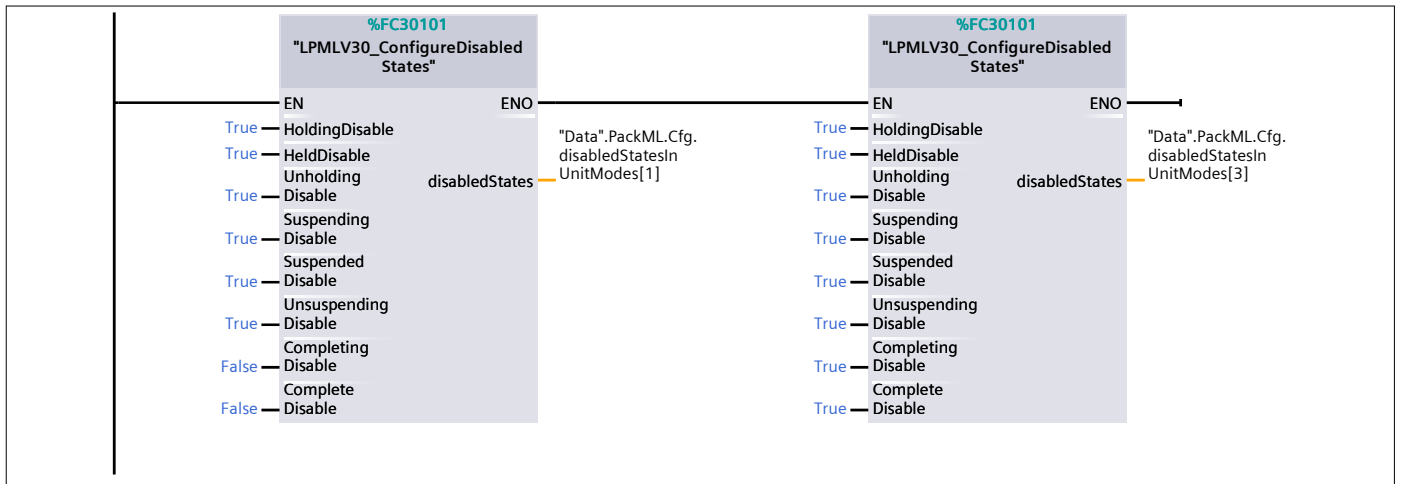
Network 2: Disable modes

Only enable modes 1 (Production) and 3 (Manual) Note that Manual can not be disabled.



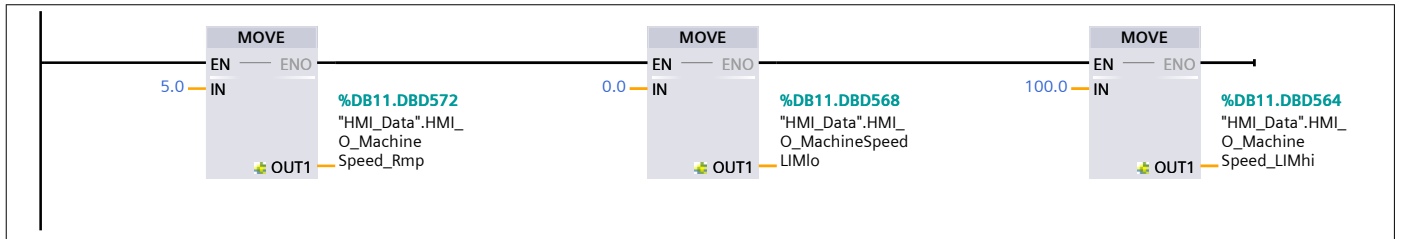
Network 3: Set disabled states

For both Production and Manual modes

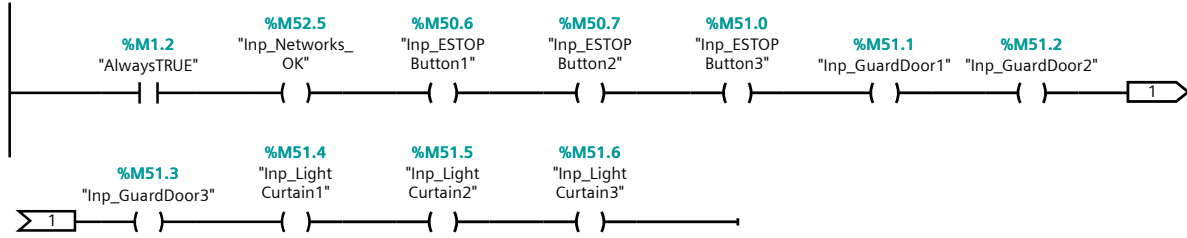


Network 4: Initialize HMI machine speed limits

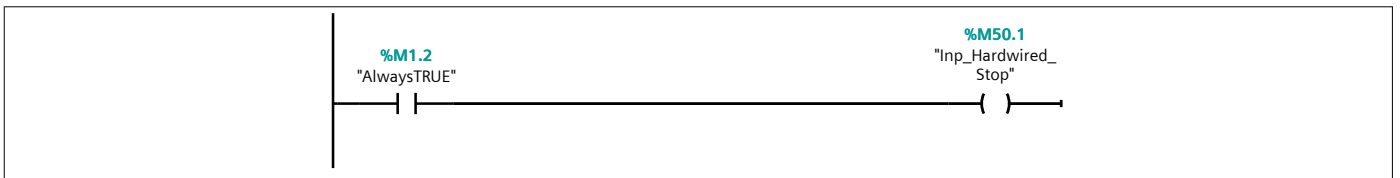
Ramp increment
Machine speed minimum
Machine speed maximum



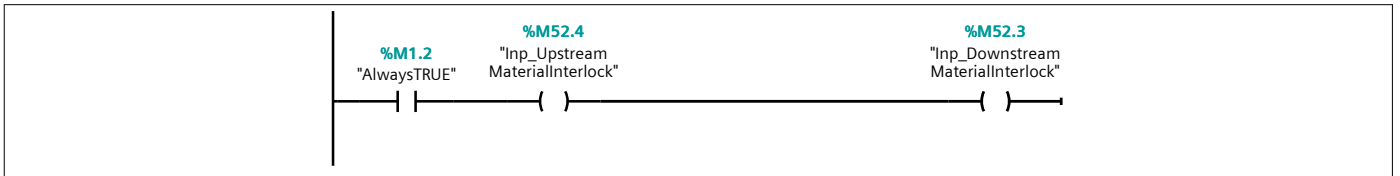
Network 5: Unused safety and status inputs



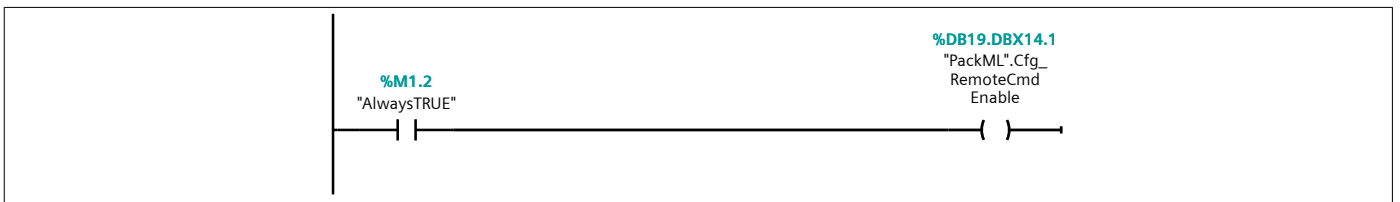
Network 6: Disabled unused stop



Network 7: Disable interlocks



Network 8: Enable remote commands



Program blocks / 0_UN01_ExampleMachine

UN01_UP01_PackML [FB105]

UN01_UP01_PackML Properties

General

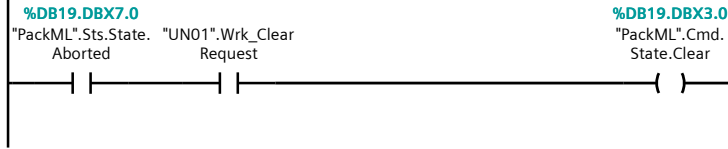
Name	UN01_UP01_PackML	Number	105	Type	FB
Language	LAD	Numbering	Manual		

Information

Title	*** PackML Template *** Controls the machines modes & states	Author	FoodBev	Comment	The
Family	FoodBev	Version	1.0	User-defined ID	OMAC

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
LPMLV30_UnitModeStateManager_Instance	"LPMLV30_UnitModeStateManager"		
LPMLV30_UnitModeStateTimes_Instance	"LPMLV30_UnitModeStateTimes"		
RemoteCommand_ModeChageRequest	Struct		Non-retain
SC_InputToBlock	Bool	false	Non-retain
SC_TrigStorage	Bool	false	Non-retain
▼ Temp			
temp_ReadClock_RetVal	Int		
temp_HMI_RetVal	Int		
temp_OEE_Availability	Real		
temp_OEE_Performance	Real		
temp_OEE_Quality	Real		
temp_Ret_Val	Int		
temp_Int	Int		
temp_string	String[10]		
temp_DisabledStates	DInt		
temp_ModeCurrent	DInt		
temp_StateCurrent	DInt		
temp_StateRequested	DInt		
Constant			

Network 1: PackML ModeManager & State Model - Clear Command

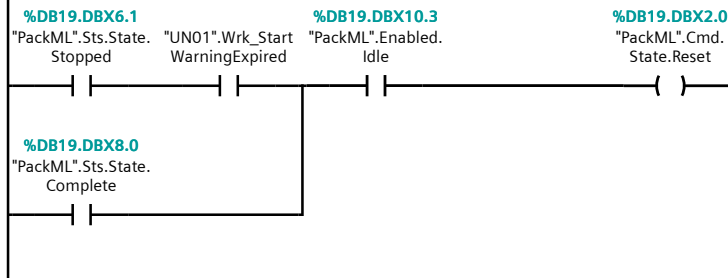


Network 2: PackML ModeManager & State Model - Reset command

STATE COMMAND
RESET

This Command is Triggered by the Unit Condition Indicating the Start Warning Cycle Has Completed. The Command Initiates in the PackML FB for the Current Mode Operation Procedure to Transition From the Stopped or Complete State to:

- 1) The Resetting State, If Resetting Is Enabled
- 2) The Idle State, If Resetting is Disabled and Idle Is Enabled
- 3) The Starting State, If Resetting and Idle Are Disabled and Starting is Enabled
- 4) The Execute State, If Resetting, Idle, and Starting Are Disabled

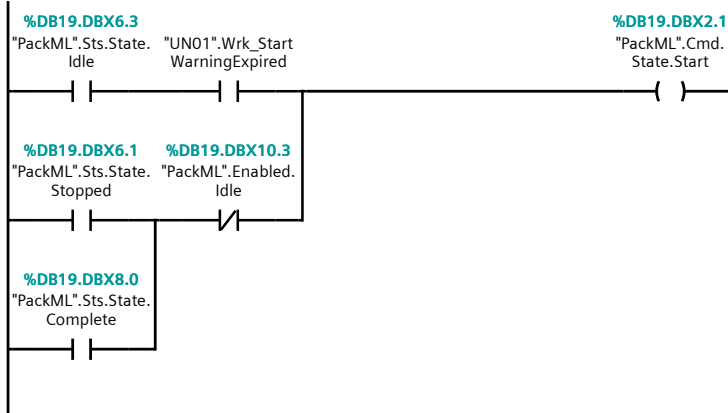


Network 3: PackML ModeManager & State Model - Start command

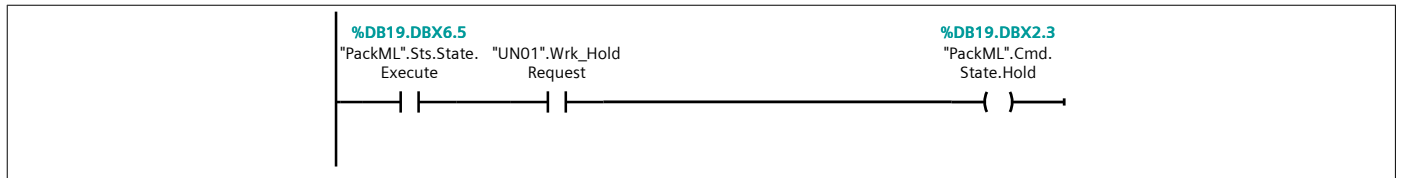
STATE COMMAND
START

This Command is Triggered by the Unit Condition Indicating the Start Warning Cycle Has Completed. The Command Initiates in the PackML FB for the Current Mode Operation Procedure to Transition From the Idle State to:

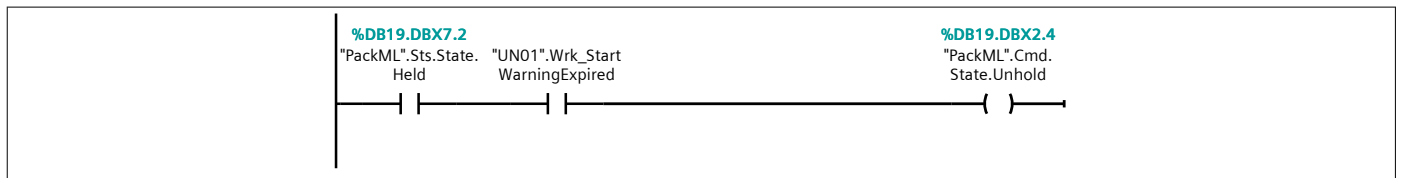
- 1) The Starting State, If Starting is Enabled
- 2) The Execute State, If Starting is Disabled



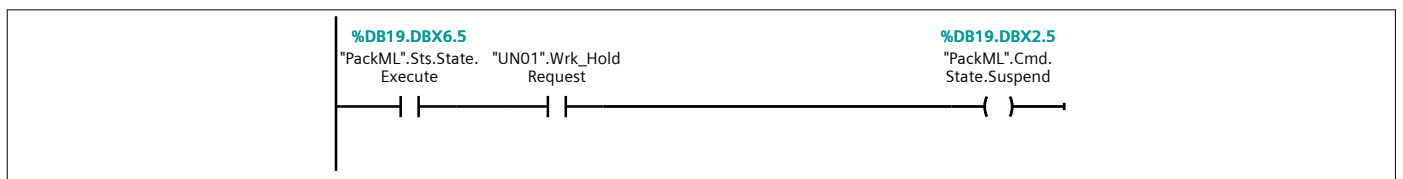
Network 4: PackML ModeManager & State Model - Hold command



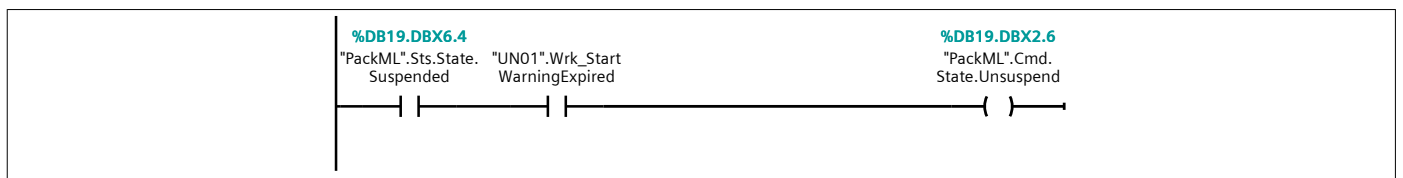
Network 5: PackML ModeManager & State Model - UnHold command



Network 6: PackML ModeManager & State Model - Suspend command



Network 7: PackML ModeManager & State Model - UnSuspend command

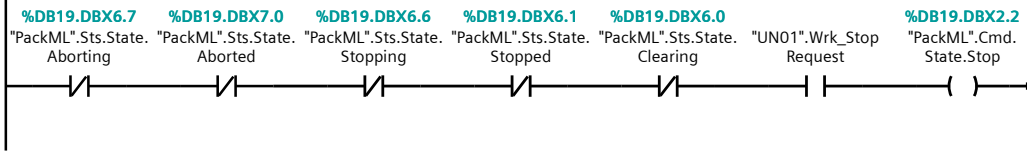


Network 8: PackML ModeManager & State Model - Stop command

STATE COMMAND
STOP

This Command is Triggered by the Unit Condition Indicating Any Stop Condition Is Present. The Command Initiates in the PackML AOI for the Current Mode Operation Procedure to Transition From the Following List of States (Resetting, Idle, Starting, Execute, Holding, Held, UnHolding, Suspending, Suspended, UnSuspending, Completing, Or Complete), When Each Is Enabled, To:

- 1) The Stopping State, If Stopping is Enabled
- 2) The Stopped State, If Stopping is Disabled



Network 9: PackML ModeManager & State Model - Abort command

STATE COMMAND
ABORT

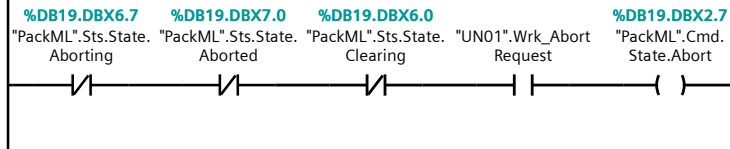
This Command is Triggered by the Unit Condition Indicating Any Fault Condition Is Present. The Command Initiates in the PackML FB for the Current Mode Operation Procedure to Transition From the Following List of States

(Resetting, Idle, Starting, Execute, Holding, Held, UnHolding, Suspending, Suspended, UnSuspending, Completing, Complete, Stopping, Stopped, Or Clearing),

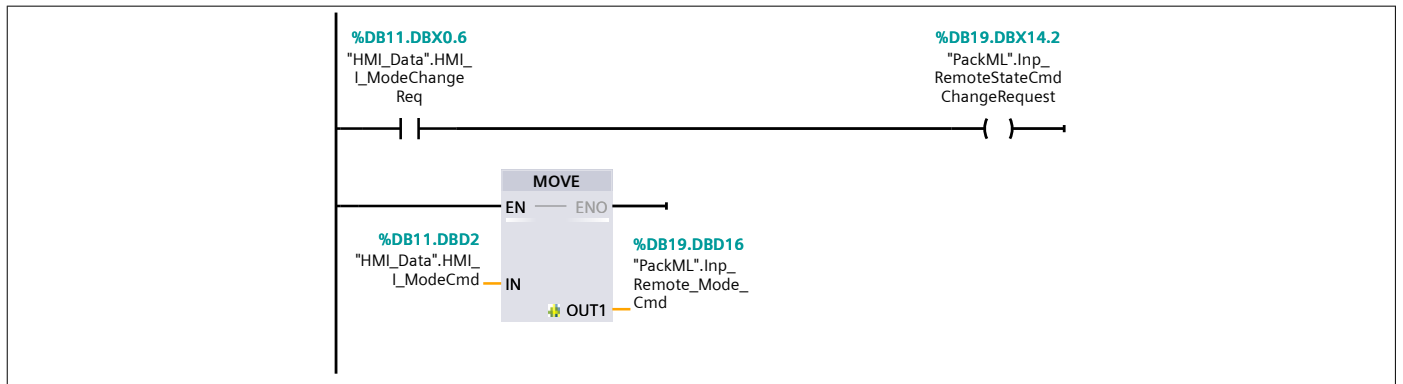
When Each Is Enabled, To:

- 1) The Aborting State, If Aborting is Enabled
- 2) The Aborted State, If Aborting is Disabled

MAKE SURE THAT ALL EQUIPMENT MODULES REPORT BACK A FAULTRESET_DONE AT SOME POINT IN TIME (EVEN IF FAULTS ARE PRESENT) TO AVOID THAT THE STATEMACHINE GETS STUCK IN CLEARING



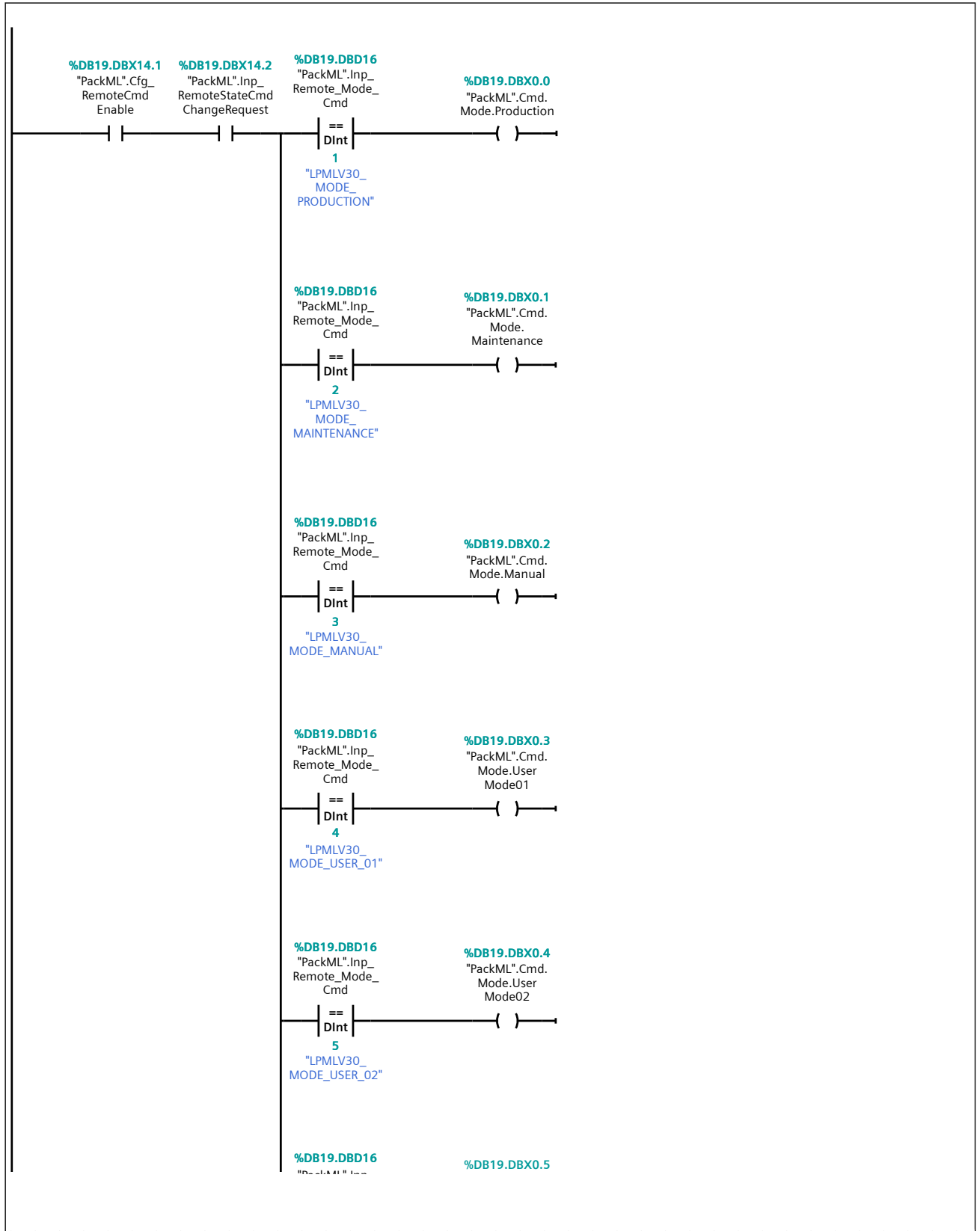
Network 10: Mode change requests from operator



Network 11: External PackML Mode Commands

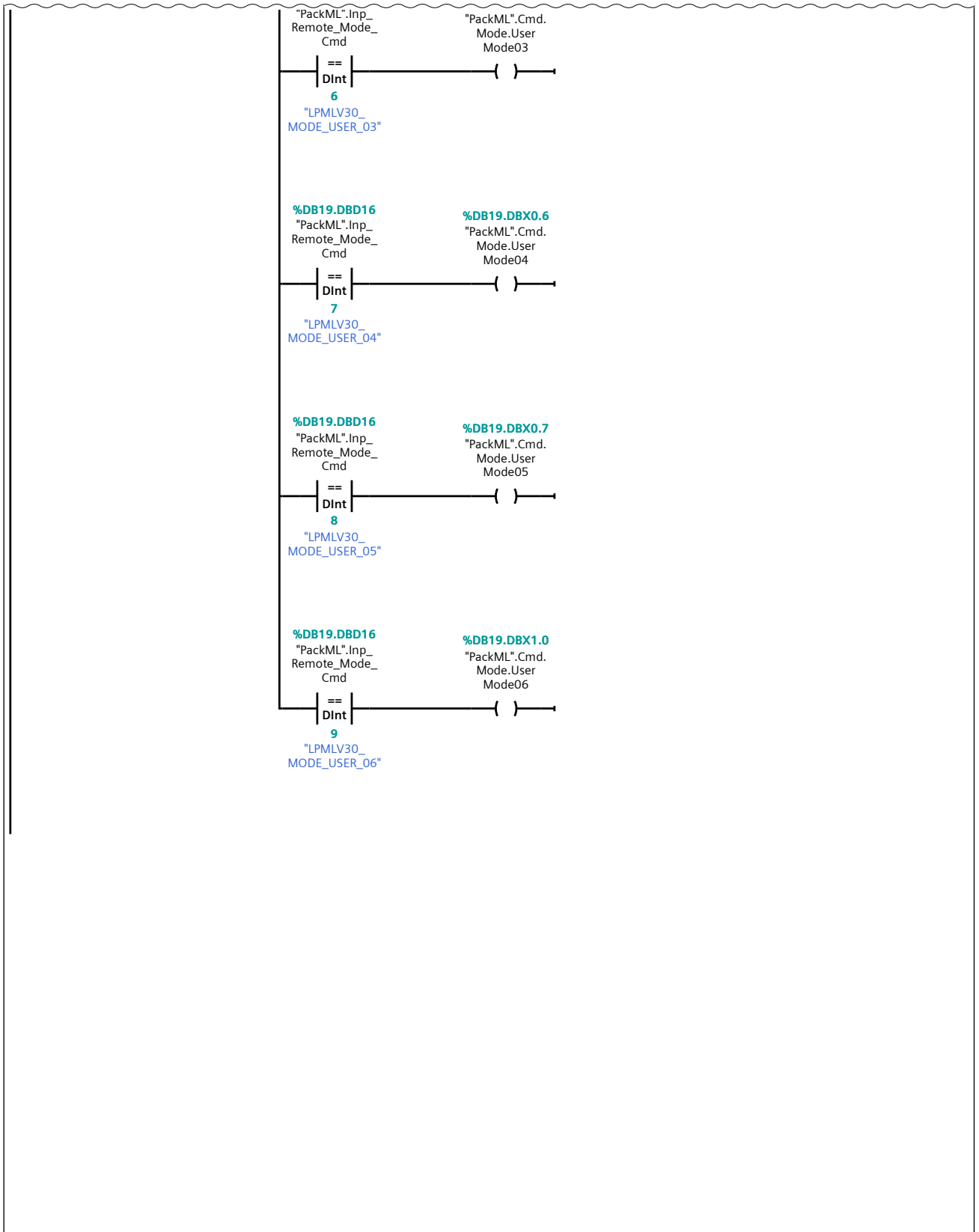
Rung from original program from CPG modified to use "PackML".Cmd.Mode. bits to change mode.

Network 11: External PackML Mode Commands (1.1 / 2.1)



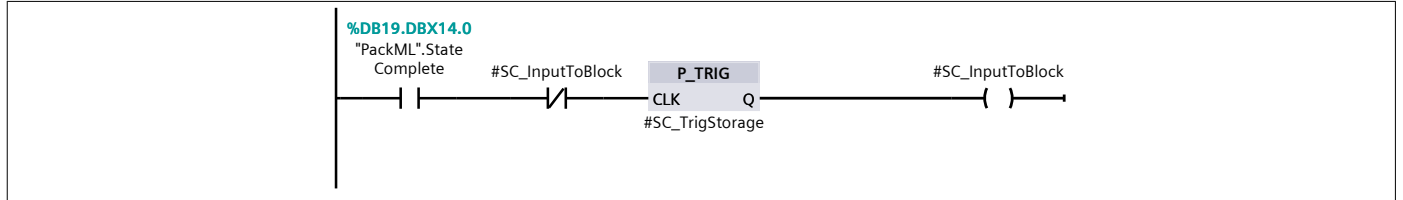
Network 11: External PackML Mode Commands (2.1 / 2.1)

1.1 (Page19 - 5)



Network 12: Enforce change in SC input to block

If state changes, but SC does not change because the state complete conditions are fulfilled as soon as the state is active, the SC input to LPMLV30_StateManager must still see a transition. This logic makes sure that the SC input is off at the next scan after it turns on.



Network 13: Call thePackML Mode Manager & State Model FB

Network 13: Call thePackML Mode Manager & State Model FB (1.1 / 2.1)

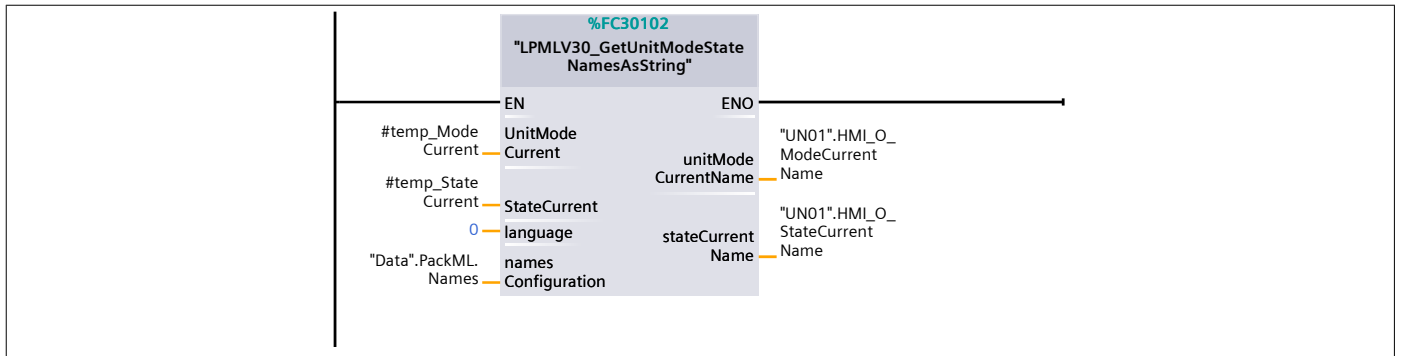


Network 13: Call thePackML Mode Manager & State Model FB (2.1 / 2.1)

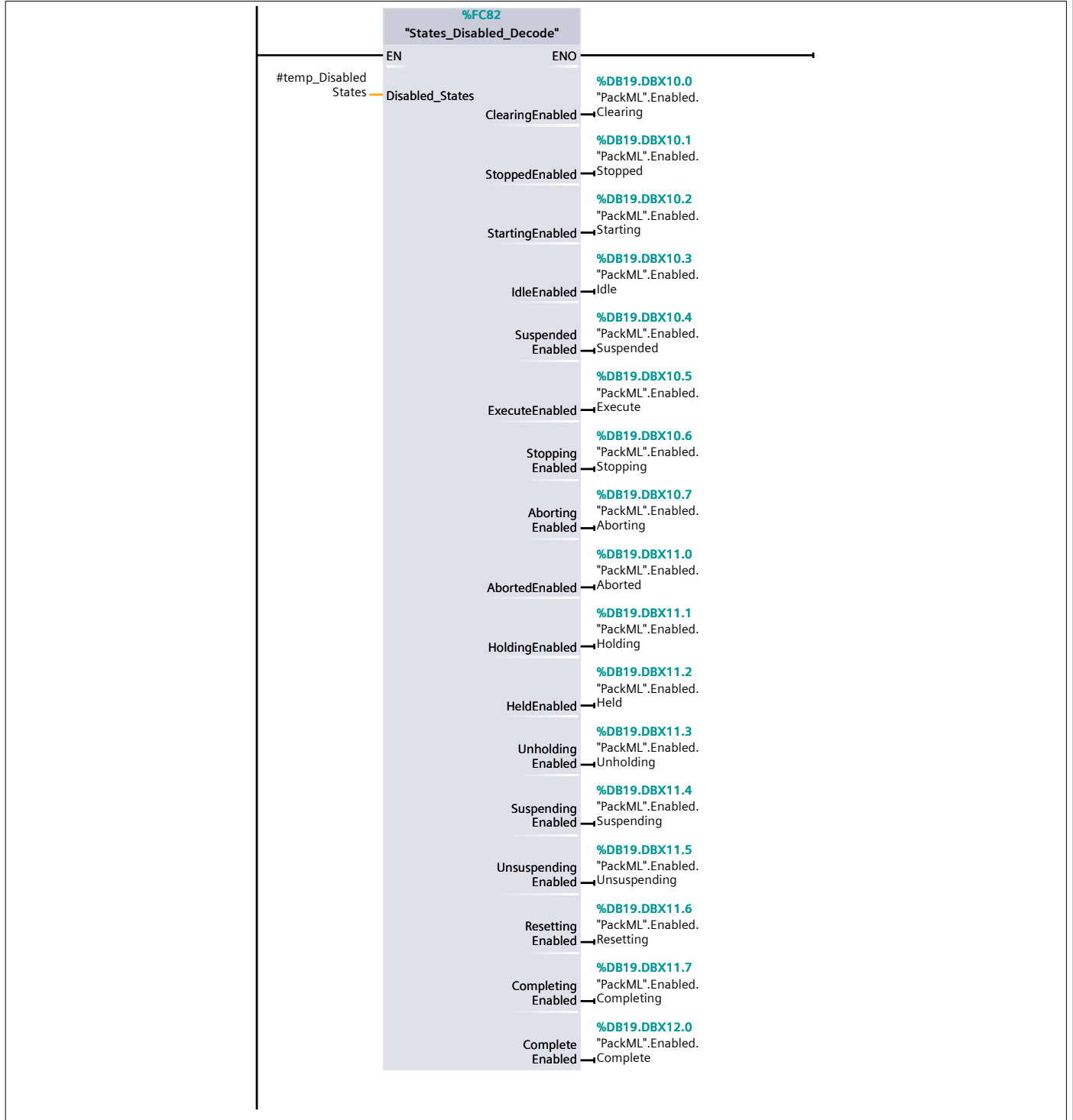
1.1 (Page19 - 8)



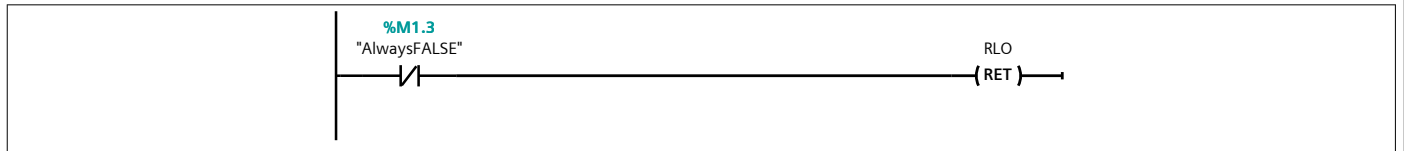
Network 14: Mode / State Names.



Network 15: Decode disabled states



Network 16: END: Update the ENO Output. (DO NOT REMOVE. Must be last rung)



Program blocks / 0_UN01_ExampleMachine

UN01_UP02_StateComplete [FB106]

UN01_UP02_StateComplete Properties

General

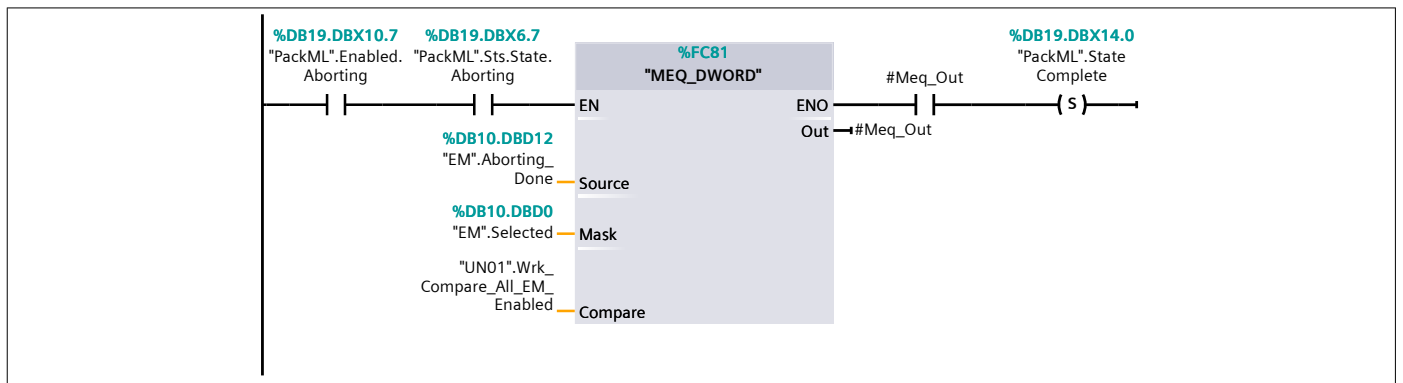
Name	UN01_UP02_StateComplete	Number	106	Type	FB
Language	LAD	Numbering	Manual		

Information

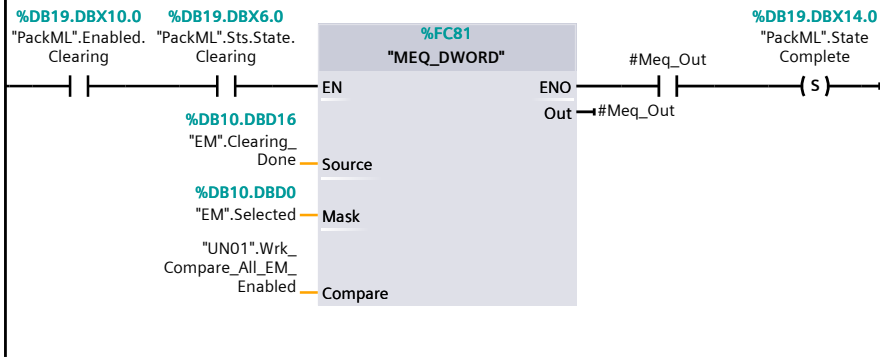
Title	State Complete Logic	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Temp_Dint	DInt	0	Non-retain
Temp_Bool	Bool	false	Non-retain
▼ Temp			
Meq_Out	Bool		
Constant			

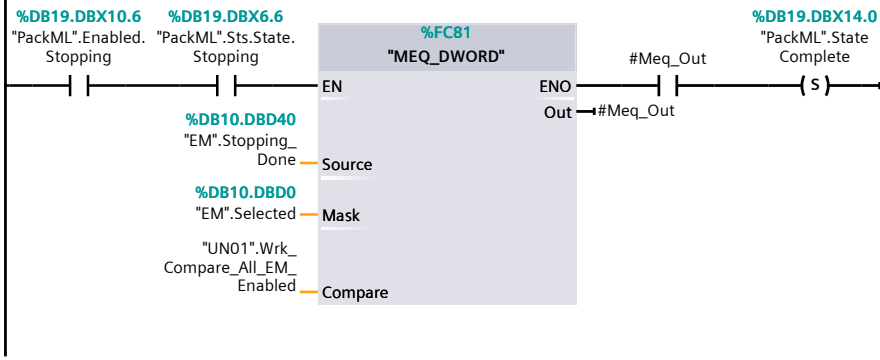
Network 1: Aborting state complete



Network 2: Clearing state complete

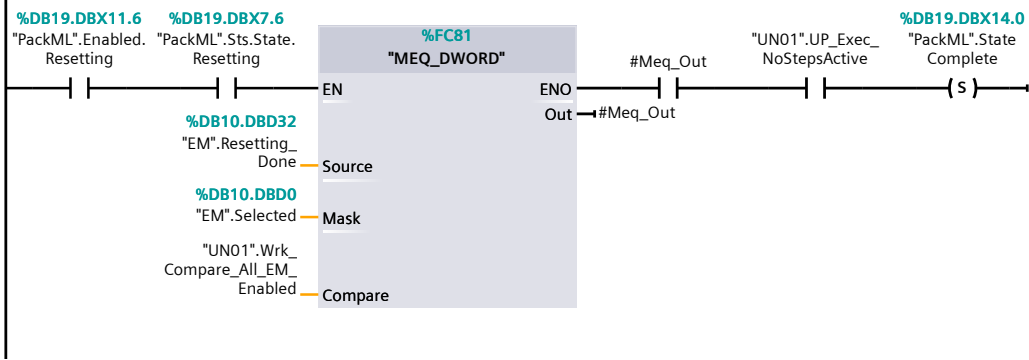


Network 3: Stopping state complete

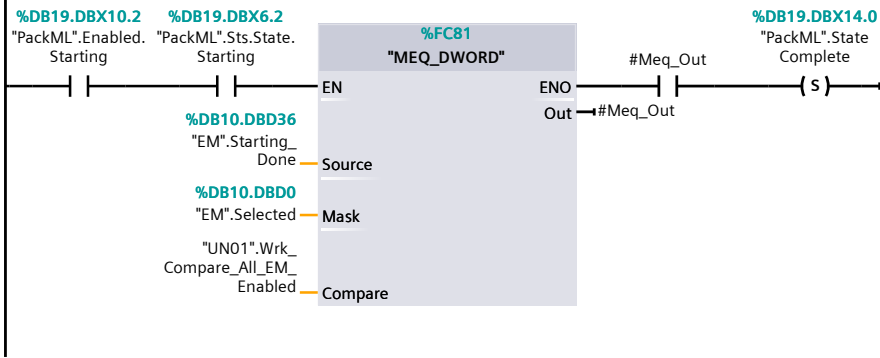


Network 4: Resetting state complete

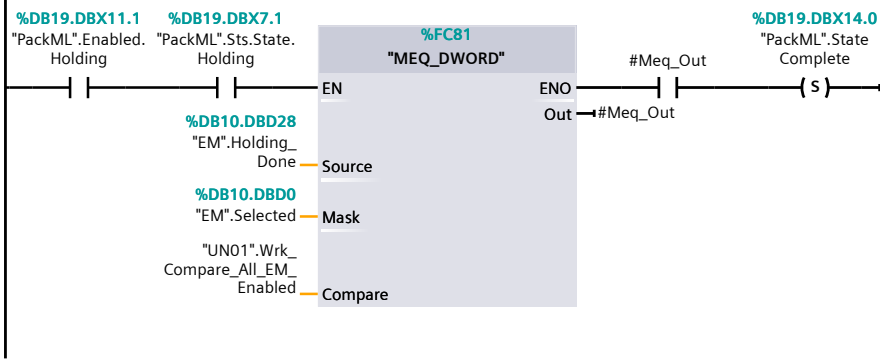
For resetting, make sure unit procedure initialized



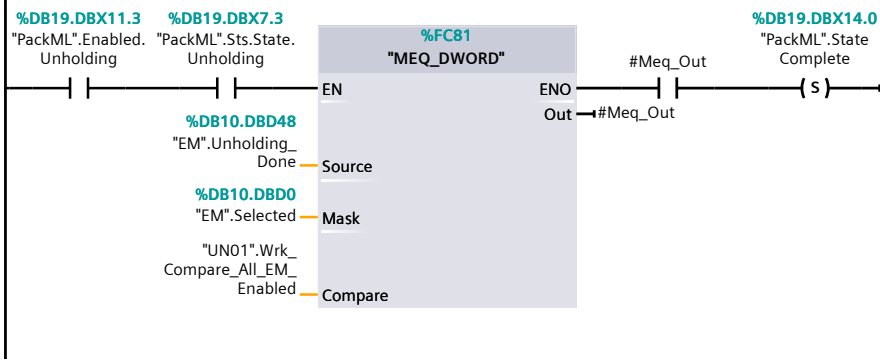
Network 5: Starting state complete



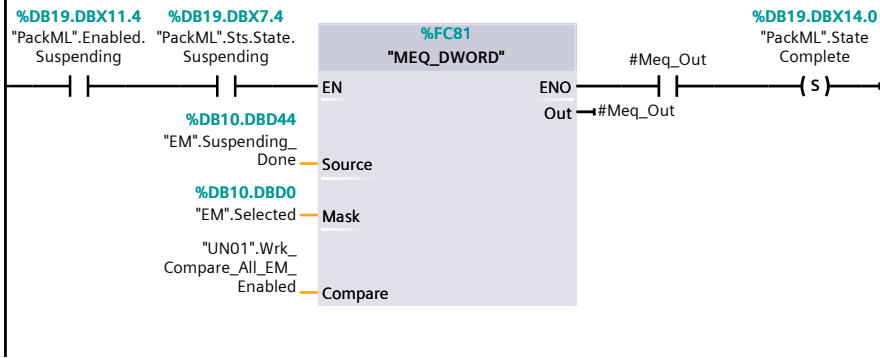
Network 6: Holding state complete



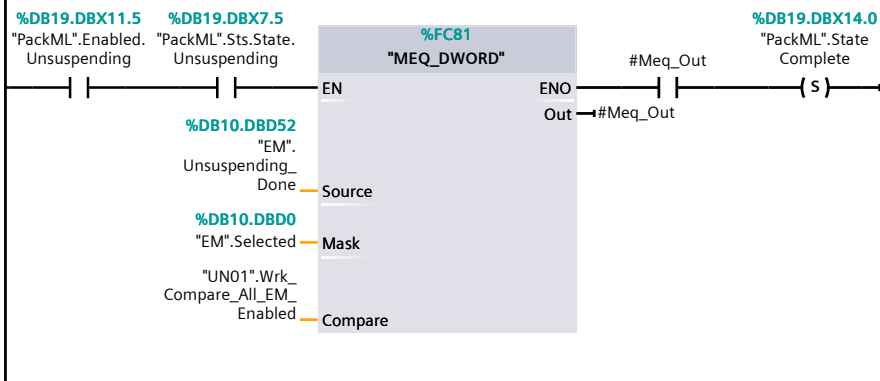
Network 7: Unholding state complete



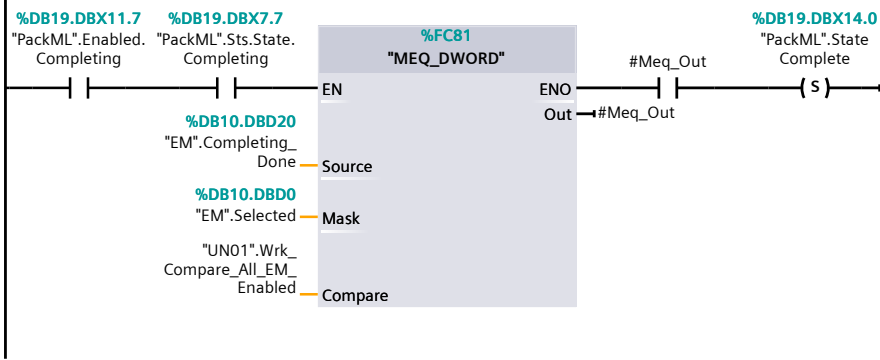
Network 8: Suspending state complete



Network 9: Unsuspending state complete

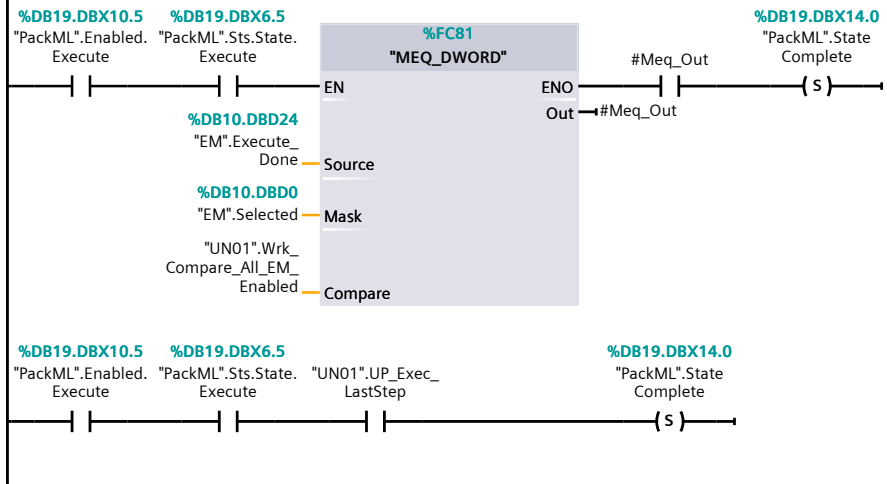


Network 10: Completing state complete



Network 11: Execute state complete

Complete when all EM's complete, or when unit procedure finished.



Program blocks / 0_UN01_ExampleMachine

UN01 [DB100]

UN01 Properties

General

Name	UN01	Number	100	Type	DB
Language	DB	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Start value	Retain
Input			
Output			
InOut			
▼ Static			
HMI_O_ModeCurrentName	String[18]	"	False
HMI_O_StateCurrentName	String[16]	"	False
HMI_O_MachineSafetyFault	DInt	0	False
HMI_O_MajorFaultMessage	DInt	0	False
HMI_O_MinorFaultMessage	DInt	0	False
HMI_O_Message	String[18]	"	False
Sts_GoodProducts	DInt	0	False
Sts_MSG_MachineFault	String	'Machine Fault'	False
Sts_MSG_MachineHealthy	String	'Machine Healthy'	False
MachineFault_Major	Bool	false	False
MachineFault_Minor	Bool	false	False
MachineFaulted	Bool	false	False
UP_Exec_Step	DInt	0	False
UP_Exec_StepDn	Array[0..32] of DInt		False
UP_Exec_LastStep	Bool	false	False
UP_Exec_NoStepsActive	Bool	false	False
Inp_BatchCountSetpoint	DInt	0	False
Wrk_StartRequest	Bool	false	False
Wrk_StopRequest	Bool	false	False
Wrk_ResetRequest	Bool	false	False
Wrk_AbortRequest	Bool	false	False
Wrk_ClearRequest	Bool	false	False
Wrk_CompleteRequest	Bool	false	False
Wrk_HoldRequest	Bool	false	False
Wrk_UnHoldRequest	Bool	false	False
Wrk_SuspendRequest	Bool	false	False
Wrk_Compare_All_EM_Enabled	DInt	-1	False
Wrk_AllSafetyOK	Bool	false	False
Wrk_EstopOK	Bool	false	False
Wrk_GuardsOK	Bool	false	False
Wrk_LightCurtainsOK	Bool	false	False

Totally Integrated
Automation Portal

Name	Data type	Start value	Retain
Wrk_NetworkCommunicationOK	Bool	false	False
Wrk_StartWarningExpired	Bool	false	False
S20_InitializeData	"UN01_SR20_Initialize"		False
CM01_OperationLocal	"UN01_CM01_OperationLocal"		False
CM03_FaultHandler	"UN01_CM03_FaultHandler"		False
UP02_StateComplete	"UN01_UP02_StateComplete"		False
UP01_PackML	"UN01_UP01_PackML"		False
UP00_Procedure	"UN01_UP00_Procedure"		False

Program blocks / 0_UN01_ExampleMachine

UN01_UP00_Procedure [FB104]

UN01_UP00_Procedure Properties

General

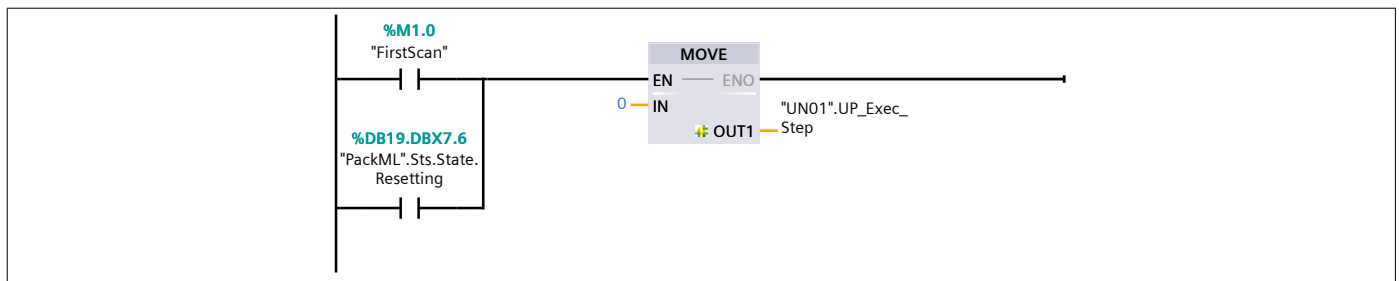
Name	UN01_UP00_Procedure	Number	104	Type	FB
Language	LAD	Numbering	Manual		

Information

Title	Execute State motion steps/commands for over-all operation	Author		Comment	
Family		Version	0.1	User-defined ID	

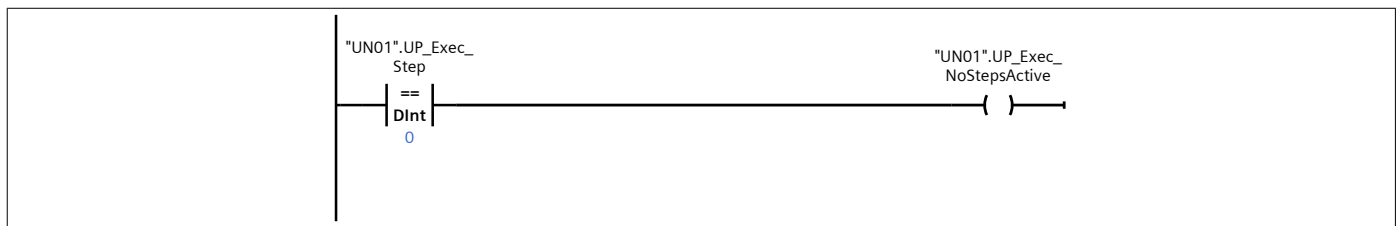
Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Wrk_Delay1	TON_TIME		Non-retain
Wrk_Delay2	TON_TIME		Non-retain
Wrk_Delay3	TON_TIME		Non-retain
Wrk_Delay4	TON_TIME		Non-retain
Push	Bool	false	Non-retain
Temp			
Constant			

Network 1: First scan or resetting - clear all steps



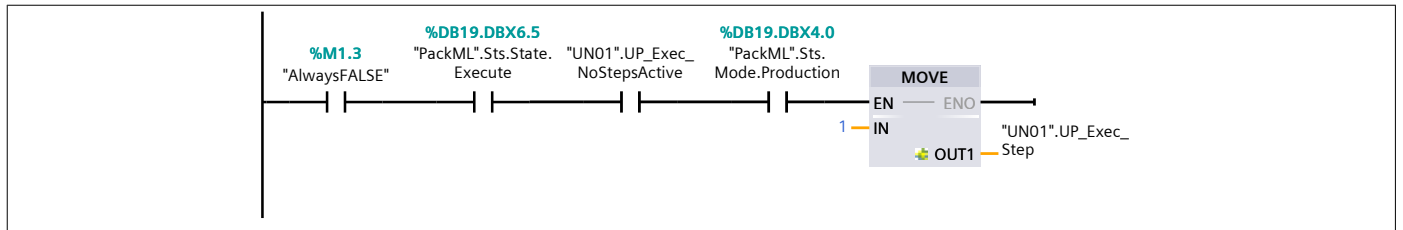
Network 2: Indication that all steps reset

Used by RESETTING state complete logic and initial execute step

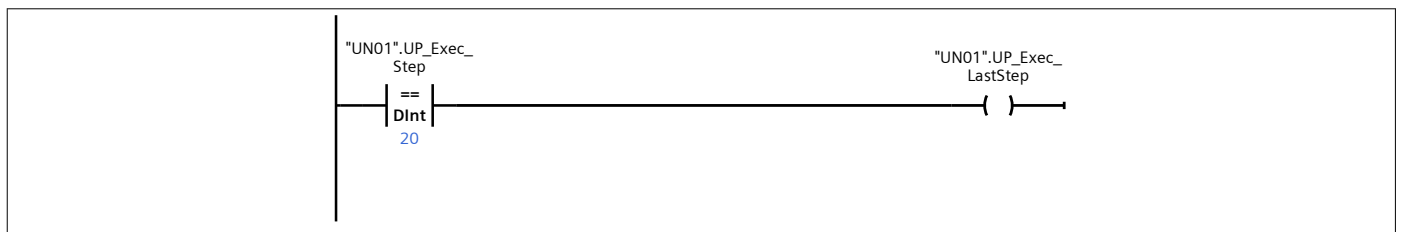


Network 3: EXECUTE - start Disabled. All procedures are in the EM's

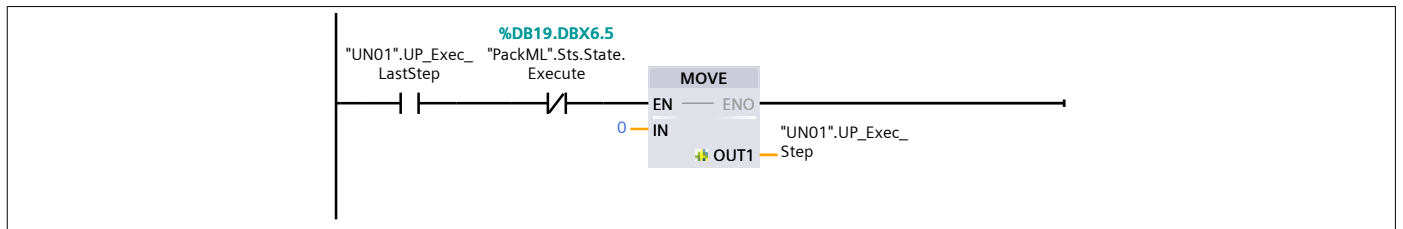
If no steps active, start in step 1



Network 4:



Network 5: Step 20 - When out of Execute, move zero into step number so it will start over at next push of start



Program blocks / Data

EM [DB10]

EM Properties

General

Name	EM	Number	10	Type	DB
Language	DB	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Start value	Retain
▼ Static			
Selected	DInt	0	False
Faulted	DInt	0	False
No_Motion	DInt	0	False
Aborting_Done	DInt	0	False
Clearing_Done	DInt	0	False
Completing_Done	DInt	0	False
Execute_Done	DInt	0	False
Holding_Done	DInt	0	False
Resetting_Done	DInt	0	False
Starting_Done	DInt	0	False
Stopping_Done	DInt	0	False
Suspending_Done	DInt	0	False
Unholding_Done	DInt	0	False
Unsuspending_Done	DInt	0	False
Fault	Array[0..31] of "EM_Faults"		False

Program blocks / Data

PackML [DB19]

PackML Properties

General

Name	PackML	Number	19	Type	DB
Language	DB	Numbering	Automatic		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Start value	Retain
▼ Static			
Cmd	Struct		False
Sts	Struct		False
Enabled	Struct		False
StateComplete	Bool	false	False
Cfg_RemoteCmdEnable	Bool	false	False
Inp_RemoteStateCmdChangeRequest	Bool	false	False
Inp_Remote_Mode_Cmd	DInt	0	False

Program blocks / Data

Axis_1_Status [DB75]

Axis_1_Status Properties

General

Name	Axis_1_Status	Number	75	Type	DB
Language	DB	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Start value	Retain
▼ Static			
Status	"Servo_Status-Word_Type"		False
Error	"Servo_Error-Word_Type"		False
Warning	"Servo_Warning-Word_Type"		False

Program blocks / Data

HMI_Data [DB11]

HMI_Data Properties

General

Name	HMI_Data	Number	11	Type	DB
Language	DB	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Start value	Retain
▼ Static			
HMI_I_Start	Bool	false	True
HMI_I_Stop	Bool	false	True
HMI_I_Clear	Bool	false	True
HMI_I_Reset	Bool	false	True
HMI_I_Hold	Bool	false	True
HMI_I_Unhold	Bool	false	True
HMI_I_ModeChangeReq	Bool	false	True
HMI_I_JogNegative	Bool	false	True
HMI_I_JogPositive	Bool	false	True
HMI_I_ModeCmd	DInt	0	True
HMI_I_Select_JogAxis	DInt	0	True
HMI_I_MachineSpeed_Cmd	Real	0.0	True
HMI_I_BeltSpeed	Real	100.0	True
HMI_I_MoveSpeed	Real	10.0	True
HMI_I_JerkTime	Real	50.0	True
HMI_I_YMovePosition	Real	100.0	True
HMI_I_YMoveSpeed	Real	50.0	True
HMI_O_XCoord	Real	50.0	True
HMI_O_YCoord	Real	35.0	True
HMI_O_ZCoord	Real	50.0	True
HMI_O_RCoord	Real	50.0	True
HMI_O_LAAborted	Bool	false	True
HMI_O_LAExecute	Bool	false	True
HMI_O_LAIIdle	Bool	false	True
HMI_O_LAStopped	Bool	false	True
HMI_O_GlobalModeCurrentName	String	"	True
HMI_O_GlobalStateCurrentName	String	"	True
HMI_O_MachineSpeed_LIMhi	Real	0.0	True
HMI_O_MachineSpeedLIMlo	Real	0.0	True
HMI_O_MachineSpeed_Rmp	Real	0.0	True
HMI_O_EM1_Step	Int	0	True

Program blocks / Data

Axis_2_Status [DB76]

Axis_2_Status Properties

General

Name	Axis_2_Status	Number	76	Type	DB
Language	DB	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Start value	Retain
▼ Static			
Status	"Servo_Status-Word_Type"		False
Error	"Servo_Error-Word_Type"		False
Warning	"Servo_Warning-Word_Type"		False

Program blocks / Data

Data [DB1]

Data Properties

General

Name	Data	Number	1	Type	DB
Language	DB	Numbering	Automatic		

Information

Title	Machine Data	Author		Comment	Machine Data
Family		Version	1.0	User-defined ID	

Name	Data type	Start value	Retain
▼ Static			
PackML	Struct		False
ProdSchedule_BoxesMoveSP	DInt	L#30000	True
ProdSchedule_BoxesMovePV	DInt	L#46	True
ProdSchedule_Complete	Bool	false	True
ProdSchedule_Reset	Bool	false	True
OOE_RealTime	Real	4.479167e-1	True
OOE_Availabilty	Real	0.0	False
OOE_Performance	Real	0.0	False
OOE_Quality	Real	0.0	False

Program blocks / EM01_Axis01_W

EM01_00_Main [FB200]

EM01_00_Main Properties

General

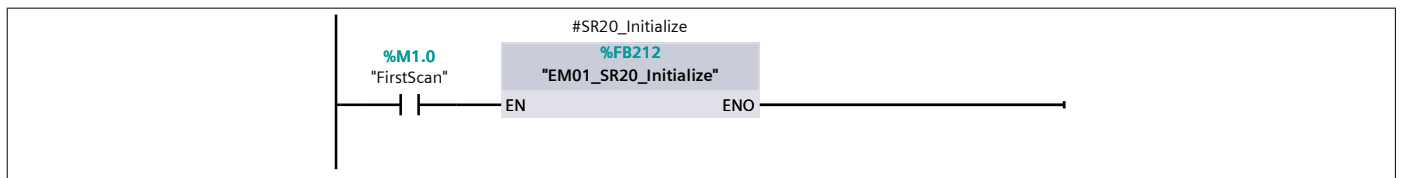
Name	EM01_00_Main	Number	200	Type	FB
Language	LAD	Numbering	Manual		

Information

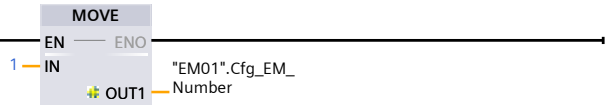
Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Cfg_EM_Number	Int	0	Non-retain
Par_Jog_Accel	Real	0.0	Non-retain
Par_Jog_Decel	Real	0.0	Non-retain
Par_Jog_Speed	Real	0.0	Non-retain
CM00_EM_Procedure	"EM01_CM00_Procedure"		
CM01_EMConditions	"EM01_CM01_EM-Conditions"		
CM02_01_Conv	"EM01_CM02_ServoAxisObject"		
CM03_01_AxisJog_Conv	"EM01_CM03_ServoAxisJog"		
SR03_FaultHandler	"EM01_SR03_Fault Handler"		
SR20_Initialize	"EM01_SR20_Initialize"		
Wrk_Temp_Bit	Bool	false	Non-retain
Temp			
Constant			

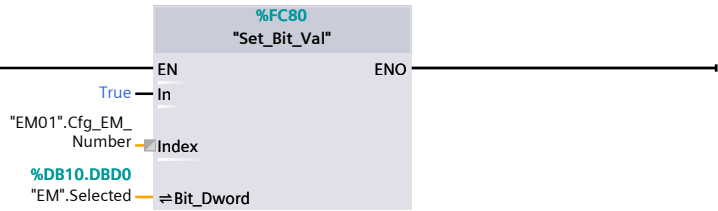
Network 1:



Network 2: Set EM number



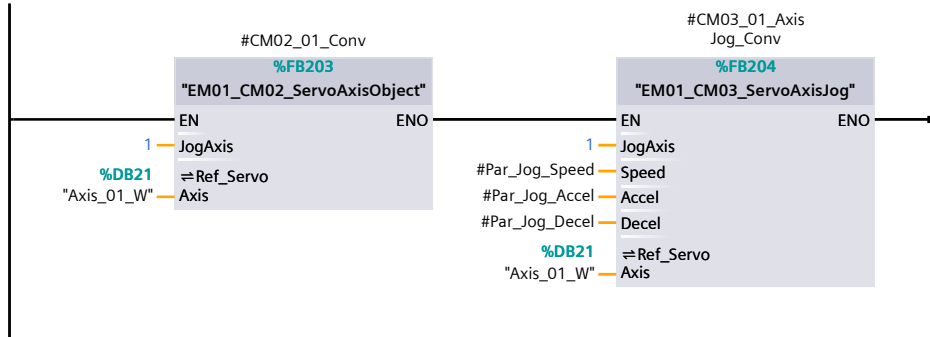
Network 3: Equipment module is selected and active



Network 4:



Network 5:



Network 6:

#CM01_EM
Conditions

#SR03_Fault
Handler

%FB202

%FB211

"EM01_CM01_EMConditions"

"EM01_SR03_FaultHandler"

EN

ENO

EN

ENO

Program blocks / EM01_Axis01_W

EM01_CM00_Procedure [FB201]

EM01_CM00_Procedure Properties

General

Name	EM01_CM00_Procedure	Number	201	Type	FB
Language	LAD	Numbering	Manual		

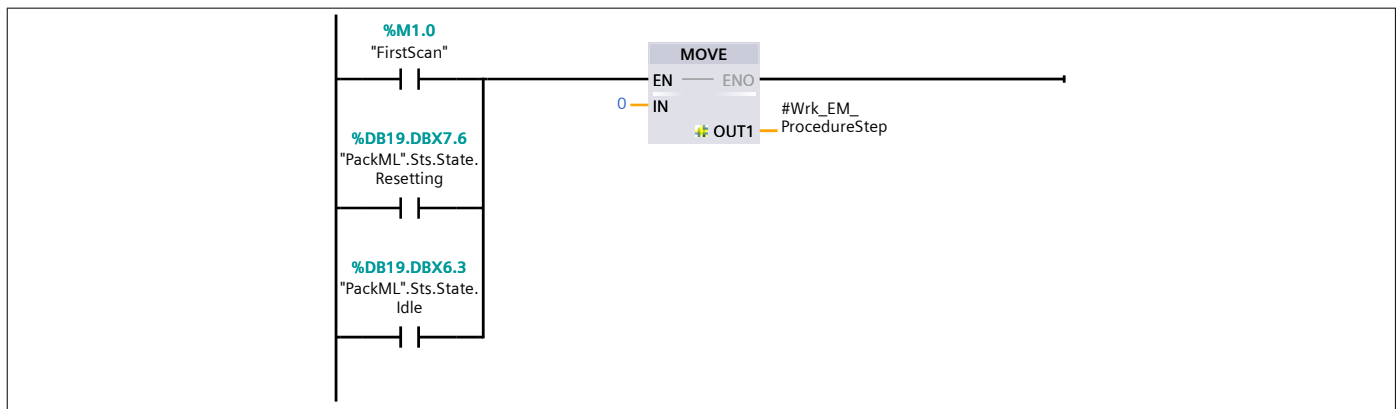
Information

Title	EM Procedure Sequence	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
ErrorID	Word	16#0	Non-retain
Sts_Err	Int	0	Non-retain
Wrk_EM_ProcedureStep	DInt	0	Non-retain
Wrk_EM_ProcedureNoStepsActive	Bool	false	Non-retain
Wrk_EM_ProcedureDone	Bool	false	Non-retain
Wrk_EM_ProcedureFault	Bool	false	Non-retain
Wrk_MC_MoveVel	MC_MOVEVELOCITY		
Wrk_MC_Home	MC_HOME		
Wrk_Trans_MoveVel	Bool	false	Non-retain
Wrk_Trans_Home	Bool	false	Non-retain
Web_Drive_Velocity	LReal	0.0	Non-retain
Web_Homed	Bool	false	Non-retain
Web_Speed	LReal	0.0	Non-retain
Temp			
Constant			

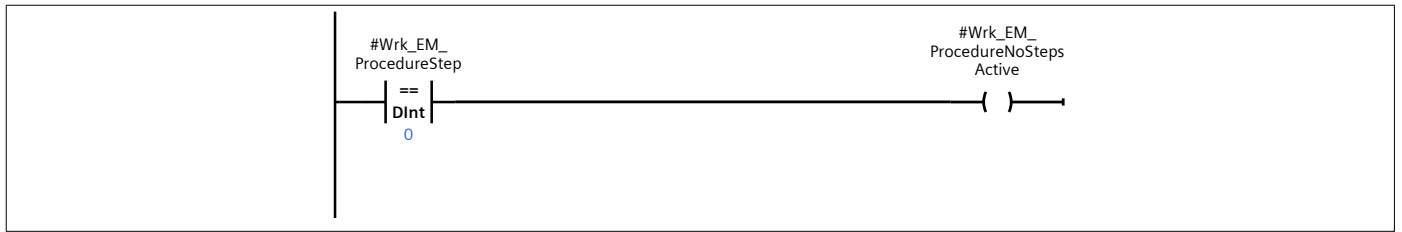
Network 1: Reset step number

First scan or resetting or idle - clear all steps



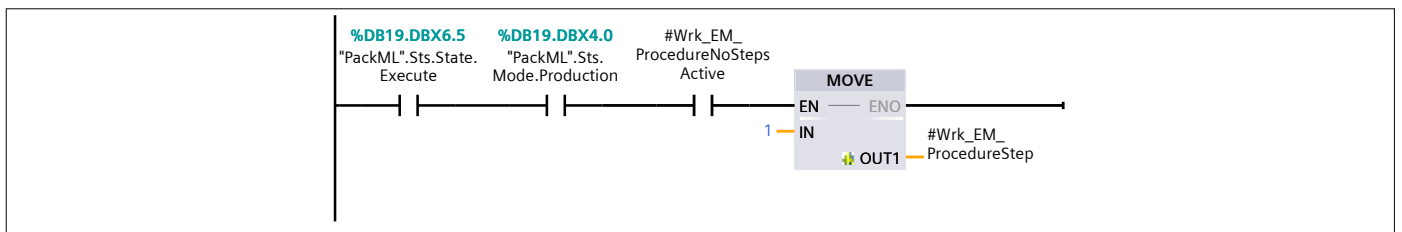
Network 2: Indication that all steps reset

Used by RESETING state and initial execute step



Network 3: EXECUTE - Sequence disabled.

If no steps active go to step 1

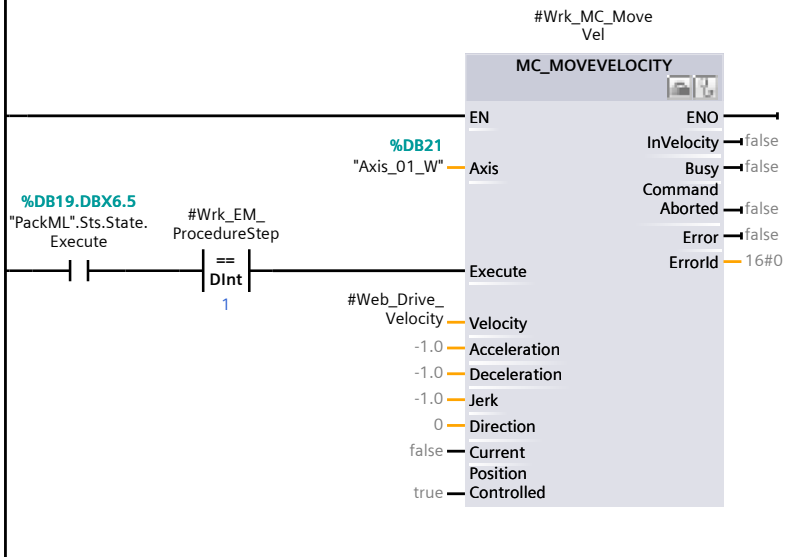


Network 4: Step 1 - Calculate web drive velocity

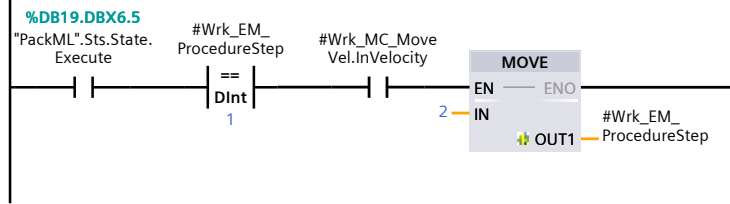
```

0001 IF #Wrk_EM_ProcedureStep = 1 THEN
0002     //
0003     // Make sure web velocity is > zero. If not, set to 100 mm/sec
0004     #Web_Speed := "HMI_Data".HMI_I_BeltSpeed;
0005     IF #Web_Speed <= 0.0 THEN
0006         #Web_Speed := 100.0;
0007     END_IF;
0008     // Calculate servo velocity in rad/sec, taking into account radius of web
    drive and transmission from motor to conveyor
0009     //
0010     #Web_Drive_Velocity := (#Web_Speed / ("Web_Drive_Pulley_Radi-
        us")) * ("Web_NTTeeth_Drive" / "Web_NTTeeth_Motor");
0011 END_IF;
    
```

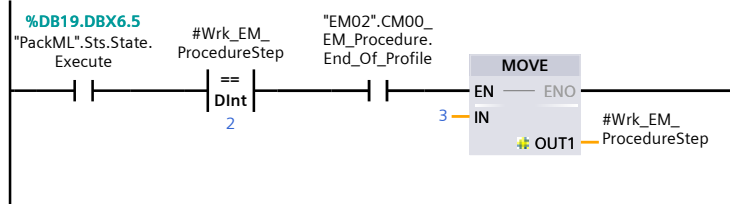
Network 5: Step 1 - Run web at desired speed.



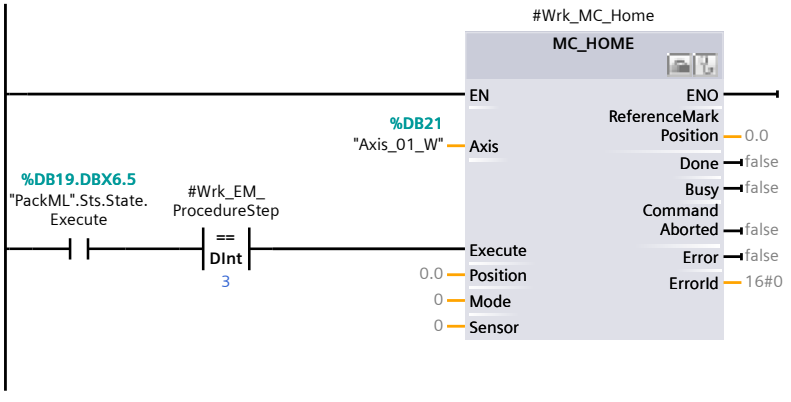
Network 6: Step 1 - Start web. Transition out when at desired speed



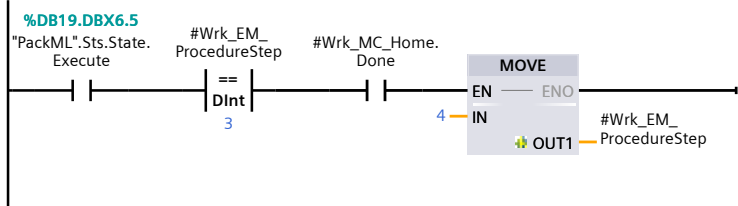
Network 7: Step 2 - Wait for cam end-of-profile



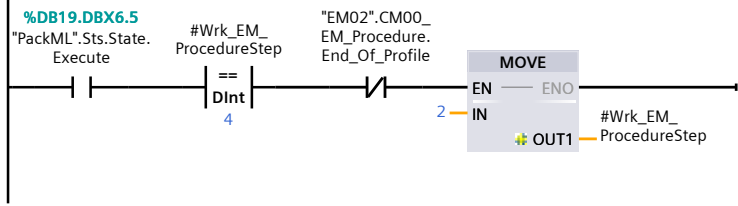
Network 8: Step 3 - Home web position



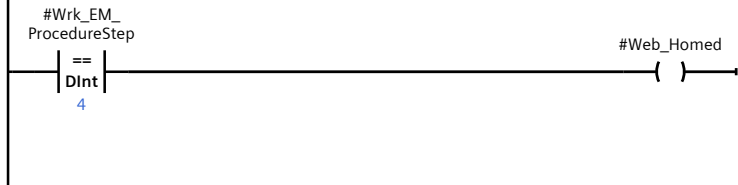
Network 9: Step 3 - Home axis to set position back to zero



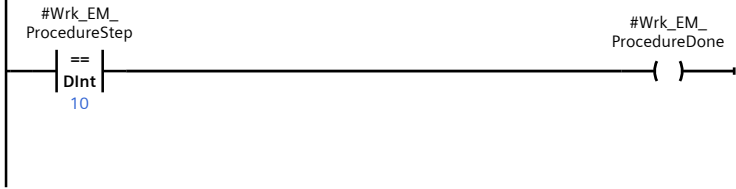
Network 10: Step 4 - Wait for end of cam profile indication off. Next step is to wait for cam end of profile



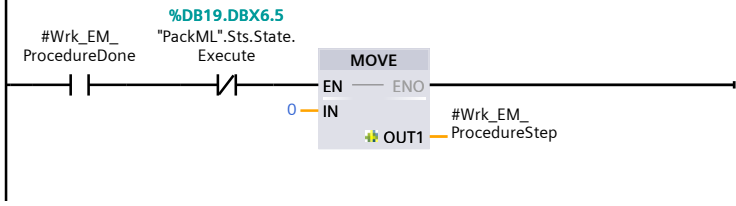
Network 11: Web homed indication in step 4



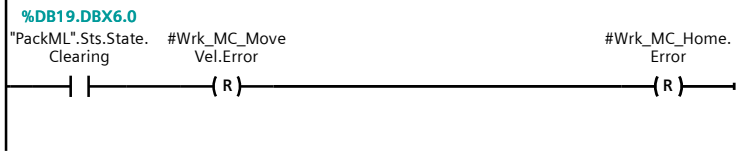
Network 12: Last step - set done



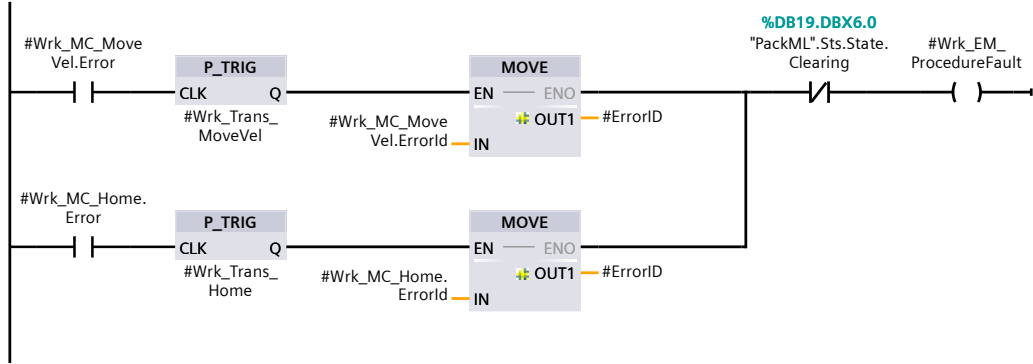
Network 13: When done, wait for out of execute step, then set step back to zero so can restart without resetting



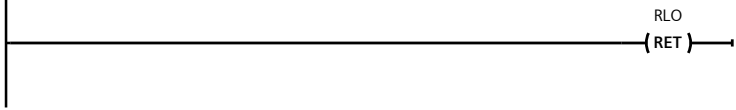
Network 14: Clearing state resets all motion errors



Network 15: Put any error handling here



Network 16:



Program blocks / EM01_Axis01_W

EM01_CM02_ServoAxisObject [FB203]

EM01_CM02_ServoAxisObject Properties

General

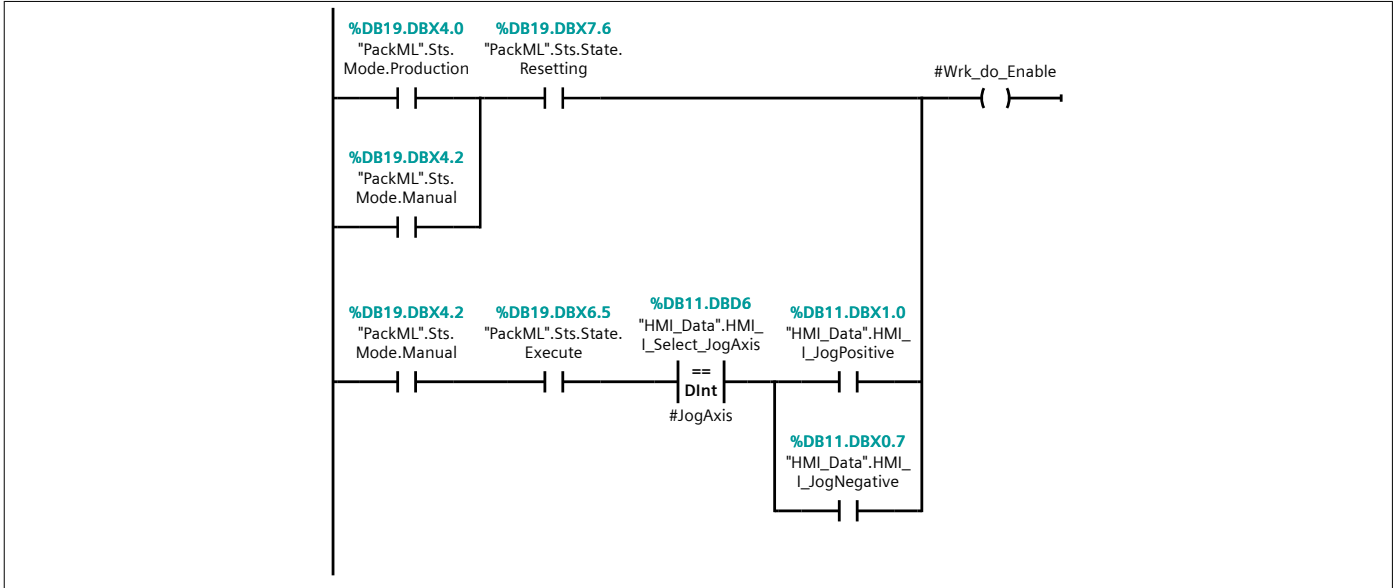
Name	EM01_CM02_ServoAxisObject	Number	203	Type	FB
Language	LAD	Numbering	Manual		

Information

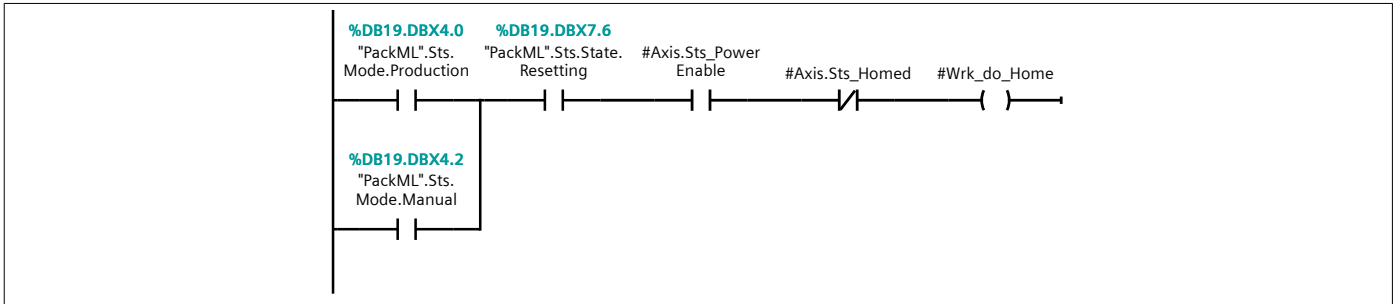
Title	Servo Axis object for X axis	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
JogAxis	DInt	0	Non-retain
Output			
▼ InOut			
Ref_ServoAxis	TO_PositioningAxis		
▼ Static			
Axis	"Axis_ObjectPOS"		
Wrk_DisableDelay	TON_TIME		Non-retain
Wrk_DisableDelay_IN	Bool	false	Non-retain
Wrk_do_Enable	Bool	false	Non-retain
Wrk_do_Home	Bool	false	Non-retain
Wrk_do_Stop	Bool	false	Non-retain
Wrk_do_Disable	Bool	false	Non-retain
Wrk_Temp_Bit	Bool	false	Non-retain
Temp			
Constant			

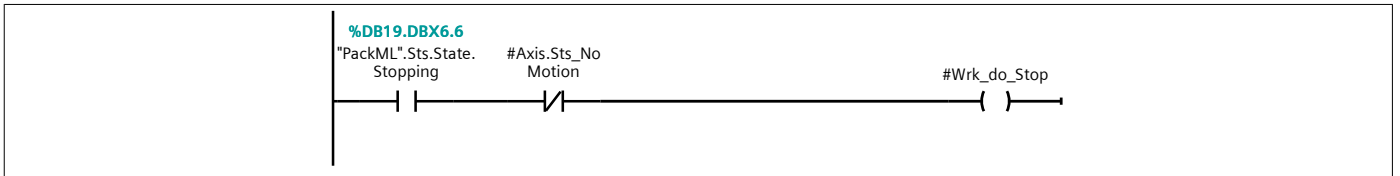
Network 1: Do_Enable logic



Network 2: Do_Home logic



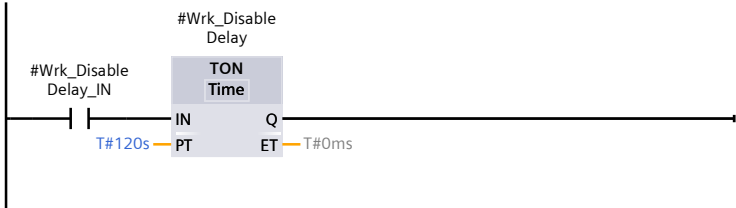
Network 3: Do_Stop logic



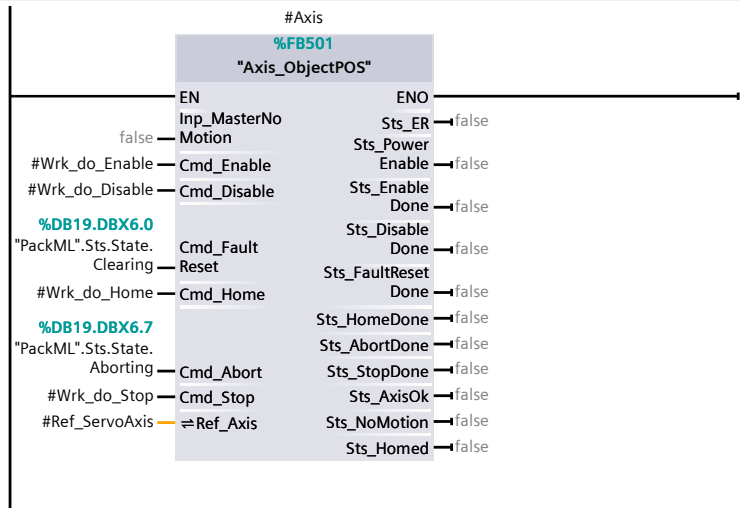
Network 4: Do_Disable logic



Network 5: Disable delay timer for manual mode - if no jogging for 2 min, disable axis



Network 6:



Program blocks / EM01_Axis01_W

EM01_SR20_Initialize [FB212]

EM01_SR20_Initialize Properties

General

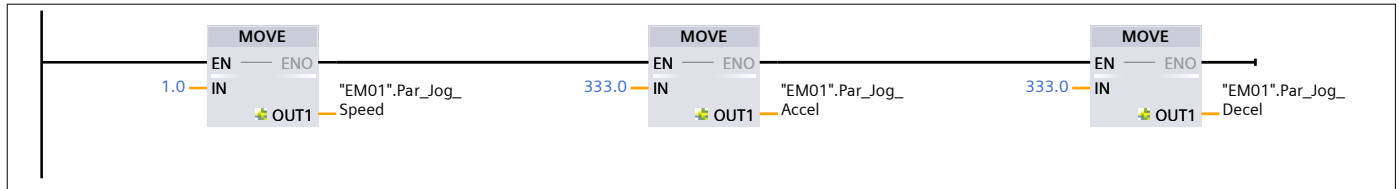
Name	EM01_SR20_Initialize	Number	212	Type	FB
Language	LAD	Numbering	Manual		

Information

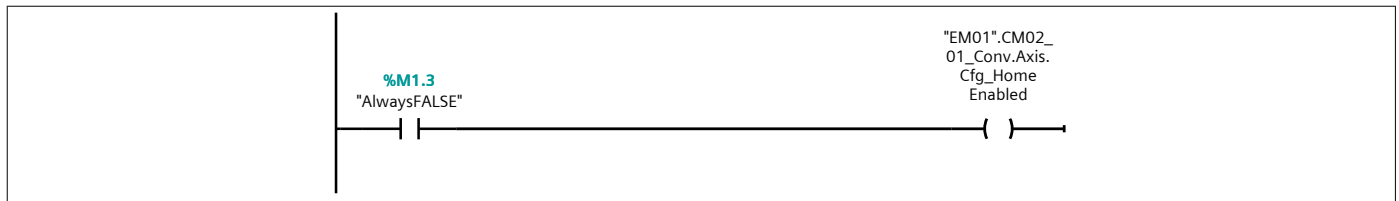
Title	Initialization for EM	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Temp_Dint	DInt	0	Non-retain
Temp			
Constant			

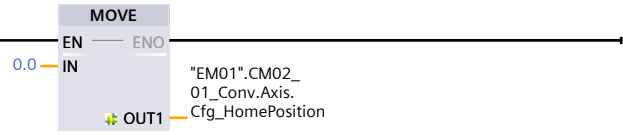
Network 2:



Network 3: Home disabled. So do not have to wait for long resetting and will cut at first registration mark.



Network 4: Home position



"EM01".CM02_
01_Conv.Axis.
Cfg_HomePosition

Program blocks / EM01_Axis01_W

EM01 [DB201]

EM01 Properties

General

Name	EM01	Number	201	Type	DB
Language	DB	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Start value	Retain
Input			
Output			
InOut			
▼ Static			
Cfg_EM_Number	Int	0	False
Par_Jog_Accel	Real	0.0	False
Par_Jog_Decel	Real	0.0	False
Par_Jog_Speed	Real	0.0	False
CM00_EM_Procedure	"EM01_CM00_Procedure"		False
CM01_EMConditions	"EM01_CM01_EMConditions"		False
CM02_01_Conv	"EM01_CM02_ServoAxisObject"		False
CM03_01_AxisJog_Conv	"EM01_CM03_ServoAxisJog"		False
SR03_FaultHandler	"EM01_SR03_FaultHandler"		False
SR20_Initialize	"EM01_SR20_Initialize"		False
Wrk_Temp_Bit	Bool	false	False

Program blocks / EM01_Axis01_W

EM01_CM03_ServoAxisJog [FB204]

EM01_CM03_ServoAxisJog Properties

General

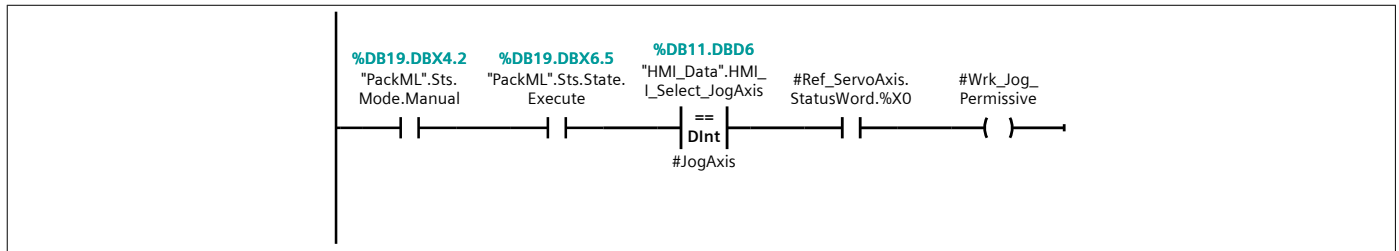
Name	EM01_CM03_ServoAxis-Jog	Number	204	Type	FB
Language	LAD	Numbering	Manual		

Information

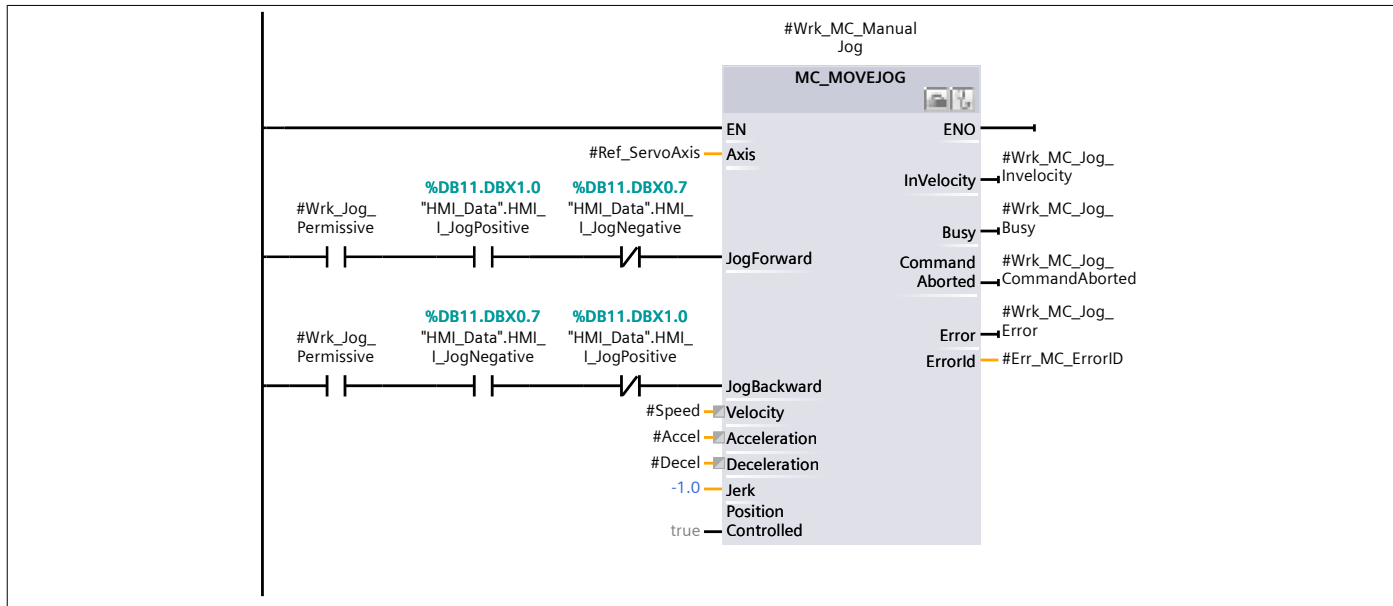
Title	Jog servo axis in manual mode	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
JogAxis	DInt	0	Non-retain
Speed	Real	0.0	Non-retain
Accel	Real	0.0	Non-retain
Decel	Real	0.0	Non-retain
Output			
▼ InOut			
Ref_ServoAxis	TO_PositioningAxis		
▼ Static			
Sts_JogFault	Bool	false	Non-retain
Err_MC_ErrorID	Word	16#0	Non-retain
Wrk_Jog_Permissive	Bool	false	Non-retain
Wrk_MC_ManualJog	MC_MOVEJOG		
Wrk_MC_Jog_Invelocity	Bool	false	Non-retain
Wrk_MC_Jog_Busy	Bool	false	Non-retain
Wrk_MC_Jog_CommandAborted	Bool	false	Non-retain
Wrk_MC_Jog_Error	Bool	false	Non-retain
Temp			
Constant			

Network 1: RELEASE JOG FUNCTION (-> SELECT EM NUMBER VIA HMI)



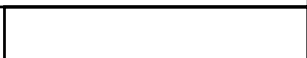
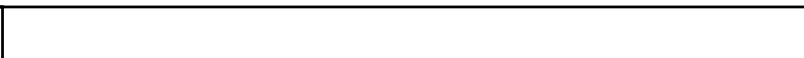
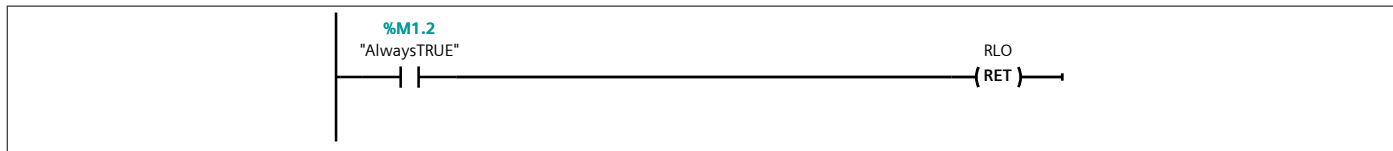
Network 2: JOG SERVO AXIS



Network 3:



Network 4: END: Update the ENO Output. (DO NOT REMOVE. Must be last rung)



Program blocks / EM01_Axis01_W

EM01_CM01_EMConditions [FB202]

EM01_CM01_EMConditions Properties

General

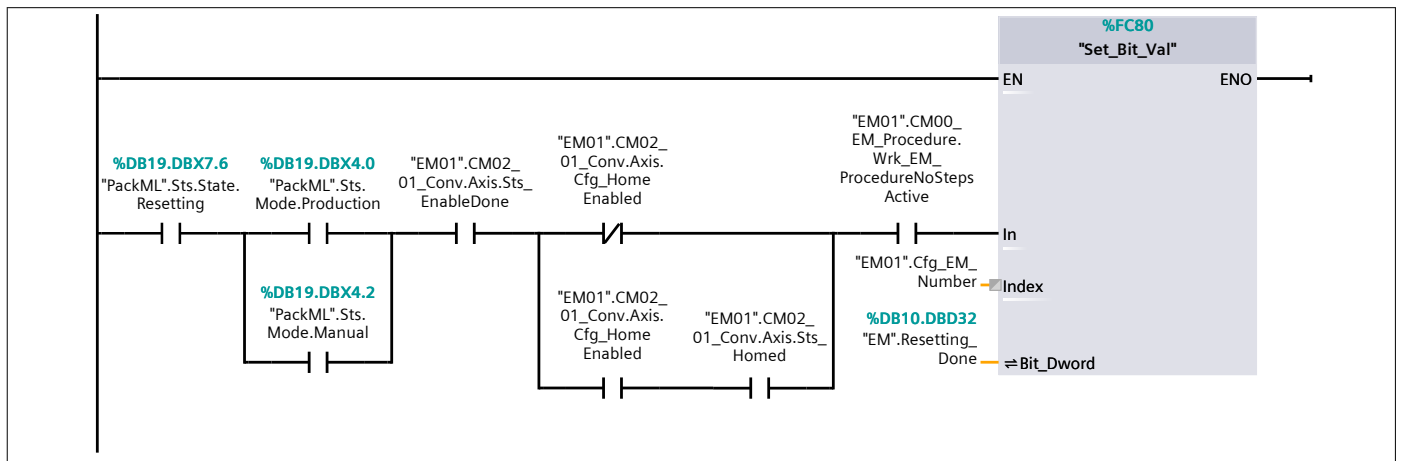
Name	EM01_CM01_EMConditions	Number	202	Type	FB
Language	LAD	Numbering	Manual		

Information

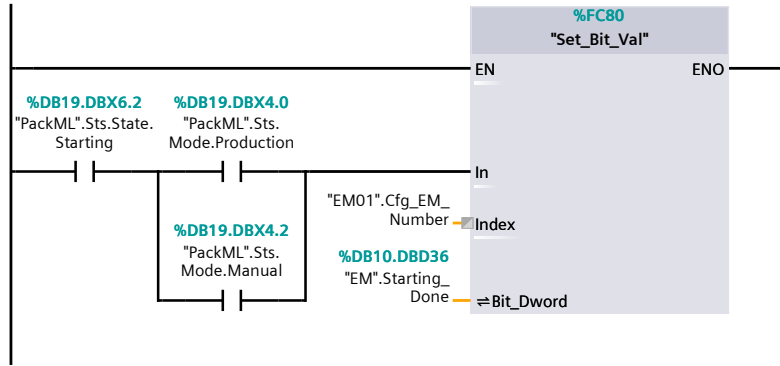
Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Wrk_Temp	DInt	0	Non-retain
Wrk_Push	Bool	true	Non-retain
Temp			
Constant			

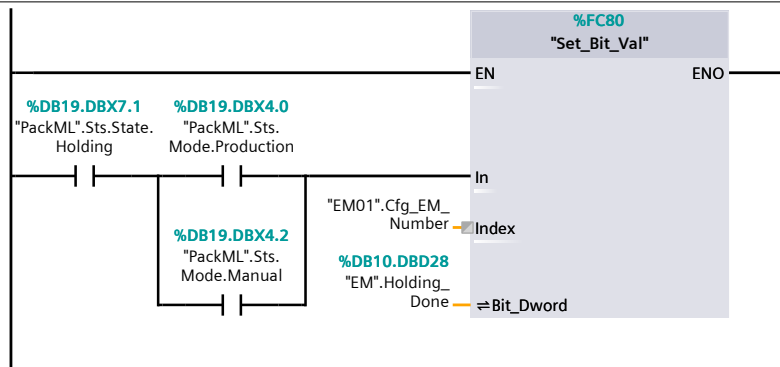
Network 1: Resetting state



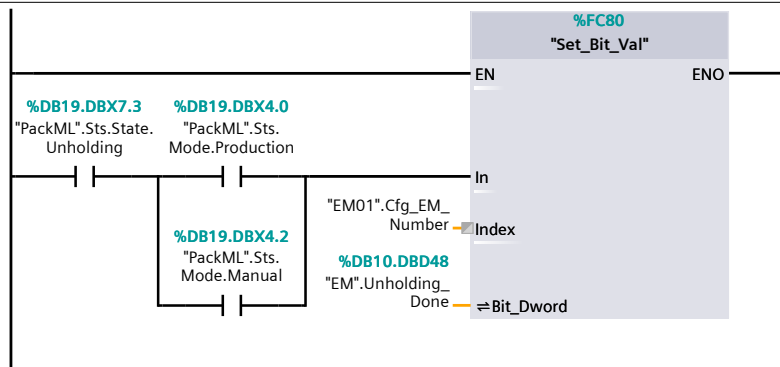
Network 2: Starting state



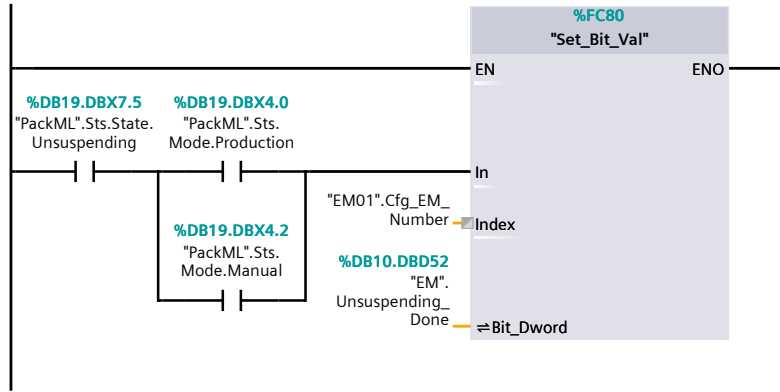
Network 3: Holding state



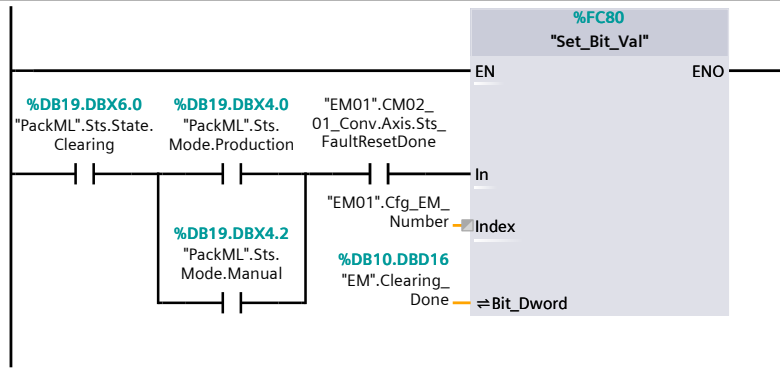
Network 4: Unholding



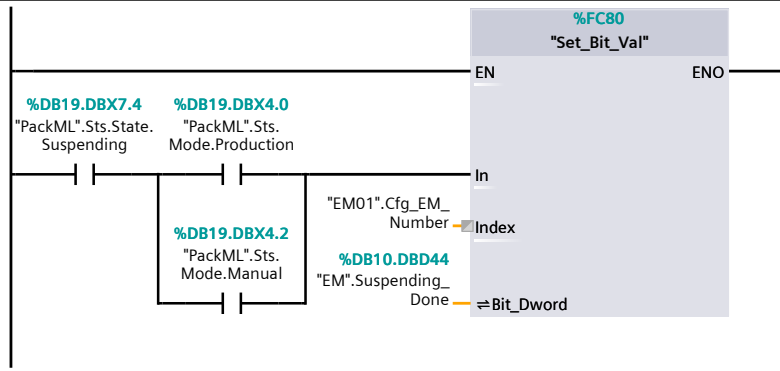
Network 5: Unsuspending



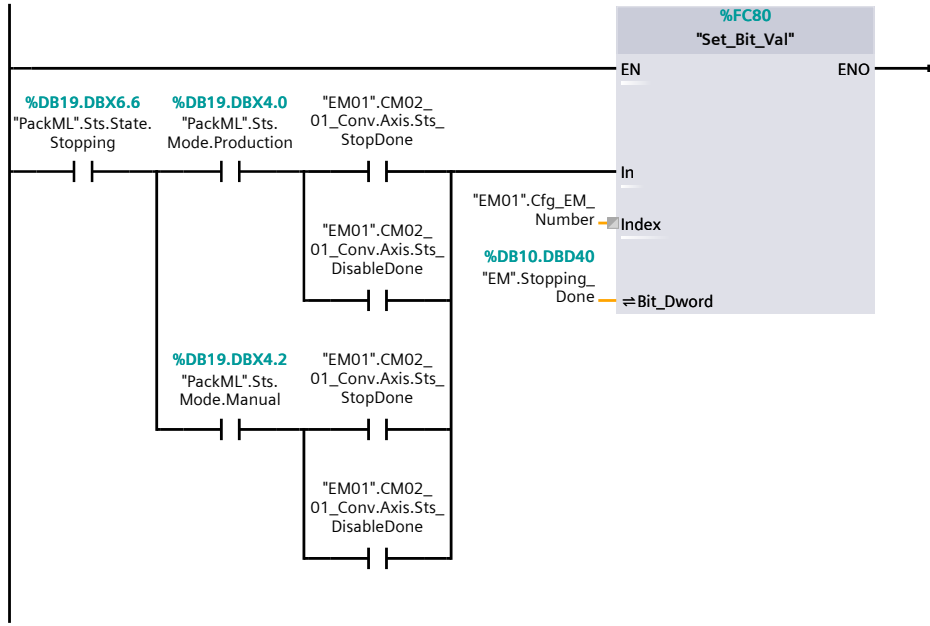
Network 6: Clearing state



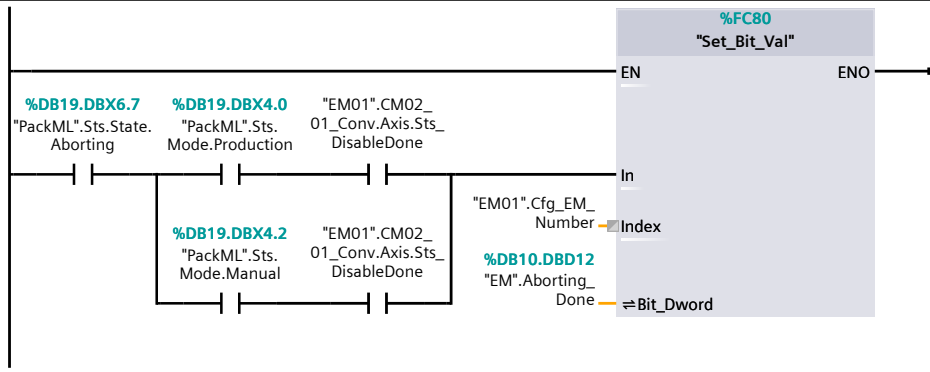
Network 7: Suspending state



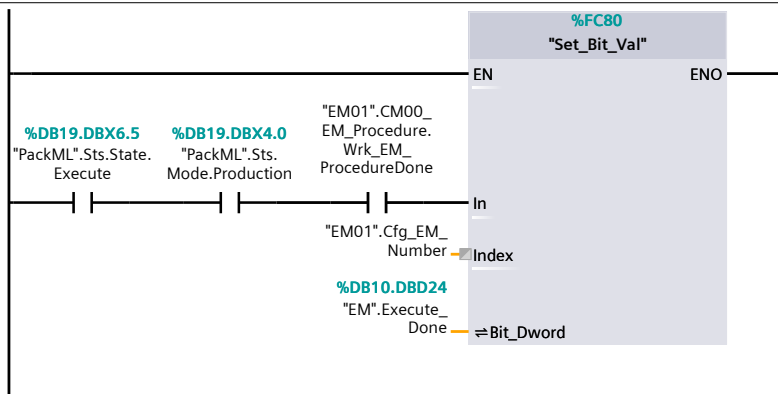
Network 8: Stopping state



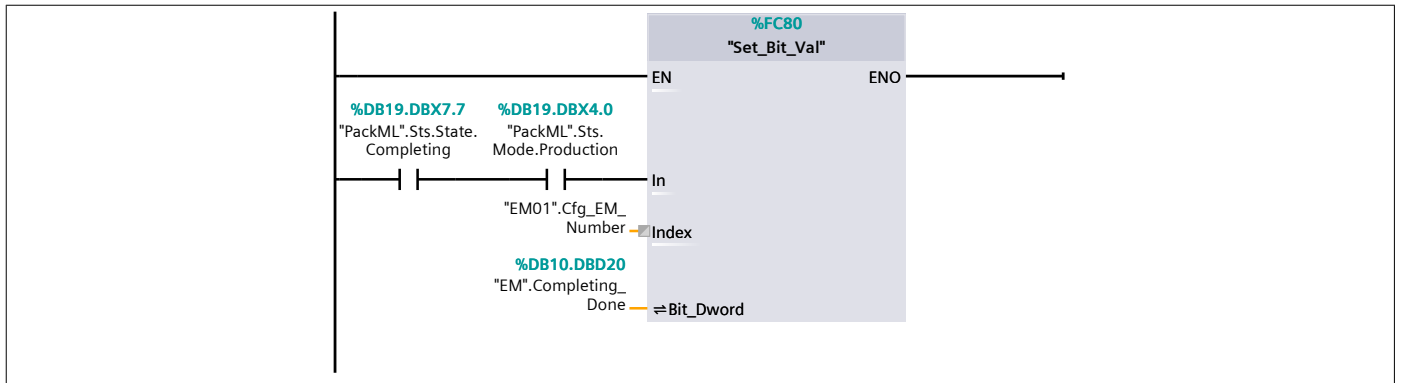
Network 9: Aborting state



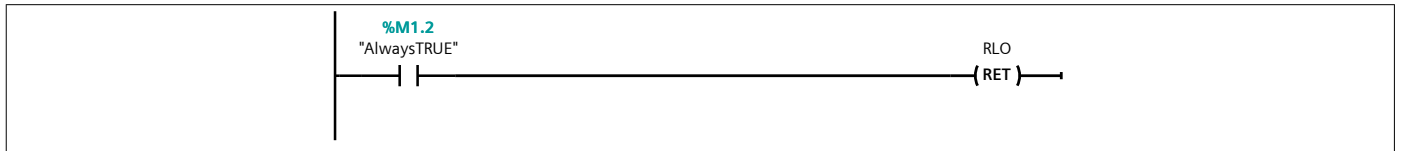
Network 10: Execute state



Network 11: Completing State



Network 12:



Program blocks / EM01_Axis01_W

EM01_SR03_FaultHandler [FB211]

EM01_SR03_FaultHandler Properties

General

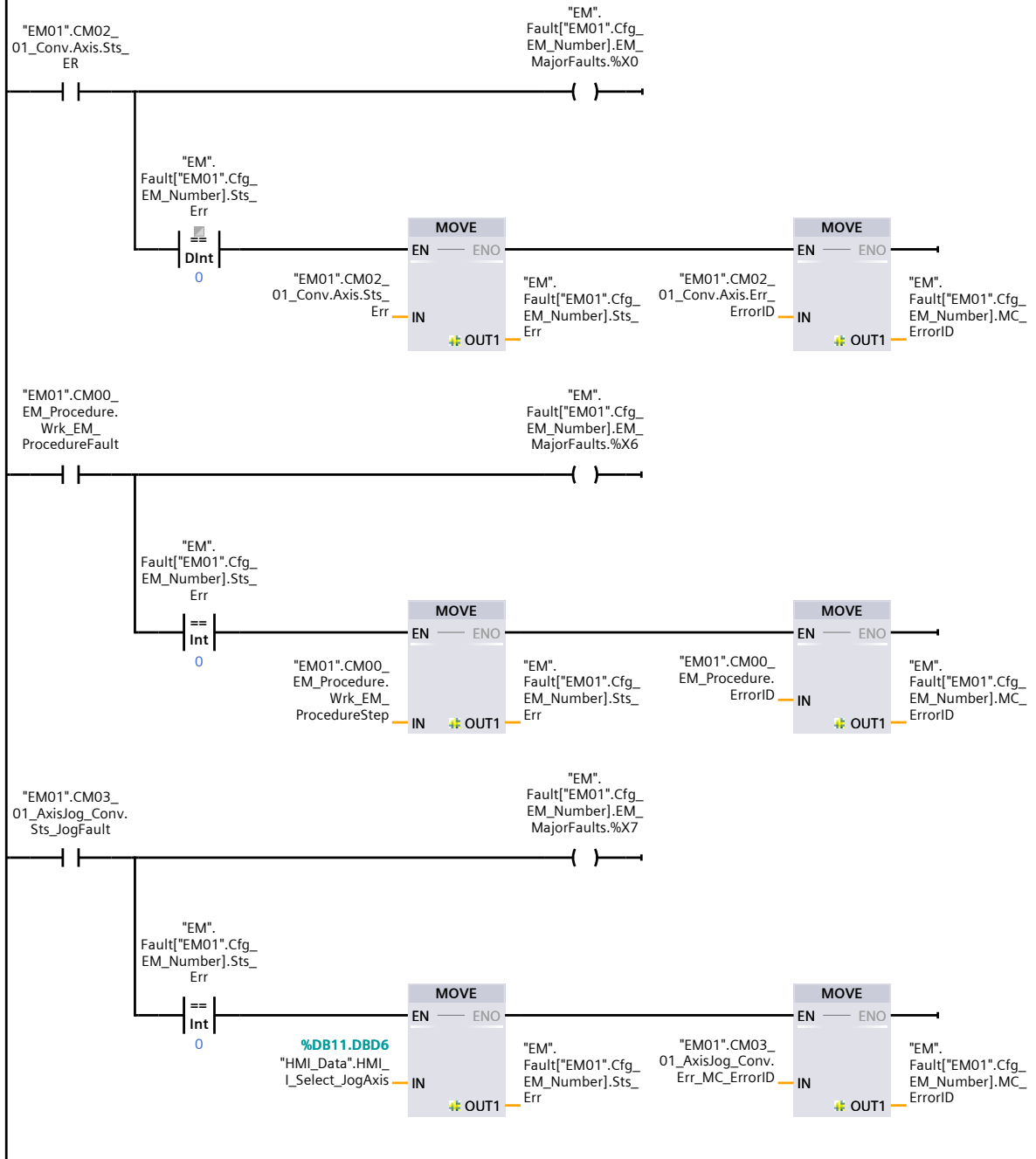
Name	EM01_SR03_FaultHandler	Number	211	Type	FB
Language	LAD	Numbering	Manual		

Information

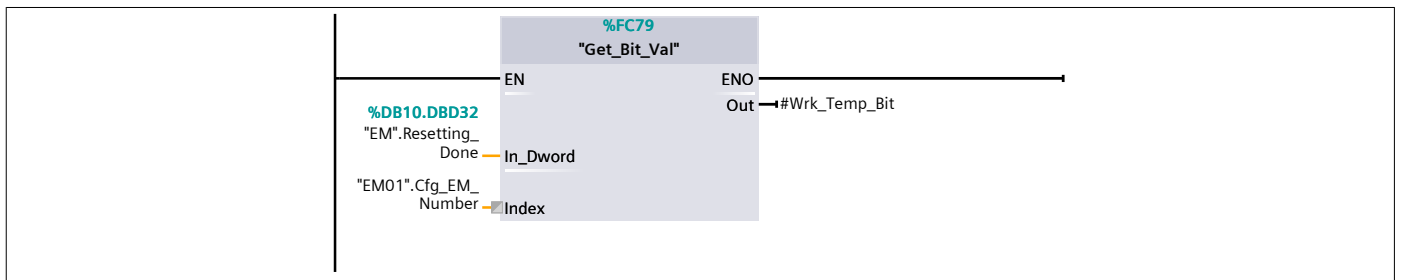
Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Wrk_Temp	DInt	0	Non-retain
Wrk_Temp_Bit	Bool	false	Non-retain
Temp			
Constant			

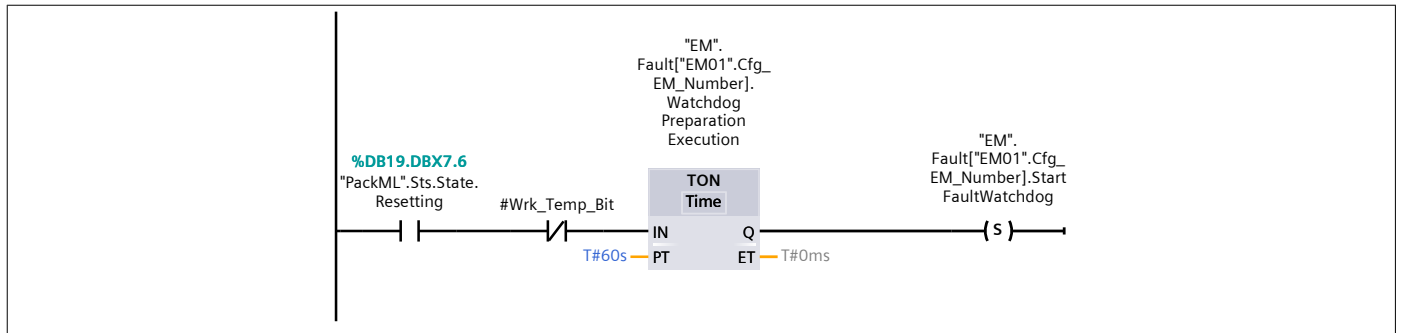
Network 1: Map axis errors, procedure motion errors, and jogging errors into bits of the EM_Major-Faults



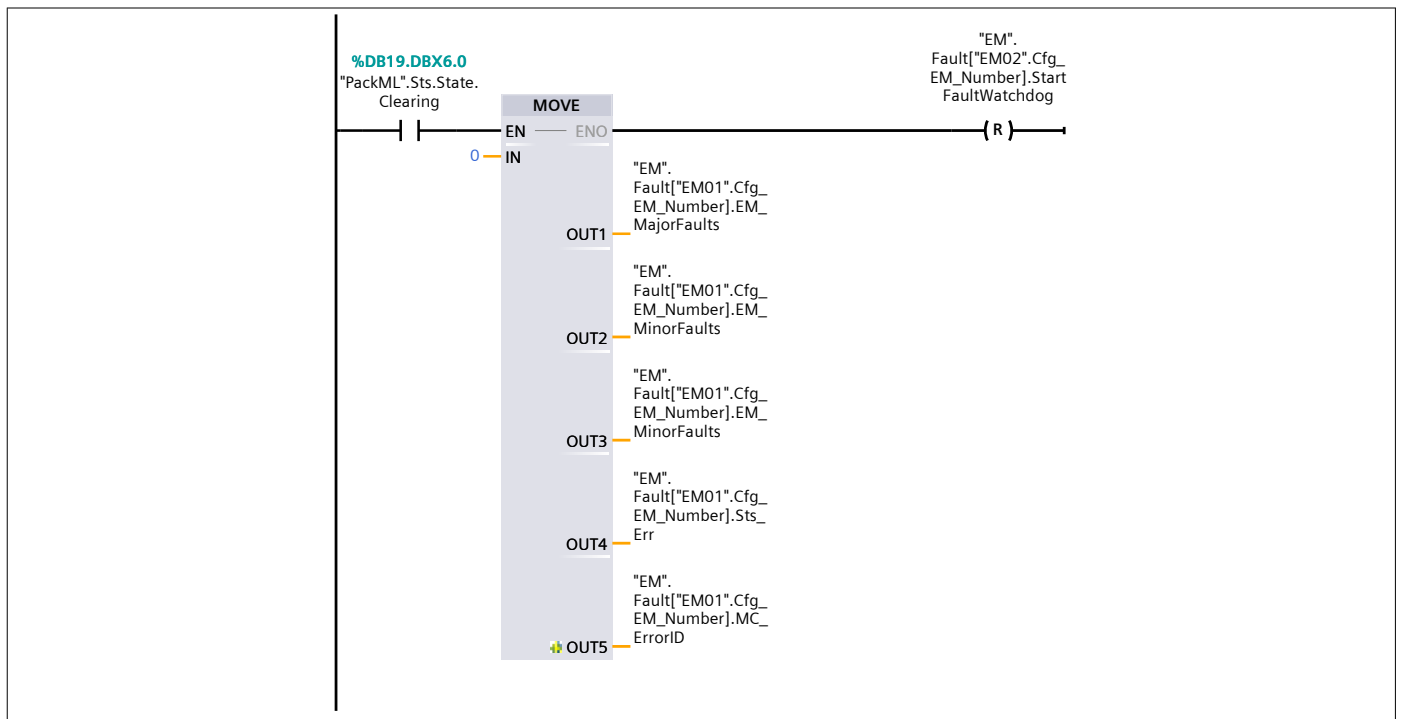
Network 2: Get resetting status for this EM



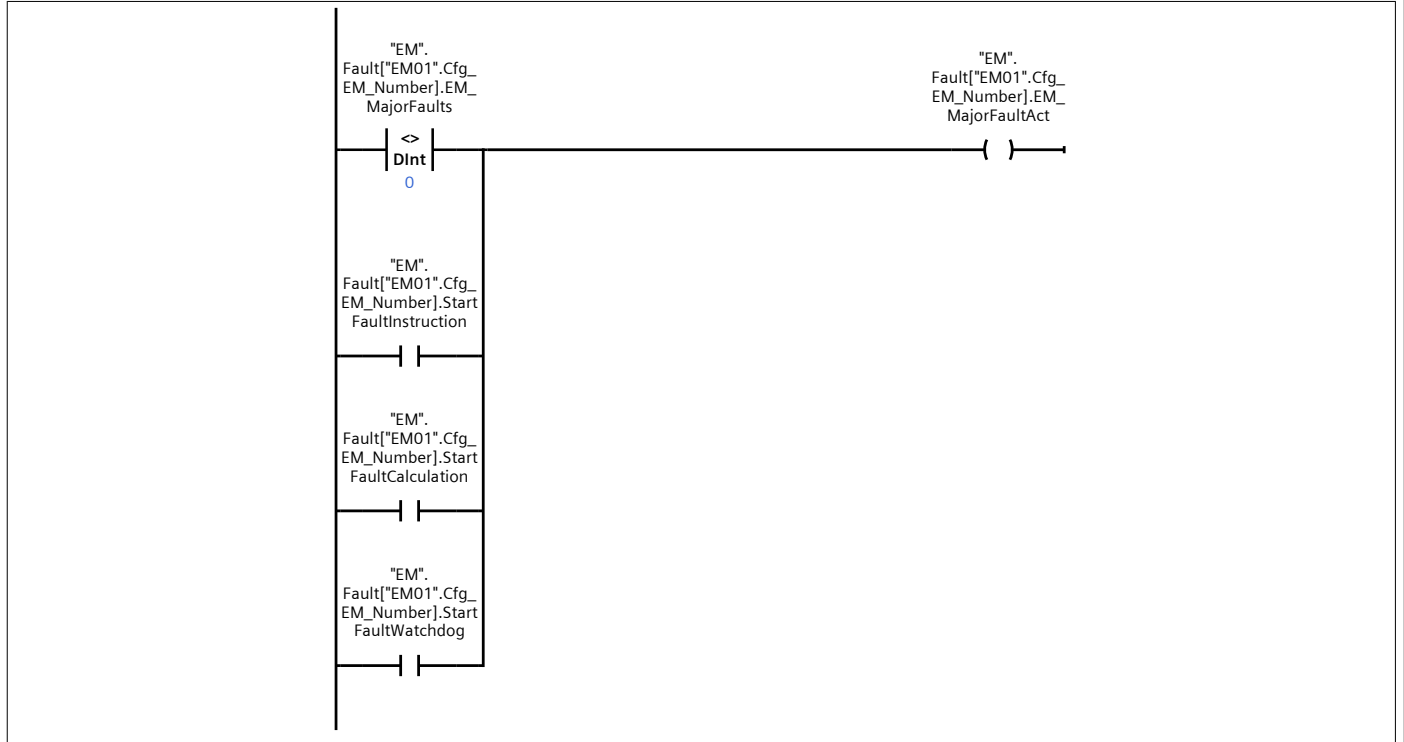
Network 3: Watchdog for resetting - CREATE AN ERROR IF TIMEOUT OCCURS



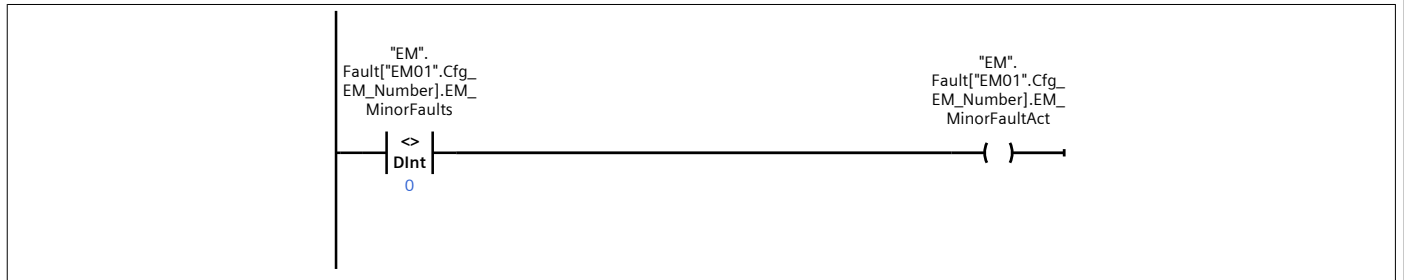
Network 4: Clear watchdog and instruction fault error details



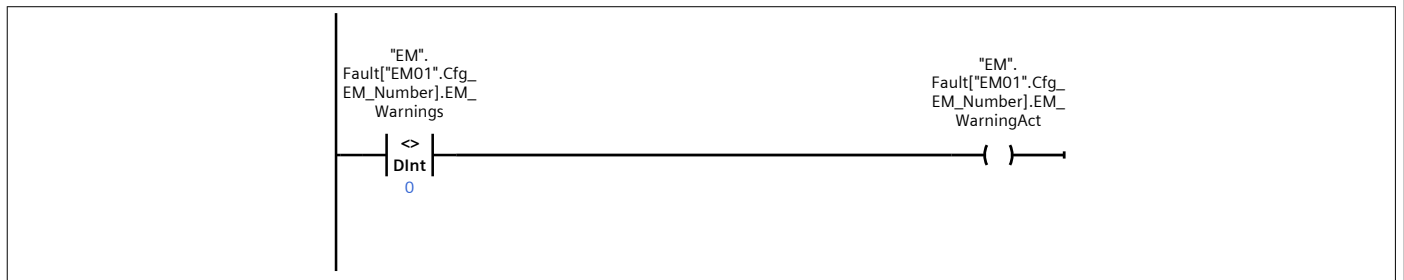
Network 5: Summary of EM faults



Network 6:



Network 7:



Program blocks / LPMLV30

LPMLV30_ConfigureDisabledStates [FC30101]

LPMLV30_ConfigureDisabledStates Properties

General

Name	LPMLV30_ConfigureDisabledStates	Number	30101	Type	FC
Language	SCL	Numbering	Automatic		

Information

Title	LPMLV30_ConfigureDisabledStates	Author	APC_ERLF	Comment	Support: tech.team.motioncontrol@siemens.com Fax: +49 (0) 9131/98-1297
Family	OMAC	Version		User-defined ID	

Name	Data type	Default value
▼ Input		
HoldingDisable	Bool	
HeldDisable	Bool	
UnholdingDisable	Bool	
SuspendingDisable	Bool	
SuspendedDisable	Bool	
UnsuspendingDisable	Bool	
CompletingDisable	Bool	
CompleteDisable	Bool	
▼ Output		
disabledStates	DInt	
InOut		
Temp		
▼ Constant		
DISABLE_ALL_STATES	DWord	16#FFFF_FFFF
▼ Return		
LPMLV30_ConfigureDisabledStates	Void	

```

0001 //
-----
-----
0002 // SIEMENS AG
0003 // (c) Copyright 2015 All Rights Reserved
0004 //-----
-----
0005 // Library: LPMLV30
0006 // Tested with: S7-1200 with FW version V4.1, S7-1500 with FW version V1.7
0007 // Engineering: TIA Portal V13 SP1
0008 // Restrictions: ---
0009 // Requirements: S7-1200 / S7-1500
0010 // Functionality: The function generates a double integer value which represents
the state configuration
0011 // for one unit mode.

```

```
0012 //-----  
-----  
0013 // Change log table:  
0014 // Version Date Expert in charge Changes applied  
0015 // 03.00.01 29.05.2015 RK First released version  
0016 //  
-----  
-----  
0017 // Function: LPMLV30_ConfigureDisabledStates  
0018 //  
-----  
-----  
0019  
0020 #disabledStates := DWORD_TO_DINT(#DISABLE_ALL_STATES);  
0021  
0022 #disabledStates.%X0 := FALSE; // Undefined state is mandatory  
0023 #disabledStates.%X1 := FALSE; // Clearing state is mandatory  
0024 #disabledStates.%X2 := FALSE; // Stopped state is mandatory  
0025 #disabledStates.%X3 := FALSE; // Starting state is mandatory  
0026 #disabledStates.%X4 := FALSE; // Idle state is mandatory  
0027 #disabledStates.%X5 := #SuspendedDisable;  
0028 #disabledStates.%X6 := FALSE; // Execute state is mandatory  
0029 #disabledStates.%X7 := FALSE; // Stopping state is mandatory  
0030 #disabledStates.%X8 := FALSE; // Aborting state is mandatory  
0031 #disabledStates.%X9 := FALSE; // Aborted state is mandatory  
0032 #disabledStates.%X10 := #HoldingDisable;  
0033 #disabledStates.%X11 := #HeldDisable;  
0034 #disabledStates.%X12 := #UnholdingDisable;  
0035 #disabledStates.%X13 := #SuspendingDisable;  
0036 #disabledStates.%X14 := #UnsuspendingDisable;  
0037 #disabledStates.%X15 := FALSE; // Resetting state is mandatory  
0038 #disabledStates.%X16 := #CompletingDisable;  
0039 #disabledStates.%X17 := #CompleteDisable;  
0040  
0041 //  
-----  
-----  
0042 // End of function: LPMLV30_ConfigureDisabledStates  
0043 //  
-----  
-----
```

Program blocks / LPMLV30

LPMLV30_ConfigureDisabledUnitModes [FC30100]

LPMLV30_ConfigureDisabledUnitModes Properties

General

Name	LPMLV30_ConfigureDisabledUnitModes	Number	30100	Type	FC
Language	SCL	Numbering	Automatic		

Information

Title	LPMLV30_ConfigureDisabledUnitModes	Author	APC_ERLF	Comment	Support: tech.team.motioncontrol@siemens.com Fax: +49 (0) 9131/98-1297
Family	OMAC	Version		User-defined ID	

Name	Data type	Default value
▼ Input		
ProductionModeDisable	Bool	
MaintenanceModeDisable	Bool	
UserMode01Disable	Bool	
UserMode02Disable	Bool	
UserMode03Disable	Bool	
UserMode04Disable	Bool	
UserMode05Disable	Bool	
UserMode06Disable	Bool	
UserMode07Disable	Bool	
UserMode08Disable	Bool	
▼ Output		
disabledUnitModes	Array[0.."LPMLV30_MAX_MODES_UPPER_LIM"] of Bool	
InOut		
▼ Temp		
templ	Int	
Constant		
▼ Return		
LPMLV30_ConfigureDisabledUnitModes	Void	

```

0001 //
=====
0002 // SIEMENS AG
0003 // (c) Copyright 2015 All Rights Reserved
0004 //-----
0005 // Library: LPMLV30
0006 // Tested with: S7-1200 with FW version V4.1, S7-1500 with FW version V1.7
0007 // Engineering: TIA Portal V13 SP1

```

```

0008 // Restrictions: ---
0009 // Requirements: S7-1200 / S7-1500
0010 // Functionality: The function generates the unit mode configuration as Array[]
of Bool.
0011 //-----
-----
0012 // Change log table:
0013 // Version Date Expert in charge Changes applied
0014 // 03.00.01 29.05.2015 RK First released version
0015 //
=====
=====
0016 // Function: LPMLV30_ConfigureDisabledUnitModes
0017 //
=====
=====

0018
0019 // Initialize unitMode configuration
0020 FOR #tempI := 0 TO "LPMLV30_MAX_MODES_UPPER_LIM" DO
0021     #disabledUnitModes[#tempI] := TRUE;
0022 END_FOR;
0023
0024 // Set unitMode configuration
0025 #disabledUnitModes["LPMLV30_MODE_INVALID"] := FALSE;
0026 #disabledUnitModes["LPMLV30_MODE_PRODUCTION"] := #ProductionModeDisable;
0027 #disabledUnitModes["LPMLV30_MODE_MAINTENANCE"] := #MaintenanceModeDisable;
0028 #disabledUnitModes["LPMLV30_MODE_MANUAL"] := FALSE;
0029 #disabledUnitModes["LPMLV30_MODE_USER_01"] := #UserMode01Disable;
0030 #disabledUnitModes["LPMLV30_MODE_USER_02"] := #UserMode02Disable;
0031 #disabledUnitModes["LPMLV30_MODE_USER_03"] := #UserMode03Disable;
0032 #disabledUnitModes["LPMLV30_MODE_USER_04"] := #UserMode04Disable;
0033 #disabledUnitModes["LPMLV30_MODE_USER_05"] := #UserMode05Disable;
0034 #disabledUnitModes["LPMLV30_MODE_USER_06"] := #UserMode06Disable;
0035 #disabledUnitModes["LPMLV30_MODE_USER_07"] := #UserMode07Disable;
0036 #disabledUnitModes["LPMLV30_MODE_USER_08"] := #UserMode08Disable;
0037
0038 //
=====
=====
0039 // End of function: LPMLV30_ConfigureDisabledUnitModes
0040 //
=====
=====

```

Program blocks / LPMLV30

LPMLV30_GetUnitModeStateNamesAsString [FC30102]

LPMLV30_GetUnitModeStateNamesAsString Properties

General

Name	LPMLV30_GetUnitModeStateNamesAsString	Number	30102	Type	FC
Language	SCL	Numbering	Automatic		

Information

Title	LPMLV30_GetUnitModeStateNamesAsString	Author	APC_ERLF	Comment	Support: tech.team.motioncontrol@siemens.com Fax: +49 (0) 9131/98-1297
Family	OMAC	Version		User-defined ID	

Name	Data type	Default value
▼ Input		
UnitModeCurrent	DInt	
StateCurrent	DInt	
language	Int	
namesConfiguration	"typeLPMLV30_NamesConfiguration"	
▼ Output		
unitModeCurrentName	String	
stateCurrentName	String	
InOut		
▼ Temp		
tempUnitModeCurrent	DInt	
tempStateCurrent	DInt	
tempLanguage	Int	
Constant		
▼ Return		
LPMLV30_GetUnitModeStateNamesAsString	Void	

```

0001 //
=====
0002 // SIEMENS AG
0003 // (c) Copyright 2015 All Rights Reserved
0004 //-----
-----
0005 // Library: LPMLV30
0006 // Tested with: S7-1200 with FW version V4.1, S7-1500 with FW version V1.7
0007 // Engineering: TIA Portal V13 SP1
0008 // Restrictions: ---
0009 // Requirements: S7-1200 / S7-1500
0010 // Functionality: Gets the names for current unit mode and state in one language
0011 // corresponding to the names configuration
0012 //-----
-----

```

```
0013 // Change log table:
0014 // Version Date Expert in charge Changes applied
0015 // 03.00.01 29.05.2015 RK First released version
0016 //
=====
0017 // Function: LPMLV30_GetUnitModeStateNamesAsString
0018 //
=====

0019
0020 #tempUnitModeCurrent := #UnitModeCurrent;
0021 #tempStateCurrent    := #StateCurrent;
0022 #tempLanguage        := #language;
0023
0024 IF #tempUnitModeCurrent < 0 OR #tempUnitModeCurrent > "LPMLV30_MODES_UPPER_LIM" THEN
0025     #tempUnitModeCurrent := "LPMLV30_MODE_INVALID";
0026 END_IF;
0027 IF #tempStateCurrent < 0 OR #tempStateCurrent > "LPMLV30_STATES_UPPER_LIM" THEN
0028     #tempStateCurrent := "LPMLV30_STATE_UNDEFINED";
0029 END_IF;
0030 IF #tempLanguage < 0 OR #tempLanguage > "LPMLV30_LANGUAGES_UPPER_LIM" THEN
0031     #tempLanguage := "LPMLV30_LANGUAGE_1";
0032 END_IF;
0033
0034 #unitModeCurrentName := #namesConfiguration.unitModesNames[#tempLanguage, #tempUnitModeCurrent];
0035 #stateCurrentName    := #namesConfiguration.statesNames[#tempLanguage, #tempStateCurrent];
0036
0037 //
=====
0038 // End of function: LPMLV30_GetUnitModeStateNamesAsString
0039 //
=====
```

Program blocks / LPMLV30

LPMLV30_UnitModeStateManager [FB30100]

LPMLV30_UnitModeStateManager Properties

General

Name	LPMLV30_UnitModeState-Manager	Number	30100	Type	FB
Language	SCL	Numbering	Automatic		

Information

Title	LPMLV30_UnitModeState-Manager	Author	APC_ERLF	Comment	Support: tech.team.motioncontrol@siemens.com Fax: +49 (0) 9131/98-1297
Family	OMAC	Version		User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
UnitMode	DInt	"LPMLV30_MODE_INVALID"	Non-retain
UnitModeChangeRequest	Bool	false	Non-retain
CntrlCmd	DInt	"LPMLV30_CMD_UNDEFINED"	Non-retain
CmdChangeRequest	Bool	false	Non-retain
enableBooleanInterface	Bool	false	Non-retain
ProductionModeRequest	Bool	false	Non-retain
MaintenanceModeRequest	Bool	false	Non-retain
ManualModeRequest	Bool	false	Non-retain
UserMode01Request	Bool	false	Non-retain
UserMode02Request	Bool	false	Non-retain
UserMode03Request	Bool	false	Non-retain
UserMode04Request	Bool	false	Non-retain
UserMode05Request	Bool	false	Non-retain
UserMode06Request	Bool	false	Non-retain
UserMode07Request	Bool	false	Non-retain
UserMode08Request	Bool	false	Non-retain
ResetCmdRequest	Bool	false	Non-retain
StartCmdRequest	Bool	false	Non-retain
StopCmdRequest	Bool	false	Non-retain
HoldCmdRequest	Bool	false	Non-retain
UnholdCmdRequest	Bool	false	Non-retain
SuspendCmdRequest	Bool	false	Non-retain
UnsuspendCmdRequest	Bool	false	Non-retain
AbortCmdRequest	Bool	false	Non-retain
ClearCmdRequest	Bool	false	Non-retain
CompleteCmdRequest	Bool	false	Non-retain
SC	Bool	false	Non-retain
configuration	"typeLPMLV30_Configuration"		Non-retain
▼ Output			
UnitModeCurrent	DInt	"LPMLV30_MODE_MANUAL"	Non-retain

Name	Data type	Default value	Retain
UnitModeRequested	DInt	0	Non-retain
UnitModeChangeInProgress	Bool	false	Non-retain
StateCurrent	DInt	"LPMLV30_STATE_STOPPED"	Non-retain
StateRequested	DInt	0	Non-retain
StateChangeInProgress	Bool	false	Non-retain
ProductionModeActive	Bool	false	Non-retain
MaintenanceModeActive	Bool	false	Non-retain
ManualModeActive	Bool	false	Non-retain
UserMode01Active	Bool	false	Non-retain
UserMode02Active	Bool	false	Non-retain
UserMode03Active	Bool	false	Non-retain
UserMode04Active	Bool	false	Non-retain
UserMode05Active	Bool	false	Non-retain
UserMode06Active	Bool	false	Non-retain
UserMode07Active	Bool	false	Non-retain
UserMode08Active	Bool	false	Non-retain
ClearingStateActive	Bool	false	Non-retain
StoppedStateActive	Bool	false	Non-retain
StartingStateActive	Bool	false	Non-retain
IdleStateActive	Bool	false	Non-retain
SuspendedStateActive	Bool	false	Non-retain
ExecuteStateActive	Bool	false	Non-retain
StoppingStateActive	Bool	false	Non-retain
AbortingStateActive	Bool	false	Non-retain
AbortedStateActive	Bool	false	Non-retain
HoldingStateActive	Bool	false	Non-retain
HeldStateActive	Bool	false	Non-retain
UnholdingStateActive	Bool	false	Non-retain
SuspendingStateActive	Bool	false	Non-retain
UnsuspendingStateActive	Bool	false	Non-retain
ResettingStateActive	Bool	false	Non-retain
CompletingStateActive	Bool	false	Non-retain
CompleteStateActive	Bool	false	Non-retain
StatesDisabled	DInt	0	Non-retain
diagnostics	"typeLPMLV30_Diagnostics"		Non-retain
InOut			
▼ Static			
statConfiguration	"typeLPMLV30_Configuration"		Non-retain
statUnitModeChangeRequest	Bool	false	Non-retain
statCmdChangeRequest	Bool	false	Non-retain
statFirstCycle	Bool	true	Non-retain
statSCOld	Bool	false	Non-retain
statUnitMode	DInt	"LPMLV30_MODE_MANUAL"	Non-retain
statUnitModeOld	DInt	"LPMLV30_MODE_MANUAL"	Non-retain
statUnitModeCurrentInternal	DInt	"LPMLV30_MODE_MANUAL"	Non-retain
statUnitModeCurrentInternalOld	DInt	"LPMLV30_MODE_INVALID"	Non-retain
statStateCurrentOld	DInt	"LPMLV30_STATE_STOPPED"	Non-retain

Name	Data type	Default value	Retain
statStateInternal	DInt	"LPMLV30_STATE_STOPPED"	Non-retain
statStateInternalOld	DInt	"LPMLV30_STATE_UNDEFINED"	Non-retain
statCntrlCmdOld	DInt	"LPMLV30_CMD_UNDEFINED"	Non-retain
statStatesInCurrentMode	Struct		Non-retain
statBooleanInterfaceModeOld	DWord	16#0	Non-retain
statBooleanInterfaceCmdOld	Word	16#0	Non-retain
statCmdChangeRequestOld	Bool	false	Non-retain
statUnitModeChangeRequestOld	Bool	false	Non-retain
▼ Temp			
tempCntrlCmd	DInt		
tempBooleanInterfaceMode	DWord		
tempBooleanInterfaceCmd	Word		
tempSCInternal	Bool		
tempRetVal	Int		
tempDiagnosticsBufferIndex	Int		
tempDiagnosticsBufferIndexSC	Int		
tempDiagnosticsBufferIndexCmd	Int		
▼ Constant			
MSG_NO_MESSAGE	Byte	16#00	
MSG_MODE_CHANGED_SUCCESSFULLY	Byte	16#01	
MSG_STATE_CHANGED_SUCCESSFULLY	Byte	16#02	
MSG_MODE_ALREADY_ACTIVE	Byte	16#03	
MSG_MODE_NOT_DEFINED	Byte	16#80	
MSG_CMD_NOT_DEFINED	Byte	16#81	
MSG_REQ_MODE_NOT_CONFIGURED	Byte	16#82	
MSG_MODE_TRANSITION_NOT_ALLOWED	Byte	16#83	
MSG_CMD_NOT_ALLOWED	Byte	16#84	
MSG_SC_NOT_ALLOWED	Byte	16#85	
MSG_STATE_CONFIG_FORCED	Byte	16#86	

```

0001 //
-----
-----
0002 // SIEMENS AG
0003 // (c) Copyright 2015 All Rights Reserved
0004 //-----
-----
0005 // Library: LPMLV30
0006 // Tested with: S7-1200 with FW version V4.1, S7-1500 with FW version V1.7
0007 // Engineering: TIA Portal V13 SP1
0008 // Restrictions: ---
0009 // Requirements: S7-1200 / S7-1500
0010 // Functionality: Management of unit modes and states according to ISA
TR88.00.02 - June 4, 2014
0011 //-----
-----
0012 // Change log table:
0013 // Version Date Expert in charge Changes applied

```

```

0014 // 03.00.01 29.05.2015 RK First released version
0015 //
=====
0016 // Function block: LPMLV30_UnitModeStateManager
0017 //
=====

0018
0019 IF NOT #statCmdChangeRequest AND #statCmdChangeRequestOld THEN // Falling edge
    CmdChangeRequest
0020     #statCntrlCmdOld := "LPMLV30_CMD_UNDEFINED";
0021 END_IF;
0022 #statCmdChangeRequestOld := #statCmdChangeRequest;
0023
0024 IF NOT #statUnitModeChangeRequest AND #statUnitModeChangeRequestOld THEN // Fall-
    ing edge UnitModeChangeRequest
0025     #statUnitModeOld := "LPMLV30_MODE_INVALID";
0026 END_IF;
0027 #statUnitModeChangeRequestOld := #statUnitModeChangeRequest;
0028
0029 #tempDiagnosticsBufferIndex := #diagnostics.bufferIndex; // Work with tempora-
    ry variable during block execution
0030 #tempDiagnosticsBufferIndexSC := -1;
0031 #tempDiagnosticsBufferIndexCmd := -1;
0032
0033 IF #statFirstCycle THEN // Initialize values in first cycle
0034     #tempDiagnosticsBufferIndex := -1;
0035
0036     #statConfiguration := #configuration; // Copy configuration from input
0037     #statConfiguration.disabledUnitModes["LPMLV30_MODE_MANUAL"] := FALSE; // Unit
    mode Manual is mandatory
0038
0039 // Initialize unit mode
0040 #statUnitMode := "LPMLV30_MODE_MANUAL";
0041 #statUnitModeOld := "LPMLV30_MODE_MANUAL";
0042 #UnitModeCurrent := "LPMLV30_MODE_MANUAL";
0043 #statUnitModeCurrentInternal := "LPMLV30_MODE_MANUAL";
0044 #statUnitModeCurrentInternalOld := "LPMLV30_MODE_INVALID";
0045 #statBooleanInterfaceModeOld := 0;
0046
0047 // Initialize state
0048 #StateCurrent := "LPMLV30_STATE_STOPPED";
0049 #StateRequested := "LPMLV30_STATE_STOPPED";
0050 #statStateInternal := "LPMLV30_STATE_STOPPED";
0051 #statStateInternalOld := "LPMLV30_STATE_UNDEFINED";
0052 #statStateCurrentOld := "LPMLV30_STATE_STOPPED";
0053
0054 // Initialize control command
0055 #tempCntrlCmd := "LPMLV30_CMD_UNDEFINED";
0056 #statCntrlCmdOld := "LPMLV30_CMD_UNDEFINED";
0057 #statBooleanInterfaceCmdOld := 0;
0058 END_IF; // End: Initialize values in first cycle
0059

```

```

0060 //-----
0061 // Evaluation of input parameters -> UnitMode, CntrlCmd
0062 //-----
0063 IF NOT #statFirstCycle THEN // As from second cycle
0064   IF NOT #enableBooleanInterface THEN // Set unit mode and control command ac-
cording to input parameters
0065     #statUnitMode := #UnitMode;
0066     #tempCntrlCmd := #CntrlCmd;
0067
0068     // Set change requests from input
0069     #statUnitModeChangeRequest := #UnitModeChangeRequest;
0070     #statCmdChangeRequest      := #CmdChangeRequest;
0071   ELSE // Read boolean interface inputs
0072     #statUnitModeChangeRequest := TRUE; // Change request static TRUE
0073     #statUnitMode              := "LPMLV30_MODE_INVALID";
0074     #tempBooleanInterfaceMode  := 0;
0075
0076     // Collect unit mode from boolean inputs
0077     #tempBooleanInterfaceMode.%X1 := #ProductionModeRequest;
0078     #tempBooleanInterfaceMode.%X2 := #MaintenanceModeRequest;
0079     #tempBooleanInterfaceMode.%X3 := #ManualModeRequest;
0080     #tempBooleanInterfaceMode.%X4 := #UserMode01Request;
0081     #tempBooleanInterfaceMode.%X5 := #UserMode02Request;
0082     #tempBooleanInterfaceMode.%X6 := #UserMode03Request;
0083     #tempBooleanInterfaceMode.%X7 := #UserMode04Request;
0084     #tempBooleanInterfaceMode.%X8 := #UserMode05Request;
0085     #tempBooleanInterfaceMode.%X9 := #UserMode06Request;
0086     #tempBooleanInterfaceMode.%X10 := #UserMode07Request;
0087     #tempBooleanInterfaceMode.%X11 := #UserMode08Request;
0088     // #tempBooleanInterfaceMode.%X12 := #UserMode09Request;
0089     // #tempBooleanInterfaceMode.%X13 := #UserMode10Request;
0090     // #tempBooleanInterfaceMode.%X14 := #UserMode11Request;
0091     // #tempBooleanInterfaceMode.%X15 := #UserMode12Request;
0092     // #tempBooleanInterfaceMode.%X16 := #UserMode13Request;
0093     // #tempBooleanInterfaceMode.%X17 := #UserMode14Request;
0094     // #tempBooleanInterfaceMode.%X18 := #UserMode15Request;
0095     // #tempBooleanInterfaceMode.%X19 := #UserMode16Request;
0096     // #tempBooleanInterfaceMode.%X20 := #UserMode17Request;
0097     // #tempBooleanInterfaceMode.%X21 := #UserMode18Request;
0098     // #tempBooleanInterfaceMode.%X22 := #UserMode19Request;
0099     // #tempBooleanInterfaceMode.%X23 := #UserMode20Request;
0100     // #tempBooleanInterfaceMode.%X24 := #UserMode21Request;
0101     // #tempBooleanInterfaceMode.%X25 := #UserMode22Request;
0102     // #tempBooleanInterfaceMode.%X26 := #UserMode23Request;
0103     // #tempBooleanInterfaceMode.%X27 := #UserMode24Request;
0104     // #tempBooleanInterfaceMode.%X28 := #UserMode25Request;
0105     // #tempBooleanInterfaceMode.%X29 := #UserMode26Request;
0106     // #tempBooleanInterfaceMode.%X30 := #UserMode27Request;
0107     // #tempBooleanInterfaceMode.%X31 := #UserMode28Request;
0108
0109     // Selection indicator: Rising edge at input, Priority (in case of multiple
selection): Manual, Maintenance, User Mode 01..28, Production

```

```

0110 IF #tempBooleanInterfaceMode.%X3 AND NOT #statBooleanInterfaceMo-
deOld.%X3 THEN
0111 #statUnitMode := "LPMLV30_MODE_MANUAL";
0112 ELSIF #tempBooleanInterfaceMode.%X2 AND NOT #statBooleanInterfaceMo-
deOld.%X2 THEN
0113 #statUnitMode := "LPMLV30_MODE_MAINTENANCE";
0114 ELSIF #tempBooleanInterfaceMode.%X4 AND NOT #statBooleanInterfaceMo-
deOld.%X4 THEN
0115 #statUnitMode := "LPMLV30_MODE_USER_01";
0116 ELSIF #tempBooleanInterfaceMode.%X5 AND NOT #statBooleanInterfaceMo-
deOld.%X5 THEN
0117 #statUnitMode := "LPMLV30_MODE_USER_02";
0118 ELSIF #tempBooleanInterfaceMode.%X6 AND NOT #statBooleanInterfaceMo-
deOld.%X6 THEN
0119 #statUnitMode := "LPMLV30_MODE_USER_03";
0120 ELSIF #tempBooleanInterfaceMode.%X7 AND NOT #statBooleanInterfaceMo-
deOld.%X7 THEN
0121 #statUnitMode := "LPMLV30_MODE_USER_04";
0122 ELSIF #tempBooleanInterfaceMode.%X8 AND NOT #statBooleanInterfaceMo-
deOld.%X8 THEN
0123 #statUnitMode := "LPMLV30_MODE_USER_05";
0124 ELSIF #tempBooleanInterfaceMode.%X9 AND NOT #statBooleanInterfaceMo-
deOld.%X9 THEN
0125 #statUnitMode := "LPMLV30_MODE_USER_06";
0126 ELSIF #tempBooleanInterfaceMode.%X10 AND NOT #statBooleanInterfaceMo-
deOld.%X10 THEN
0127 #statUnitMode := "LPMLV30_MODE_USER_07";
0128 ELSIF #tempBooleanInterfaceMode.%X11 AND NOT #statBooleanInterfaceMo-
deOld.%X11 THEN
0129 #statUnitMode := "LPMLV30_MODE_USER_08";
0130 ELSIF #tempBooleanInterfaceMode.%X12 AND NOT #statBooleanInterfaceMo-
deOld.%X12 THEN
0131 #statUnitMode := "LPMLV30_MODE_USER_09";
0132 ELSIF #tempBooleanInterfaceMode.%X13 AND NOT #statBooleanInterfaceMo-
deOld.%X13 THEN
0133 #statUnitMode := "LPMLV30_MODE_USER_10";
0134 ELSIF #tempBooleanInterfaceMode.%X14 AND NOT #statBooleanInterfaceMo-
deOld.%X14 THEN
0135 #statUnitMode := "LPMLV30_MODE_USER_11";
0136 ELSIF #tempBooleanInterfaceMode.%X15 AND NOT #statBooleanInterfaceMo-
deOld.%X15 THEN
0137 #statUnitMode := "LPMLV30_MODE_USER_12";
0138 ELSIF #tempBooleanInterfaceMode.%X16 AND NOT #statBooleanInterfaceMo-
deOld.%X16 THEN
0139 #statUnitMode := "LPMLV30_MODE_USER_13";
0140 ELSIF #tempBooleanInterfaceMode.%X17 AND NOT #statBooleanInterfaceMo-
deOld.%X17 THEN
0141 #statUnitMode := "LPMLV30_MODE_USER_14";
0142 ELSIF #tempBooleanInterfaceMode.%X18 AND NOT #statBooleanInterfaceMo-
deOld.%X18 THEN
0143 #statUnitMode := "LPMLV30_MODE_USER_15";
0144 ELSIF #tempBooleanInterfaceMode.%X19 AND NOT #statBooleanInterfaceMo-
deOld.%X19 THEN
0145 #statUnitMode := "LPMLV30_MODE_USER_16";

```

```

0146  ELSIF #tempBooleanInterfaceMode.%X20 AND NOT #statBooleanInterfaceMo-
deOld.%X20 THEN
0147      #statUnitMode := "LPMLV30_MODE_USER_17";
0148  ELSIF #tempBooleanInterfaceMode.%X21 AND NOT #statBooleanInterfaceMo-
deOld.%X21 THEN
0149      #statUnitMode := "LPMLV30_MODE_USER_18";
0150  ELSIF #tempBooleanInterfaceMode.%X22 AND NOT #statBooleanInterfaceMo-
deOld.%X22 THEN
0151      #statUnitMode := "LPMLV30_MODE_USER_19";
0152  ELSIF #tempBooleanInterfaceMode.%X23 AND NOT #statBooleanInterfaceMo-
deOld.%X23 THEN
0153      #statUnitMode := "LPMLV30_MODE_USER_20";
0154  ELSIF #tempBooleanInterfaceMode.%X24 AND NOT #statBooleanInterfaceMo-
deOld.%X24 THEN
0155      #statUnitMode := "LPMLV30_MODE_USER_21";
0156  ELSIF #tempBooleanInterfaceMode.%X25 AND NOT #statBooleanInterfaceMo-
deOld.%X25 THEN
0157      #statUnitMode := "LPMLV30_MODE_USER_22";
0158  ELSIF #tempBooleanInterfaceMode.%X26 AND NOT #statBooleanInterfaceMo-
deOld.%X26 THEN
0159      #statUnitMode := "LPMLV30_MODE_USER_23";
0160  ELSIF #tempBooleanInterfaceMode.%X27 AND NOT #statBooleanInterfaceMo-
deOld.%X27 THEN
0161      #statUnitMode := "LPMLV30_MODE_USER_24";
0162  ELSIF #tempBooleanInterfaceMode.%X28 AND NOT #statBooleanInterfaceMo-
deOld.%X28 THEN
0163      #statUnitMode := "LPMLV30_MODE_USER_25";
0164  ELSIF #tempBooleanInterfaceMode.%X29 AND NOT #statBooleanInterfaceMo-
deOld.%X29 THEN
0165      #statUnitMode := "LPMLV30_MODE_USER_26";
0166  ELSIF #tempBooleanInterfaceMode.%X30 AND NOT #statBooleanInterfaceMo-
deOld.%X30 THEN
0167      #statUnitMode := "LPMLV30_MODE_USER_27";
0168  ELSIF #tempBooleanInterfaceMode.%X31 AND NOT #statBooleanInterfaceMo-
deOld.%X31 THEN
0169      #statUnitMode := "LPMLV30_MODE_USER_28";
0170  ELSIF #tempBooleanInterfaceMode.%X1 AND NOT #statBooleanInterfaceMo-
deOld.%X1 THEN
0171      #statUnitMode := "LPMLV30_MODE_PRODUCTION";
0172  END_IF;
0173  #statBooleanInterfaceModeOld := #tempBooleanInterfaceMode; // Save values for
edge detection
0174
0175  // Evaluate CntrlCmd in boolean interface; priority descending: abort, stop,
other commands
0176  #statCmdChangeRequest := TRUE; // Change request static TRUE
0177  #tempBooleanInterfaceCmd := 0;
0178
0179  #tempBooleanInterfaceCmd.%X1 := #ResetCmdRequest;
0180  #tempBooleanInterfaceCmd.%X2 := #StartCmdRequest;
0181  #tempBooleanInterfaceCmd.%X4 := #HoldCmdRequest;
0182  #tempBooleanInterfaceCmd.%X5 := #UnholdCmdRequest;
0183  #tempBooleanInterfaceCmd.%X6 := #SuspendCmdRequest;
0184  #tempBooleanInterfaceCmd.%X7 := #UnsuspendCmdRequest;
0185  #tempBooleanInterfaceCmd.%X9 := #ClearCmdRequest;

```

```

0186 #tempBooleanInterfaceCmd.%X10 := #CompleteCmdRequest; // Remember, complete
command is not specified in the OMAC standard
0187
0188 IF #AbortCmdRequest THEN
0189 #tempCntrlCmd := "LPMLV30_CMD_ABORT";
0190 ELSIF #StopCmdRequest THEN
0191 #tempCntrlCmd := "LPMLV30_CMD_STOP";
0192 ELSE
0193 #tempCntrlCmd := "LPMLV30_CMD_UNDEFINED";
0194
0195 // Selection indicator: Rising edge at input, Priority (in case of multiple
selection): complete, reset, start, hold, unhold, suspend, unsuspend, clear
0196 IF #tempBooleanInterfaceCmd.%X10 AND NOT #statBooleanInterfaceCm-
dOld.%X10 THEN
0197 #tempCntrlCmd := "LPMLV30_CMD_COMPLETE";
0198 ELSIF #tempBooleanInterfaceCmd.%X1 AND NOT #statBooleanInterfaceCm-
dOld.%X1 THEN
0199 #tempCntrlCmd := "LPMLV30_CMD_RESET";
0200 ELSIF #tempBooleanInterfaceCmd.%X2 AND NOT #statBooleanInterfaceCm-
dOld.%X2 THEN
0201 #tempCntrlCmd := "LPMLV30_CMD_START";
0202 ELSIF #tempBooleanInterfaceCmd.%X4 AND NOT #statBooleanInterfaceCm-
dOld.%X4 THEN
0203 #tempCntrlCmd := "LPMLV30_CMD_HOLD";
0204 ELSIF #tempBooleanInterfaceCmd.%X5 AND NOT #statBooleanInterfaceCm-
dOld.%X5 THEN
0205 #tempCntrlCmd := "LPMLV30_CMD_UNHOLD";
0206 ELSIF #tempBooleanInterfaceCmd.%X6 AND NOT #statBooleanInterfaceCm-
dOld.%X6 THEN
0207 #tempCntrlCmd := "LPMLV30_CMD_SUSPEND";
0208 ELSIF #tempBooleanInterfaceCmd.%X7 AND NOT #statBooleanInterfaceCm-
dOld.%X7 THEN
0209 #tempCntrlCmd := "LPMLV30_CMD_UNsuspend";
0210 ELSIF #tempBooleanInterfaceCmd.%X9 AND NOT #statBooleanInterfaceCm-
dOld.%X9 THEN
0211 #tempCntrlCmd := "LPMLV30_CMD_CLEAR";
0212 END_IF;
0213 END_IF; // End: abort or stop cmd
0214
0215 #statBooleanInterfaceCmdOld := #tempBooleanInterfaceCmd; // Save values for
edge detection
0216 END_IF; // End: boolean interface
0217 END_IF; // End: As from second cycle
0218 // End: Evaluation of input parameters -> UnitMode, CntrlCmd
0219
0220 //-----
//-----
0221 // Unit mode manager
0222 //-----
//-----
0223 IF NOT #statFirstCycle THEN // As from second cycle
0224 //-----
0225 //Message handling for mode manager: head part
0226 //-----

```

```

0227 IF #statUnitModeChangeRequest AND #statUnitMode <> #statUnitModeOld AND #statU-
nitMode <> "LPMLV30_MODE_INVALID" THEN
0228 #tempDiagnosticsBufferIndex := #tempDiagnosticsBufferIndex + 1; // Next buffer
index
0229 IF #tempDiagnosticsBufferIndex > "LPMLV30_DIAG_BUFFER_UPPER_LIM" THEN
0230 #tempDiagnosticsBufferIndex := 0;
0231 END_IF;
0232 // Check for defined unit modes
0233 #tempRetVal := RD_SYS_T(#diagnostics.buffer[#tempDiagnosticsBufferIndex].time-
stamp);
0234 #diagnostics.buffer[#tempDiagnosticsBufferIndex].UnitModeCur-
rent := DINT_TO_BYTE(#UnitModeCurrent);
0235 #diagnostics.buffer[#tempDiagnosticsBufferIndex].StateCur-
rent := DINT_TO_BYTE(#StateCurrent);
0236 #diagnostics.buffer[#tempDiagnosticsBufferIndex].Uni-
tMode := DINT_TO_BYTE(#statUnitMode);
0237 #diagnostics.buffer[#tempDiagnosticsBufferIn-
dex].CntrlCmd := DINT_TO_BYTE(#tempCntrlCmd);
0238 #diagnostics.buffer[#tempDiagnosticsBufferIndex].SC := #SC;
0239 #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_NO_MESSAGE;
0240
0241 IF #statUnitMode < "LPMLV30_MODE_INVALID" OR #statUnit-
Mode > "LPMLV30_MODE_USER_28" THEN
0242 #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_MODE_NOT_DE-
FINED;
0243 ELSIF #statUnitMode = #statUnitModeCurrentInternal THEN
0244 #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_MODE_AL-
READY_ACTIVE;
0245 END_IF;
0246 END_IF; // End: message handling for mode manager: head part
0247 END_IF; // End: As from second cycle
0248
0249 //-----
0250 // Unit mode machine
0251 //-----
0252 // Check if unit mode has to be changed
0253 IF (#statUnitModeChangeRequest AND
0254 #statUnitMode <> #statUnitModeCurrentInternal AND
0255 #statUnitMode <> #statUnitModeOld AND
0256 #statUnitMode > "LPMLV30_MODE_INVALID" AND
0257 #statUnitMode <= "LPMLV30_MODE_USER_28"
0258 ) OR
0259 #statFirstCycle
0260 THEN
0261 // Check if state is available
0262 IF #statStateInternal = "LPMLV30_STATE_EXECUTE" OR // Every state but not in
execute
0263 #statStateInternal <> #StateRequested // Means to be in wait state or target
state
0264 THEN
0265 #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_MODE_TRANSI-
TION_NOT_ALLOWED;
0266 ELSE
0267 IF #statConfiguration.disabledUnitModes[#statUnitMode] = FALSE THEN
0268 // State in target unit mode available

```

```
0269     IF ((NOT #statConfiguration.disabledStatesInUnitModes[#statUnit-
0270 Mode].%X17 = FALSE) AND (#statStateInternal = "LPMLV30_STATE_COMPLETE")) OR
0271     ((NOT #statConfiguration.disabledStatesInUnitModes[#statUnit-
0272 Mode].%X11 = FALSE) AND (#statStateInternal = "LPMLV30_STATE_HELD")) OR
0273     ((NOT #statConfiguration.disabledStatesInUnitModes[#statUnit-
0274 Mode].%X5 = FALSE) AND (#statStateInternal = "LPMLV30_STATE_SUSPENDED"))
0275     THEN
0276     #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_MODE_TRANSI-
0277     TION_NOT_ALLOWED;
0278     ELSE
0279     // Set outputs
0280     #UnitModeChangeInProgress := TRUE;
0281     #UnitModeRequested := #statUnitMode;
0282     // Set internal variable
0283     #statUnitModeCurrentInternal := #statUnitMode;
0284     // Read current configuration for state
0285     #statStatesInCurrentMode.clearing := NOT #statConfiguration.disabledStatesI-
0286     nUnitModes[#statUnitMode].%X1;
0287     #statStatesInCurrentMode.stopped := NOT #statConfiguration.disabledStatesInU-
0288     nitModes[#statUnitMode].%X2;
0289     #statStatesInCurrentMode.starting := NOT #statConfiguration.disabledStatesI-
0290     nUnitModes[#statUnitMode].%X3;
0291     #statStatesInCurrentMode.idle := NOT #statConfiguration.disabledStatesInU-
0292     nitModes[#statUnitMode].%X4;
0293     #statStatesInCurrentMode.suspended := NOT #statConfiguration.disabledStatesI-
0294     nUnitModes[#statUnitMode].%X5;
0295     #statStatesInCurrentMode.execute := NOT #statConfiguration.disabledStatesInU-
0296     nitModes[#statUnitMode].%X6;
0297     #statStatesInCurrentMode.stopping := NOT #statConfiguration.disabledStatesI-
0298     nUnitModes[#statUnitMode].%X7;
0299     #statStatesInCurrentMode.aborting := NOT #statConfiguration.disabledStatesI-
0300     nUnitModes[#statUnitMode].%X8;
0301     #statStatesInCurrentMode.aborted := NOT #statConfiguration.disabledStatesInU-
0302     nitModes[#statUnitMode].%X9;
0303     #statStatesInCurrentMode.holding := NOT #statConfiguration.disabledStatesInU-
0304     nitModes[#statUnitMode].%X10;
0305     #statStatesInCurrentMode.held := NOT #statConfiguration.disabledStatesInU-
0306     nitModes[#statUnitMode].%X11;
0307     #statStatesInCurrentMode.unholding := NOT #statConfiguration.disabledStatesI-
0308     nUnitModes[#statUnitMode].%X12;
0309     #statStatesInCurrentMode.suspending := NOT #statConfiguration.disabledState-
0310     sInUnitModes[#statUnitMode].%X13;
0311     #statStatesInCurrentMode.unsuspending := NOT #statConfiguration.disabledSta-
0312     tesInUnitModes[#statUnitMode].%X14;
0313     #statStatesInCurrentMode.resetting := NOT #statConfiguration.disabledStatesI-
0314     nUnitModes[#statUnitMode].%X15;
0315     #statStatesInCurrentMode.completing := NOT #statConfiguration.disabledState-
0316     sInUnitModes[#statUnitMode].%X16;
0317     #statStatesInCurrentMode.complete := NOT #statConfiguration.disabledStatesI-
0318     nUnitModes[#statUnitMode].%X17;
0319     END_IF; // End: state in target unit mode available
0320     ELSE
0321     #diagnostics.buffer[#tempDiagnosticsBufferIndex].mes-
0322     sage := #MSG_REQ_MODE_NOT_CONFIGURED;
```

```

0302  END_IF;
0303  END_IF;
0304
0305  // Set the minimum required states for every unit mode
0306  IF NOT #statStatesInCurrentMode.stopped OR
0307  NOT #statStatesInCurrentMode.idle OR
0308  NOT #statStatesInCurrentMode.execute OR
0309  NOT #statStatesInCurrentMode.aborted OR
0310  NOT #statStatesInCurrentMode.resetting OR
0311  NOT #statStatesInCurrentMode.starting OR
0312  NOT #statStatesInCurrentMode.stopping OR
0313  NOT #statStatesInCurrentMode.clearing OR
0314  NOT #statStatesInCurrentMode.aborting OR
0315  (NOT #statStatesInCurrentMode.complete AND #statStatesInCurrentMode.complet-
ing) OR
0316  (NOT #statStatesInCurrentMode.suspended AND (#statStatesInCurrentMode.suspend-
ing OR #statStatesInCurrentMode.unsuspending)) OR
0317  (NOT #statStatesInCurrentMode.held AND (#statStatesInCurrentMode.hold-
ing OR #statStatesInCurrentMode.unholding))
0318  THEN
0319  #statStatesInCurrentMode.stopped := TRUE;
0320  #statStatesInCurrentMode.idle := TRUE;
0321  #statStatesInCurrentMode.execute := TRUE;
0322  #statStatesInCurrentMode.aborted := TRUE;
0323
0324  #statStatesInCurrentMode.resetting := TRUE;
0325  #statStatesInCurrentMode.starting := TRUE;
0326  #statStatesInCurrentMode.aborting := TRUE;
0327  #statStatesInCurrentMode.stopping := TRUE;
0328  #statStatesInCurrentMode.clearing := TRUE;
0329
0330  // Held
0331  IF NOT #statStatesInCurrentMode.held
0332  AND (#statStatesInCurrentMode.holding OR #statStatesInCurrentMode.unholding)
0333  THEN
0334  #statStatesInCurrentMode.held := TRUE;
0335  #statStatesInCurrentMode.holding := TRUE;
0336  #statStatesInCurrentMode.unholding := TRUE;
0337  END_IF;
0338  // Suspended
0339  IF NOT #statStatesInCurrentMode.suspended
0340  AND (#statStatesInCurrentMode.suspending OR #statStatesInCurrentMode.unsus-
pending)
0341  THEN
0342  #statStatesInCurrentMode.suspended := TRUE;
0343  #statStatesInCurrentMode.suspending := TRUE;
0344  #statStatesInCurrentMode.unsuspending := TRUE;
0345  END_IF;
0346  // Complete
0347  IF NOT #statStatesInCurrentMode.complete
0348  AND #statStatesInCurrentMode.completing
0349  THEN
0350  #statStatesInCurrentMode.complete := TRUE;
0351  #statStatesInCurrentMode.completing := TRUE;
0352  END_IF;

```

```
0353
0354   #tempDiagnosticsBufferIndex := #tempDiagnosticsBufferIndex + 1; // Next buffer
index
0355   IF #tempDiagnosticsBufferIndex > "LPMLV30_DIAG_BUFFER_UPPER_LIM" THEN
0356     #tempDiagnosticsBufferIndex := 0;
0357   END_IF;
0358
0359   // Write diagnostics
0360   #tempRetVal := RD_SYS_T(#diagnostics.buffer[#tempDiagnosticsBufferIndex].time-
stamp);
0361   #diagnostics.buffer[#tempDiagnosticsBufferIndex].UnitModeCur-
rent := DINT_TO_BYTE(#UnitModeCurrent);
0362   #diagnostics.buffer[#tempDiagnosticsBufferIndex].StateCur-
rent := DINT_TO_BYTE(#StateCurrent);
0363   #diagnostics.buffer[#tempDiagnosticsBufferIndex].Uni-
tMode := DINT_TO_BYTE(#statUnitMode);
0364   #diagnostics.buffer[#tempDiagnosticsBufferIn-
dex].CntrlCmd := DINT_TO_BYTE(#tempCntrlCmd);
0365   #diagnostics.buffer[#tempDiagnosticsBufferIndex].SC := #SC;
0366   #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_STATE_CON-
FIG_FORCED;
0367   END_IF; // End: set the minimum required states for every unit mode
0368
0369   #UnitModeCurrent := #statUnitModeCurrentInternal; // Write current unit mode
on output
0370
0371   // Write correct state configuration of current mode
0372   #StatesDisabled.%X1 := NOT #statStatesInCurrentMode.clearing;
0373   #StatesDisabled.%X2 := NOT #statStatesInCurrentMode.stopped;
0374   #StatesDisabled.%X3 := NOT #statStatesInCurrentMode.starting;
0375   #StatesDisabled.%X4 := NOT #statStatesInCurrentMode.idle;
0376   #StatesDisabled.%X5 := NOT #statStatesInCurrentMode.suspended;
0377   #StatesDisabled.%X6 := NOT #statStatesInCurrentMode.execute;
0378   #StatesDisabled.%X7 := NOT #statStatesInCurrentMode.stopping;
0379   #StatesDisabled.%X8 := NOT #statStatesInCurrentMode.aborting;
0380   #StatesDisabled.%X9 := NOT #statStatesInCurrentMode.aborted;
0381   #StatesDisabled.%X10 := NOT #statStatesInCurrentMode.holding;
0382   #StatesDisabled.%X11 := NOT #statStatesInCurrentMode.held;
0383   #StatesDisabled.%X12 := NOT #statStatesInCurrentMode.unholding;
0384   #StatesDisabled.%X13 := NOT #statStatesInCurrentMode.suspending;
0385   #StatesDisabled.%X14 := NOT #statStatesInCurrentMode.unsuspending;
0386   #StatesDisabled.%X15 := NOT #statStatesInCurrentMode.resetting;
0387   #StatesDisabled.%X16 := NOT #statStatesInCurrentMode.completing;
0388   #StatesDisabled.%X17 := NOT #statStatesInCurrentMode.complete;
0389
0390   //-----
0391   // Message handling for mode manager: foot part
0392   //-----
0393   IF NOT #statFirstCycle THEN
0394     IF #diagnostics.buffer[#tempDiagnosticsBufferIndex].message = #MSG_NO_MES-
SAGE THEN // ELSE: Unit mode wasn't changed successfully
0395       #diagnostics.buffer[#tempDiagnosticsBufferIndex].mes-
sage := #MSG_MODE_CHANGED_SUCCESSFULLY;
0396     END_IF;
0397     END_IF;
```

```

0398 ELSE
0399     #UnitModeChangeInProgress := FALSE; // No unit mode change
0400 END_IF; // End: check if unit mode has to be changed
0401 // End: Unit mode manager
0402
0403 //-----
0404 // State manager
0405 //-----
0406 IF NOT #statFirstCycle THEN // As from second cycle
0407     IF #SC AND NOT #statSCold THEN // Detect a rising edge at state complete signal
0408         #tempSCInternal := TRUE;
0409     ELSE
0410         #tempSCInternal := FALSE;
0411     END_IF;
0412     #statSCold := #SC; // End: Detect a rising edge at state complete signal
0413
0414 //-----
0415 // Message handling for state manager: head part
0416 //-----
0417 IF #tempSCInternal THEN // Check if rising edge of SC was detected
0418     #tempDiagnosticsBufferIndex := #tempDiagnosticsBufferIndex + 1; // Next buffer
index
0419 IF #tempDiagnosticsBufferIndex > "LPMLV30_DIAG_BUFFER_UPPER_LIM" THEN
0420     #tempDiagnosticsBufferIndex := 0;
0421 END_IF;
0422 #tempDiagnosticsBufferIndexSC := #tempDiagnosticsBufferIndex;
0423
0424 // Check current state to use SC
0425 IF #statStateInternal = "LPMLV30_STATE_CLEARING"      OR
0426     #statStateInternal = "LPMLV30_STATE_STARTING"     OR
0427     #statStateInternal = "LPMLV30_STATE_STOPPING"    OR
0428     #statStateInternal = "LPMLV30_STATE_ABORTING"   OR
0429     #statStateInternal = "LPMLV30_STATE_HOLDING"    OR
0430     #statStateInternal = "LPMLV30_STATE_UNHOLDING"  OR
0431     #statStateInternal = "LPMLV30_STATE_SUSPENDING" OR
0432     #statStateInternal = "LPMLV30_STATE_UNSPENDING" OR
0433     #statStateInternal = "LPMLV30_STATE_RESETTING"  OR
0434     #statStateInternal = "LPMLV30_STATE_COMPLETING" OR
0435     #statStateInternal = "LPMLV30_STATE_EXECUTE"
0436 THEN // Write diagnostics
0437     #tempRetVal := RD_SYS_T(#diagnostics.buffer[#tempDiagnosticsBufferIn-
dex].timestamp);
0438     #diagnostics.buffer[#tempDiagnosticsBufferIndex].UnitModeCur-
rent := DINT_TO_BYTE(#UnitModeCurrent);
0439     #diagnostics.buffer[#tempDiagnosticsBufferIndex].StateCur-
rent := DINT_TO_BYTE(#StateCurrent);
0440     #diagnostics.buffer[#tempDiagnosticsBufferIndex].Uni-
tMode := DINT_TO_BYTE(#statUnitMode);
0441     #diagnostics.buffer[#tempDiagnosticsBufferIn-
dex].CntrlCmd := DINT_TO_BYTE(#tempCntrlCmd);
0442     #diagnostics.buffer[#tempDiagnosticsBufferIndex].SC := #SC;
0443     #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_NO_MESSAGE;
0444 ELSE // Write diagnostics

```

```

0445     #tempRetVal := RD_SYS_T(#diagnostics.buffer[#tempDiagnosticsBufferIn-
0446     dex].timestamp);
0446     #diagnostics.buffer[#tempDiagnosticsBufferIndex].UnitModeCur-
0447     rent := DINT_TO_BYTE(#UnitModeCurrent);
0447     #diagnostics.buffer[#tempDiagnosticsBufferIndex].StateCur-
0448     rent := DINT_TO_BYTE(#StateCurrent);
0448     #diagnostics.buffer[#tempDiagnosticsBufferIndex].Uni-
0449     tMode := DINT_TO_BYTE(#statUnitMode);
0449     #diagnostics.buffer[#tempDiagnosticsBufferIn-
0450     dex].CntrlCmd := DINT_TO_BYTE(#tempCntrlCmd);
0450     #diagnostics.buffer[#tempDiagnosticsBufferIndex].SC := #SC;
0451     #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_SC_NOT_AL-
LOWED;
0452     END_IF; // End: Check current state to use SC
0453     END_IF;
0454
0455     // Check if defined control command was sent correctly
0456     IF #statCmdChangeRequest AND #tempCntrlCmd <> #statCntrlCm-
dOld AND #tempCntrlCmd <> "LPMLV30_CMD_UNDEFINED" THEN
0457     IF #tempCntrlCmd = "LPMLV30_CMD_RESET" OR
0458     #tempCntrlCmd = "LPMLV30_CMD_START" OR
0459     #tempCntrlCmd = "LPMLV30_CMD_STOP" OR
0460     #tempCntrlCmd = "LPMLV30_CMD_HOLD" OR
0461     #tempCntrlCmd = "LPMLV30_CMD_UNHOLD" OR
0462     #tempCntrlCmd = "LPMLV30_CMD_SUSPEND" OR
0463     #tempCntrlCmd = "LPMLV30_CMD_UNsuspend" OR
0464     #tempCntrlCmd = "LPMLV30_CMD_ABORT" OR
0465     #tempCntrlCmd = "LPMLV30_CMD_CLEAR" OR
0466     #tempCntrlCmd = "LPMLV30_CMD_COMPLETE" // Info: Complete command not in the
OMAC standard
0467     THEN
0468     #tempDiagnosticsBufferIndex := #tempDiagnosticsBufferIndex + 1; // Next buf-
fer index
0469     IF #tempDiagnosticsBufferIndex > "LPMLV30_DIAG_BUFFER_UPPER_LIM" THEN
0470     #tempDiagnosticsBufferIndex := 0;
0471     END_IF;
0472     #tempDiagnosticsBufferIndexCmd := #tempDiagnosticsBufferIndex;
0473
0474     // Write diagnostics
0475     #tempRetVal := RD_SYS_T(#diagnostics.buffer[#tempDiagnosticsBufferIn-
dex].timestamp);
0476     #diagnostics.buffer[#tempDiagnosticsBufferIndex].UnitModeCur-
rent := DINT_TO_BYTE(#UnitModeCurrent);
0477     #diagnostics.buffer[#tempDiagnosticsBufferIndex].StateCur-
rent := DINT_TO_BYTE(#StateCurrent);
0478     #diagnostics.buffer[#tempDiagnosticsBufferIndex].Uni-
tMode := DINT_TO_BYTE(#statUnitMode);
0479     #diagnostics.buffer[#tempDiagnosticsBufferIn-
dex].CntrlCmd := DINT_TO_BYTE(#tempCntrlCmd);
0480     #diagnostics.buffer[#tempDiagnosticsBufferIndex].SC := #SC;
0481     #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_NO_MESSAGE;
0482     // Write message if CntrlCmd is not allowed
0483     IF ((#statStateInternal = "LPMLV30_STATE_CLEARING" OR
0484     #statStateInternal = "LPMLV30_STATE_STARTING" OR
0485     #statStateInternal = "LPMLV30_STATE_STOPPING" OR

```

```

0486     #statStateInternal = "LPMLV30_STATE_ABORTING" OR
0487     #statStateInternal = "LPMLV30_STATE_HOLDING" OR
0488     #statStateInternal = "LPMLV30_STATE_UNHOLDING" OR
0489     #statStateInternal = "LPMLV30_STATE_SUSPENDING" OR
0490     #statStateInternal = "LPMLV30_STATE_UNSPENDING" OR
0491     #statStateInternal = "LPMLV30_STATE_RESETTING" OR
0492     #statStateInternal = "LPMLV30_STATE_COMPLETING") AND
0493     NOT (#tempCntrlCmd = "LPMLV30_CMD_ABORT") AND NOT (#tempCntrlCmd =
"LPMLV30_CMD_STOP")) OR
0494     (#statStateInternal = "LPMLV30_STATE_ABORT-
TED" AND (#tempCntrlCmd = "LPMLV30_CMD_ABORT"))
0495     THEN
0496     #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_CMD_NOT_AL-
LOWED;
0497     END_IF;
0498     ELSE
0499     // Not defined control command was sent
0500     #tempDiagnosticsBufferIndex := #tempDiagnosticsBufferIndex + 1; // Next buf-
fer index
0501     IF #tempDiagnosticsBufferIndex > "LPMLV30_DIAG_BUFFER_UPPER_LIM" THEN
0502     #tempDiagnosticsBufferIndex := 0;
0503     END_IF;
0504     #tempDiagnosticsBufferIndexCmd := #tempDiagnosticsBufferIndex;
0505
0506     // Write diagnostics
0507     #tempRetVal := RD_SYS_T(#diagnostics.buffer[#tempDiagnosticsBufferIn-
dex].timestamp);
0508     #diagnostics.buffer[#tempDiagnosticsBufferIndex].UnitModeCur-
rent := DINT_TO_BYTE(#UnitModeCurrent);
0509     #diagnostics.buffer[#tempDiagnosticsBufferIndex].StateCur-
rent := DINT_TO_BYTE(#StateCurrent);
0510     #diagnostics.buffer[#tempDiagnosticsBufferIndex].Uni-
tMode := DINT_TO_BYTE(#statUnitMode);
0511     #diagnostics.buffer[#tempDiagnosticsBufferIn-
dex].CntrlCmd := DINT_TO_BYTE(#tempCntrlCmd);
0512     #diagnostics.buffer[#tempDiagnosticsBufferIndex].SC := #SC;
0513     #diagnostics.buffer[#tempDiagnosticsBufferIndex].message := #MSG_CMD_NOT_DE-
FINED;
0514     END_IF;
0515     END_IF; // End: message handling for state manager: head part
0516
0517     //-----
0518     // State machine
0519     //-----
0520     IF #statCmdChangeRequest AND #tempCntrlCmd <> #statCntrlCm-
dOld AND #tempCntrlCmd = "LPMLV30_CMD_ABORT" AND // Check if
abort command was sent correctly
0521     #statStateInternal <> "LPMLV30_STATE_ABORTED" AND
0522     #statStateInternal <> "LPMLV30_STATE_ABORTING"
0523     THEN
0524     // Abort command was sent correctly
0525     #StateChangeInProgress := TRUE;
0526     #StateRequested := "LPMLV30_STATE_ABORTED";
0527     // Select next state
0528     IF #statStatesInCurrentMode.aborting THEN

```



```

0576     #StateRequested      := "LPMLV30_STATE_COMPLETE";
0577     #statStateInternal  := "LPMLV30_STATE_COMPLETING";
0578     ELSIF #statStatesInCurrentMode.complete THEN
0579     #StateChangeInProgress := TRUE;
0580     #StateRequested      := "LPMLV30_STATE_COMPLETE";
0581     #statStateInternal  := "LPMLV30_STATE_COMPLETE";
0582     ELSE
0583     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0584     END_IF;
0585
0586     "LPMLV30_CMD_RESET":
0587     IF NOT #statStatesInCurrentMode.completing AND NOT #statStatesInCurrent-
Mode.complete THEN
0588     // Check if valid state for command is in use by current unit mode
0589     IF #statStatesInCurrentMode.resetting THEN
0590     #StateChangeInProgress := TRUE;
0591     #StateRequested      := "LPMLV30_STATE_IDLE";
0592     #statStateInternal  := "LPMLV30_STATE_RESETTING";
0593     ELSIF #statStatesInCurrentMode.idle THEN
0594     #StateChangeInProgress := TRUE;
0595     #StateRequested      := "LPMLV30_STATE_IDLE";
0596     #statStateInternal  := "LPMLV30_STATE_IDLE";
0597     ELSE
0598     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0599     END_IF;
0600     END_IF;
0601
0602     "LPMLV30_CMD_HOLD":
0603     // Check if valid state for command is in use by current unit mode
0604     IF #statStatesInCurrentMode.holding THEN
0605     #StateChangeInProgress := TRUE;
0606     #StateRequested      := "LPMLV30_STATE_HELD";
0607     #statStateInternal  := "LPMLV30_STATE_HOLDING";
0608     ELSIF #statStatesInCurrentMode.held THEN
0609     #StateChangeInProgress := TRUE;
0610     #StateRequested      := "LPMLV30_STATE_HELD";
0611     #statStateInternal  := "LPMLV30_STATE_HELD";
0612     ELSE
0613     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0614     END_IF;
0615
0616     "LPMLV30_CMD_SUSPEND":
0617     // Check if valid state for command is in use by current unit mode
0618     IF #statStatesInCurrentMode.suspending THEN
0619     #StateChangeInProgress := TRUE;
0620     #StateRequested      := "LPMLV30_STATE_SUSPENDED";
0621     #statStateInternal  := "LPMLV30_STATE_SUSPENDING";
0622     ELSIF #statStatesInCurrentMode.suspended THEN
0623     #StateChangeInProgress := TRUE;
0624     #StateRequested      := "LPMLV30_STATE_SUSPENDED";
0625     #statStateInternal  := "LPMLV30_STATE_SUSPENDED";
0626     ELSE

```

```

0627         #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0628         END_IF;
0629         ELSE
0630         IF #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage = #MSG_NO_MESSAGE THEN // ELSE: Message was already
written in head part of message handling
0631         #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0632         END_IF;
0633         END_CASE;
0634     END_IF;
0635
0636     "LPMLV30_STATE_COMPLETING":
0637     // Check if state is completed
0638     IF #tempSCInternal AND #statStatesInCurrentMode.complete THEN
0639         #statStateInternal := "LPMLV30_STATE_COMPLETE";
0640     ELSIF #tempSCInternal AND NOT #statStatesInCurrentMode.complete THEN
0641         #diagnostics.buffer[#tempDiagnosticsBufferIndexSC].mes-
sage := #MSG_SC_NOT_ALLOWED;
0642     END_IF;
0643
0644     "LPMLV30_STATE_COMPLETE":
0645     #StateChangeInProgress := FALSE;
0646     IF #statCmdChangeRequest AND #tempCntrlCmd <> #statCntrlCm-
dOld AND #tempCntrlCmd <> "LPMLV30_CMD_UNDEFINED" THEN // Check
if state change is requested
0647         CASE #tempCntrlCmd OF // Check control command
0648         "LPMLV30_CMD_RESET":
0649             // Check if valid state for command is in use by current unit mode
0650             IF #statStatesInCurrentMode.resetting THEN
0651                 #StateChangeInProgress := TRUE;
0652                 #StateRequested := "LPMLV30_STATE_IDLE";
0653                 #statStateInternal := "LPMLV30_STATE_RESETTING";
0654             ELSIF #statStatesInCurrentMode.idle THEN
0655                 #StateChangeInProgress := TRUE;
0656                 #StateRequested := "LPMLV30_STATE_IDLE";
0657                 #statStateInternal := "LPMLV30_STATE_IDLE";
0658             ELSE
0659                 #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0660             END_IF;
0661
0662         "LPMLV30_CMD_START":
0663             // Check if valid state for command is in use by current unit mode
0664             IF NOT #statStatesInCurrentMode.resetting AND NOT #statStatesInCurrent-
Mode.idle THEN
0665                 #StateChangeInProgress := TRUE;
0666                 #StateRequested := "LPMLV30_STATE_EXECUTE";
0667             // Check if valid state for command is in use by current unit mode
0668             IF #statStatesInCurrentMode.starting THEN
0669                 #statStateInternal := "LPMLV30_STATE_STARTING";
0670             ELSE
0671                 #statStateInternal := "LPMLV30_STATE_EXECUTE";
0672             END_IF;

```

```

0673         ELSE
0674         #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0675         END_IF;
0676         ELSE
0677         IF #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage = #MSG_NO_MESSAGE THEN // ELSE: Message was already
written in head part of message handling
0678         #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0679         END_IF;
0680         END_CASE;
0681     END_IF;
0682
0683     "LPMLV30_STATE_RESETTING":
0684     // Check if state is completed
0685     // No message handling needed -> Idle is minimum required state
0686     IF #tempSCInternal THEN // ELSE: Wait for state complete signal
0687         #statStateInternal := "LPMLV30_STATE_IDLE";
0688     END_IF;
0689
0690     "LPMLV30_STATE_IDLE":
0691     #StateChangeInProgress := FALSE;
0692     // No message handling needed -> Execute is minimum required state
0693     // Check if valid command is sent
0694     IF #statCmdChangeRequest AND #tempCntrlCmd <> #statCntrlCm-
dOld AND #tempCntrlCmd <> "LPMLV30_CMD_UNDEFINED" THEN // Check
if state change is requested
0695         CASE #tempCntrlCmd OF // Check control command
0696         "LPMLV30_CMD_START":
0697             #StateChangeInProgress := TRUE;
0698             #StateRequested := "LPMLV30_STATE_EXECUTE";
0699             // Check if valid state for command is in use by current mode
0700             IF #statStatesInCurrentMode.starting THEN
0701                 #statStateInternal := "LPMLV30_STATE_STARTING";
0702             ELSE
0703                 #statStateInternal := "LPMLV30_STATE_EXECUTE";
0704             END_IF;
0705         ELSE
0706             IF #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage = #MSG_NO_MESSAGE THEN // ELSE: Message was already
written in head part of message handling
0707             #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0708             END_IF;
0709         END_CASE;
0710     END_IF;
0711
0712     "LPMLV30_STATE_STARTING",
0713     "LPMLV30_STATE_UNHOLDING",
0714     "LPMLV30_STATE_UNSUSPENDING":
0715     // Check if state is completed
0716     IF #tempSCInternal THEN // ELSE: Wait for state complete signal
0717         #statStateInternal := "LPMLV30_STATE_EXECUTE";
0718     END_IF;

```

```

0719
0720     "LPMLV30_STATE_HOLDING":
0721     // Check if state is completed
0722     IF #tempSCInternal AND #statStatesInCurrentMode.held THEN // ELSE: Wait for
state complete signal
0723         #statStateInternal := "LPMLV30_STATE_HELD";
0724     ELSIF #tempSCInternal AND NOT #statStatesInCurrentMode.held THEN
0725         #diagnostics.buffer[#tempDiagnosticsBufferIndexSC].mes-
sage := #MSG_SC_NOT_ALLOWED;
0726     END_IF;
0727
0728     "LPMLV30_STATE_HELD":
0729     #StateChangeInProgress := FALSE;
0730     // Check if valid command is sent
0731     IF #statCmdChangeRequest AND #tempCntrlCmd <> #statCntrlCm-
dOld AND #tempCntrlCmd <> "LPMLV30_CMD_UNDEFINED" THEN // Check
if state change is requested
0732         CASE #tempCntrlCmd OF // Check control command
0733             "LPMLV30_CMD_UNHOLD":
0734                 #StateChangeInProgress := TRUE;
0735                 #StateRequested := "LPMLV30_STATE_EXECUTE";
0736                 // Check if valid state for command is in use by current mode
0737                 IF #statStatesInCurrentMode.unholding THEN
0738                     #statStateInternal := "LPMLV30_STATE_UNHOLDING";
0739                 ELSE
0740                     #statStateInternal := "LPMLV30_STATE_EXECUTE";
0741                 END_IF;
0742             ELSE
0743                 IF #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage = #MSG_NO_MESSAGE THEN // ELSE: Message was already
written in head part of message handling
0744                     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0745                 END_IF;
0746             END_CASE;
0747         END_IF;
0748
0749     "LPMLV30_STATE_SUSPENDING":
0750     // Check if state is completed
0751     IF #tempSCInternal AND #statStatesInCurrentMode.suspended THEN // ELSE: Wait
for state complete signal
0752         #statStateInternal := "LPMLV30_STATE_SUSPENDED";
0753     ELSIF #tempSCInternal AND NOT #statStatesInCurrentMode.suspended THEN
0754         #diagnostics.buffer[#tempDiagnosticsBufferIndexSC].mes-
sage := #MSG_SC_NOT_ALLOWED;
0755     END_IF;
0756
0757     "LPMLV30_STATE_SUSPENDED":
0758     #StateChangeInProgress := FALSE;
0759     // Check if valid command is sent
0760     IF #statCmdChangeRequest AND #tempCntrlCmd <> #statCntrlCm-
dOld AND #tempCntrlCmd <> "LPMLV30_CMD_UNDEFINED" THEN // Check
if state change is requested
0761         CASE #tempCntrlCmd OF // Check control command
0762             "LPMLV30_CMD_UNSPEND":

```

```

0763     #StateChangeInProgress := TRUE;
0764     #StateRequested := "LPMLV30_STATE_EXECUTE";
0765     // Check if valid state for command is in use by current mode
0766     IF #statStatesInCurrentMode.unsuspending THEN
0767     #statStateInternal := "LPMLV30_STATE_UNSUSPENDING";
0768     ELSE
0769     #statStateInternal := "LPMLV30_STATE_EXECUTE";
0770     END_IF;
0771     ELSE
0772     IF #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
message = #MSG_NO_MESSAGE THEN // ELSE: Message was already
written in head part of message handling
0773     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
message := #MSG_CMD_NOT_ALLOWED;
0774     END_IF;
0775     END_CASE;
0776     END_IF;
0777
0778     "LPMLV30_STATE_ABORTING":
0779     // Check if state is completed
0780     IF #tempSCInternal THEN // ELSE: Wait for state complete signal
0781     #statStateInternal := "LPMLV30_STATE_ABORTED";
0782     END_IF;
0783
0784     "LPMLV30_STATE_ABORTED":
0785     #StateChangeInProgress := FALSE;
0786     // Check if valid command is sent
0787     IF #statCmdChangeRequest AND #tempCntrlCmd <> #statCntrlCm-
dOld AND #tempCntrlCmd <> "LPMLV30_CMD_UNDEFINED" THEN // Check
if state change is requested
0788     CASE #tempCntrlCmd OF // Check control command
0789     "LPMLV30_CMD_CLEAR":
0790     #StateChangeInProgress := TRUE;
0791     #StateRequested := "LPMLV30_STATE_STOPPED";
0792     // Check if valid state for command is in use by current mode
0793     IF #statStatesInCurrentMode.clearing THEN
0794     #statStateInternal := "LPMLV30_STATE_CLEARING";
0795     ELSE
0796     #statStateInternal := "LPMLV30_STATE_STOPPED";
0797     END_IF;
0798     ELSE
0799     IF #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
message = #MSG_NO_MESSAGE THEN // ELSE: Message was already
written in head part of message handling
0800     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
message := #MSG_CMD_NOT_ALLOWED;
0801     END_IF;
0802     END_CASE;
0803     END_IF;
0804
0805     "LPMLV30_STATE_CLEARING",
0806     "LPMLV30_STATE_STOPPING":
0807     // Check if state is completed
0808     IF #tempSCInternal THEN // ELSE: Wait for state complete signal
0809     #statStateInternal := "LPMLV30_STATE_STOPPED";

```

```

0810     END_IF;
0811
0812     "LPMLV30_STATE_STOPPED":
0813     #StateChangeInProgress := FALSE;
0814     IF #statCmdChangeRequest AND #tempCntrlCmd <> #statCntrlCm-
dOld AND #tempCntrlCmd <> "LPMLV30_CMD_UNDEFINED" THEN // Check
if state change is requested
0815     // Check if valid command is sent
0816     CASE #tempCntrlCmd OF
0817     "LPMLV30_CMD_RESET":
0818     // Check if valid state for command is in use by current unit mode
0819     IF #statStatesInCurrentMode.resetting THEN
0820     #StateChangeInProgress := TRUE;
0821     #StateRequested      := "LPMLV30_STATE_IDLE";
0822     #statStateInternal  := "LPMLV30_STATE_RESETTING";
0823     ELSIF #statStatesInCurrentMode.idle THEN
0824     #StateChangeInProgress := TRUE;
0825     #StateRequested      := "LPMLV30_STATE_IDLE";
0826     #statStateInternal  := "LPMLV30_STATE_IDLE";
0827     ELSE
0828     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0829     END_IF;
0830
0831     "LPMLV30_CMD_START":
0832     IF NOT #statStatesInCurrentMode.resetting AND NOT #statStatesInCurrent-
Mode.idle THEN
0833     #StateChangeInProgress := TRUE;
0834     #StateRequested := "LPMLV30_STATE_EXECUTE";
0835     // Check if valid state for command is in use by current mode
0836     IF #statStatesInCurrentMode.starting THEN
0837     #statStateInternal := "LPMLV30_STATE_STARTING";
0838     ELSE
0839     #statStateInternal := "LPMLV30_STATE_EXECUTE";
0840     END_IF;
0841     ELSE
0842     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0843     END_IF;
0844     ELSE
0845     IF #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage = #MSG_NO_MESSAGE THEN // ELSE: Message was already
written in head part of message handling
0846     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_CMD_NOT_ALLOWED;
0847     END_IF;
0848     END_CASE;
0849     END_IF;
0850
0851     "LPMLV30_STATE_UNDEFINED":
0852     #StateChangeInProgress := FALSE;
0853     #StateRequested      := "LPMLV30_STATE_STOPPED";
0854     #statStateInternal  := "LPMLV30_STATE_STOPPED";
0855     ELSE
0856     ; // Undefined state

```

```

0857     END_CASE;
0858     END_IF; // End: state machine
0859
0860     //-----
0861     // Message handling for state manager: foot part
0862     //-----
0863     IF #statStateInternal <> #statStateCurrentOld THEN // ELSE: States were not
changed
0864     IF #tempDiagnosticsBufferIndexCmd >= 0 AND #tempDiagnosticsBufferIn-
dexCmd <= "LPMLV30_DIAG_BUFFER_UPPER_LIM" THEN
0865     IF #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].message = #MSG_NO_MES-
SAGE THEN
0866     #diagnostics.buffer[#tempDiagnosticsBufferIndexCmd].mes-
sage := #MSG_STATE_CHANGED_SUCCESSFULLY;
0867     END_IF;
0868     END_IF;
0869     IF #tempDiagnosticsBufferIndexSC >= 0 AND #tempDiagnosticsBufferIn-
dexSC <= "LPMLV30_DIAG_BUFFER_UPPER_LIM" THEN
0870     IF #diagnostics.buffer[#tempDiagnosticsBufferIndexSC].message = #MSG_NO_MES-
SAGE THEN
0871     #diagnostics.buffer[#tempDiagnosticsBufferIndexSC].mes-
sage := #MSG_STATE_CHANGED_SUCCESSFULLY;
0872     END_IF;
0873     END_IF;
0874     END_IF; // End: message handling for state manager: foot part
0875 END_IF; // End: As from second cycle
0876 // End: State manager
0877
0878 //-----
-----
0879 // Write outputs including boolean interface outputs
0880 //-----
-----
0881 IF #statUnitModeCurrentInternal <> #statUnitModeCurrentInternalOld THEN // Check
if new unit mode is selected
0882     #ProductionModeActive := FALSE;
0883     #MaintenanceModeActive := FALSE;
0884     #ManualModeActive := FALSE;
0885     #UserMode01Active := FALSE;
0886     #UserMode02Active := FALSE;
0887     #UserMode03Active := FALSE;
0888     #UserMode04Active := FALSE;
0889     #UserMode05Active := FALSE;
0890     #UserMode06Active := FALSE;
0891     #UserMode07Active := FALSE;
0892     #UserMode08Active := FALSE;
0893     // #UserMode09Active := FALSE;
0894     // #UserMode10Active := FALSE;
0895     // #UserMode11Active := FALSE;
0896     // #UserMode13Active := FALSE;
0897     // #UserMode14Active := FALSE;
0898     // #UserMode15Active := FALSE;
0899     // #UserMode16Active := FALSE;
0900     // #UserMode17Active := FALSE;
0901     // #UserMode18Active := FALSE;

```

```

0902 // #UserMode19Active := FALSE;
0903 // #UserMode20Active := FALSE;
0904 // #UserMode21Active := FALSE;
0905 // #UserMode22Active := FALSE;
0906 // #UserMode23Active := FALSE;
0907 // #UserMode24Active := FALSE;
0908 // #UserMode25Active := FALSE;
0909 // #UserMode26Active := FALSE;
0910 // #UserMode27Active := FALSE;
0911 // #UserMode28Active := FALSE;
0912
0913 CASE #statUnitModeCurrentInternal OF // Set new bit for current unit mode
0914 "LPMLV30_MODE_PRODUCTION":
0915     #ProductionModeActive := TRUE;
0916 "LPMLV30_MODE_MAINTENANCE":
0917     #MaintenanceModeActive := TRUE;
0918 "LPMLV30_MODE_MANUAL":
0919     #ManualModeActive := TRUE;
0920 "LPMLV30_MODE_USER_01":
0921     #UserMode01Active := TRUE;
0922 "LPMLV30_MODE_USER_02":
0923     #UserMode02Active := TRUE;
0924 "LPMLV30_MODE_USER_03":
0925     #UserMode03Active := TRUE;
0926 "LPMLV30_MODE_USER_04":
0927     #UserMode04Active := TRUE;
0928 "LPMLV30_MODE_USER_05":
0929     #UserMode05Active := TRUE;
0930 "LPMLV30_MODE_USER_06":
0931     #UserMode06Active := TRUE;
0932 "LPMLV30_MODE_USER_07":
0933     #UserMode07Active := TRUE;
0934 "LPMLV30_MODE_USER_08":
0935     #UserMode08Active := TRUE;
0936 // LPMLV30_MODE_USER_09:
0937 // #UserMode09Active := TRUE;
0938 // LPMLV30_MODE_USER_10:
0939 // #UserMode10Active := TRUE;
0940 // LPMLV30_MODE_USER_11:
0941 // #UserMode11Active := TRUE;
0942 // LPMLV30_MODE_USER_12:
0943 // #UserMode12Active := TRUE;
0944 // LPMLV30_MODE_USER_13:
0945 // #UserMode13Active := TRUE;
0946 // LPMLV30_MODE_USER_14:
0947 // #UserMode14Active := TRUE;
0948 // LPMLV30_MODE_USER_15:
0949 // #UserMode15Active := TRUE;
0950 // LPMLV30_MODE_USER_16:
0951 // #UserMode16Active := TRUE;
0952 // LPMLV30_MODE_USER_17:
0953 // #UserMode17Active := TRUE;
0954 // LPMLV30_MODE_USER_18:
0955 // #UserMode18Active := TRUE;
0956 // LPMLV30_MODE_USER_19:

```

```

0957 // #UserMode19Active := TRUE;
0958 // LPMLV30_MODE_USER_20:
0959 // #UserMode20Active := TRUE;
0960 // LPMLV30_MODE_USER_21:
0961 // #UserMode21Active := TRUE;
0962 // LPMLV30_MODE_USER_22:
0963 // #UserMode22Active := TRUE;
0964 // LPMLV30_MODE_USER_23:
0965 // #UserMode23Active := TRUE;
0966 // LPMLV30_MODE_USER_24:
0967 // #UserMode25Active := TRUE;
0968 // LPMLV30_MODE_USER_25:
0969 // #UserMode25Active := TRUE;
0970 // LPMLV30_MODE_USER_26:
0971 // #UserMode26Active := TRUE;
0972 // LPMLV30_MODE_USER_27:
0973 // #UserMode27Active := TRUE;
0974 // LPMLV30_MODE_USER_28:
0975 // #UserMode28Active := TRUE;
0976 ELSE
0977     ; // No bit is set
0978 END_CASE;
0979
0980 #statUnitModeCurrentInternalOld := #statUnitModeCurrentInternal;
0981 END_IF;
0982
0983 IF #statStateInternal <> #statStateInternalOld THEN // Check if new state is selected
0984     #StateCurrent := #statStateInternal; // Set output
0985
0986 // Set states of boolean interface out
0987 // Delete old state
0988 #ClearingStateActive := FALSE;
0989 #StoppedStateActive := FALSE;
0990 #StartingStateActive := FALSE;
0991 #IdleStateActive := FALSE;
0992 #SuspendedStateActive := FALSE;
0993 #ExecuteStateActive := FALSE;
0994 #StoppingStateActive := FALSE;
0995 #AbortingStateActive := FALSE;
0996 #AbortedStateActive := FALSE;
0997 #HoldingStateActive := FALSE;
0998 #HeldStateActive := FALSE;
0999 #UnholdingStateActive := FALSE;
1000 #SuspendingStateActive := FALSE;
1001 #UnsuspendingStateActive := FALSE;
1002 #ResettingStateActive := FALSE;
1003 #CompletingStateActive := FALSE;
1004 #CompleteStateActive := FALSE;
1005
1006 // Set new state
1007 CASE #statStateInternal OF
1008     "LPMLV30_STATE_CLEARING":
1009         #ClearingStateActive := TRUE;
1010     "LPMLV30_STATE_STOPPED":

```

```

1011     #StoppedStateActive := TRUE;
1012     "LPMLV30_STATE_STARTING":
1013     #StartingStateActive := TRUE;
1014     "LPMLV30_STATE_IDLE":
1015     #IdleStateActive := TRUE;
1016     "LPMLV30_STATE_SUSPENDED":
1017     #SuspendedStateActive := TRUE;
1018     "LPMLV30_STATE_EXECUTE":
1019     #ExecuteStateActive := TRUE;
1020     "LPMLV30_STATE_STOPPING":
1021     #StoppingStateActive := TRUE;
1022     "LPMLV30_STATE_ABORTING":
1023     #AbortingStateActive := TRUE;
1024     "LPMLV30_STATE_ABORTED":
1025     #AbortedStateActive := TRUE;
1026     "LPMLV30_STATE_HOLDING":
1027     #HoldingStateActive := TRUE;
1028     "LPMLV30_STATE_HELD":
1029     #HeldStateActive := TRUE;
1030     "LPMLV30_STATE_UNHOLDING":
1031     #UnholdingStateActive := TRUE;
1032     "LPMLV30_STATE_SUSPENDING":
1033     #SuspendingStateActive := TRUE;
1034     "LPMLV30_STATE_UNSPUSPENDING":
1035     #UnsuspendingStateActive := TRUE;
1036     "LPMLV30_STATE_RESETTING":
1037     #ResettingStateActive := TRUE;
1038     "LPMLV30_STATE_COMPLETING":
1039     #CompletingStateActive := TRUE;
1040     "LPMLV30_STATE_COMPLETE":
1041     #CompleteStateActive := TRUE;
1042     ELSE
1043     ;
1044     END_CASE; // End: Set new state
1045
1046     #statStateInternalOld := #statStateInternal;
1047 END_IF; // End: Write boolean interface outputs
1048
1049 #diagnostics.bufferIndex := #tempDiagnosticsBufferIndex; // Temporary variable
    is used during block execution -> Copy back/Set output
1050 IF #statFirstCycle THEN
1051     #statFirstCycle := FALSE;
1052 END_IF;
1053
1054 // Save values for edge detection
1055 IF #statUnitModeChangeRequest THEN
1056     #statUnitModeOld := #statUnitMode;
1057 END_IF;
1058 IF #statCmdChangeRequest THEN
1059     #statCntrlCmdOld := #tempCntrlCmd;
1060 END_IF;
1061 IF #statStateInternal <> #statStateCurrentOld THEN
1062     #statStateCurrentOld := #statStateInternal;
1063 END_IF;
1064

```

1065 //

=====

1066 // End of function block: LPMLV30_UnitModeStateManager

1067 //

=====

Program blocks / LPMLV30

LPMLV30_UnitModeStateTimes [FB30101]

LPMLV30_UnitModeStateTimes Properties

General

Name	LPMLV30_UnitModeStateTimes	Number	30101	Type	FB
Language	SCL	Numbering	Automatic		

Information

Title	LPMLV30_UnitModeStateTimes	Author	APC_ERLF	Comment	Support: tech.team.motioncontrol@siemens.com Fax: +49 (0) 9131/98-1297
Family	OMAC	Version		User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
UnitModeCurrent	DInt	"LPMLV30_MODE_INVALID"	Non-retain
StateCurrent	DInt	"LPMLV30_STATE_UNDEFINED"	Non-retain
resetTimes	Bool	false	Non-retain
▼ Output			
ModeCurrentTime	Array[0.."LPMLV30_MODES_UPPER_LIM"] of DInt		Non-retain
ModeCumulativeTime	Array[0.."LPMLV30_MODES_UPPER_LIM"] of DInt		Retain
StateCurrentTime	Array[0.."LPMLV30_MODES_UPPER_LIM", 0.."LPMLV30_STATES_UPPER_LIM"] of DInt		Non-retain
StateCumulativeTime	Array[0.."LPMLV30_MODES_UPPER_LIM", 0.."LPMLV30_STATES_UPPER_LIM"] of DInt		Retain
actualTimeCurrentMode	DInt	0	Non-retain
cumulativeTimeCurrentMode	DInt	0	Non-retain
actualTimeCurrentStates	Array[0.."LPMLV30_STATES_UPPER_LIM"] of DInt		Non-retain
cumulativeTimeCurrentStates	Array[0.."LPMLV30_STATES_UPPER_LIM"] of DInt		Non-retain
AccTimeSinceReset	DInt	0	Retain
InOut			

Name	Data type	Default value	Retain
▼ Static			
statUnitModeCurrentOld	DInt	"LPMLV30_MODE_INVALID"	Non-retain
statStateCurrentOld	DInt	"LPMLV30_STATE_UNDEFINED"	Non-retain
statResetTimesOld	Bool	false	Non-retain
statStartingPointMeasureRuntime	LReal	0.0	Non-retain
statRuntime	LReal	0.0	Non-retain
statFirstCycle	Bool	true	Non-retain
▼ Temp			
tempI	Int		
tempJ	Int		
tempMeasuredRuntime	LReal		
tempRuntimeIntegerPart	DInt		
tempUnitModeCurrent	DInt		
tempStateCurrent	DInt		
Constant			

```

0001 //
=====
0002 // SIEMENS AG
0003 // (c) Copyright 2015 All Rights Reserved
0004 //-----
-----
0005 // Library: LPMLV30
0006 // Tested with: S7-1200 with FW version V4.1, S7-1500 with FW version V1.7
0007 // Engineering: TIA Portal V13 SP1
0008 // Restrictions: ---
0009 // Requirements: S7-1200 / S7-1500
0010 // Functionality: Determination of unit modes and states times (current and also
cumulative times)
0011 //-----
-----
0012 // Change log table:
0013 // Version Date Expert in charge Changes applied
0014 // 03.00.01 29.05.2015 RK First released version
0015 //
=====
=====
0016 // Function block: LPMLV30_UnitModeStateTimes
0017 //
=====
=====
0018
0019 IF #statFirstCycle OR (#resetTimes AND NOT #statResetTimesOld) THEN
0020 // Reset times structures
0021 FOR #tempI := "LPMLV30_MODE_INVALID" TO "LPMLV30_MODES_UPPER_LIM" DO
0022 #ModeCurrentTime[#tempI] := 0;
0023 IF NOT #statFirstCycle THEN // Do only with rising edge at resetTimes
0024 #ModeCumulativeTime[#tempI] := 0;
0025 END_IF;
0026 FOR #tempJ := "LPMLV30_STATE_UNDEFINED" TO "LPMLV30_STATES_UPPER_LIM" DO
0027 #StateCurrentTime[#tempI, #tempJ] := 0;

```

```

0028     IF NOT #statFirstCycle THEN // Do only with rising edge at resetTimes
0029     #StateCumulativeTime[#tempI, #tempJ] := 0;
0030     END_IF;
0031 END_FOR;
0032 END_FOR;
0033 IF NOT #statFirstCycle THEN // Do only with rising edge at resetTimes
0034 #AccTimeSinceReset := 0;
0035 END_IF;
0036
0037 // Reset additional outputs
0038 #actualTimeCurrentMode := 0;
0039 IF NOT #statFirstCycle THEN // Do only with rising edge at resetTimes
0040 #cumulativeTimeCurrentMode := 0;
0041 END_IF;
0042 FOR #tempI := "LPMLV30_STATE_UNDEFINED" TO "LPMLV30_STATES_UPPER_LIM" DO
0043 #actualTimeCurrentStates[#tempI] := 0;
0044 IF NOT #statFirstCycle THEN // Do only with rising edge at resetTimes
0045 #cumulativeTimeCurrentStates[#tempI] := 0;
0046 END_IF;
0047 END_FOR;
0048
0049 // Set starting point for calculation of elapsed seconds
0050 #tempMeasuredRuntime := RUNTIME(#statStartingPointMeasureRuntime);
0051 #statRuntime := 0.0;
0052
0053 IF #statFirstCycle THEN
0054 #statFirstCycle := FALSE;
0055 RETURN; // Nothing else in this cycle to avoid runtime peaks
0056 END_IF;
0057 END_IF;
0058
0059 // Inputs UnitModeCurrent and StateCurrent
0060 #tempUnitModeCurrent := #UnitModeCurrent;
0061 #tempStateCurrent := #StateCurrent;
0062
0063 // Measure program runtime
0064 #tempMeasuredRuntime := RUNTIME(#statStartingPointMeasureRuntime);
0065 IF #tempMeasuredRuntime > 0.0 THEN //value OK
0066 #statRuntime := #statRuntime + #tempMeasuredRuntime;
0067 END_IF;
0068
0069 // Check if at least one second has been reached
0070 IF #statRuntime > 1.0 THEN
0071 #tempRuntimeIntegerPart := TRUNC_DINT(#statRuntime);
0072
0073 IF #statUnitModeCurrentOld >= "LPMLV30_MODE_INVALID" AND #statUnitModeCurrentOld <= "LPMLV30_MODES_UPPER_LIM" THEN
0074 //DINT overflow is not taken into account, because DINT in seconds means approx. 68 years
0075 #ModeCurrentTime[#statUnitModeCurrentOld] := #ModeCurrentTime[#statUnitModeCurrentOld] + #tempRuntimeIntegerPart;
0076 #ModeCumulativeTime[#statUnitModeCurrentOld] := #ModeCumulativeTime[#statUnitModeCurrentOld] + #tempRuntimeIntegerPart;
0077

```

```

0078 IF #statStateCurrentOld >= "LPMLV30_STATE_UNDEFINED" AND #statStateCurren-
tOld <= "LPMLV30_STATES_UPPER_LIM" THEN
0079 //DINT overflow is not taken into account, because DINT in seconds means
approx. 68 years
0080 #StateCurrentTime[#statUnitModeCurrentOld, #statStateCurren-
tOld] := #StateCurrentTime[#statUnitModeCurrentOld, #statStateCurren-
tOld] + #tempRuntimeIntegerPart;
0081 #StateCumulativeTime[#statUnitModeCurrentOld, #statStateCurren-
tOld] := #StateCumulativeTime[#statUnitModeCurrentOld, #statStateCurren-
tOld] + #tempRuntimeIntegerPart;
0082 END_IF;
0083 END_IF;
0084
0085 //DINT overflow is not taken into account, because DINT in seconds means ap-
prox. 68 years
0086 #AccTimeSinceReset := #AccTimeSinceReset + #tempRuntimeIntegerPart;
0087
0088 #statRuntime := #statRuntime - #tempRuntimeIntegerPart; //fractional part of
runtime variable for next cycle
0089 END_IF;
0090
0091 // Check if new unitMode is selected
0092 IF #tempUnitModeCurrent <> #statUnitModeCurrentOld THEN
0093 IF #tempUnitModeCurrent >= "LPMLV30_MODE_INVALID" AND #tempUnitModeCur-
rent <= "LPMLV30_MODES_UPPER_LIM" THEN
0094 #ModeCurrentTime[#tempUnitModeCurrent] := 0;
0095 IF #tempStateCurrent >= "LPMLV30_STATE_UNDEFINED" AND #tempStateCur-
rent <= "LPMLV30_STATES_UPPER_LIM" THEN
0096 #StateCurrentTime[#tempUnitModeCurrent, #tempStateCurrent] := 0;
0097 END_IF;
0098 END_IF;
0099
0100 // Check if new state is selected
0101 ELSIF #tempStateCurrent <> #statStateCurrentOld THEN
0102 IF #tempUnitModeCurrent >= "LPMLV30_MODE_INVALID" AND #tempUnitModeCur-
rent <= "LPMLV30_MODES_UPPER_LIM" AND
0103 #tempStateCurrent >= "LPMLV30_STATE_UNDEFINED" AND #tempStateCur-
rent <= "LPMLV30_STATES_UPPER_LIM"
0104 THEN
0105 #StateCurrentTime[#tempUnitModeCurrent, #tempStateCurrent] := 0;
0106 END_IF;
0107 END_IF;
0108
0109 // Update additional outputs
0110 IF #tempUnitModeCurrent >= "LPMLV30_MODE_INVALID" AND #tempUnitModeCur-
rent <= "LPMLV30_MODES_UPPER_LIM" THEN
0111 #actualTimeCurrentMode := #ModeCurrentTime[#tempUnitModeCurrent];
0112 #cumulativeTimeCurrentMode := #ModeCumulativeTime[#tempUnitModeCurrent];
0113 IF #tempStateCurrent >= "LPMLV30_STATE_UNDEFINED" AND #tempStateCur-
rent <= "LPMLV30_STATES_UPPER_LIM" THEN
0114 #actualTimeCurrentStates := #StateCurrentTime[#tempUnitModeCurrent];
0115 #cumulativeTimeCurrentStates := #StateCumulativeTime[#tempUnitModeCurrent];
0116 ELSE
0117 FOR #tempI := "LPMLV30_STATE_UNDEFINED" TO "LPMLV30_STATES_UPPER_LIM" DO
0118 #actualTimeCurrentStates[#tempI] := 0;

```

```
0119     #cumulativeTimeCurrentStates[#tempI] := 0;
0120     END_FOR;
0121     END_IF;
0122 ELSE
0123     #actualTimeCurrentMode      := 0;
0124     #cumulativeTimeCurrentMode := 0;
0125
0126     FOR #tempI := "LPMLV30_STATE_UNDEFINED" TO "LPMLV30_STATES_UPPER_LIM" DO
0127         #actualTimeCurrentStates[#tempI]      := 0;
0128         #cumulativeTimeCurrentStates[#tempI] := 0;
0129     END_FOR;
0130 END_IF;
0131
0132 // Save values for state change detection
0133 #statResetTimesOld      := #resetTimes;
0134 #statUnitModeCurrentOld := #tempUnitModeCurrent;
0135 #statStateCurrentOld   := #tempStateCurrent;
0136
0137 //
=====
=====
0138 // End of Function block: LPMLV30_UnitModeStateTimes
0139 //
=====
=====
```

Program blocks / Servo_Control

Axis_ObjectPOS [FB501]

Axis_ObjectPOS Properties

General

Name	Axis_ObjectPOS	Number	501	Type	FB
Language	LAD	Numbering	Manual		

Information

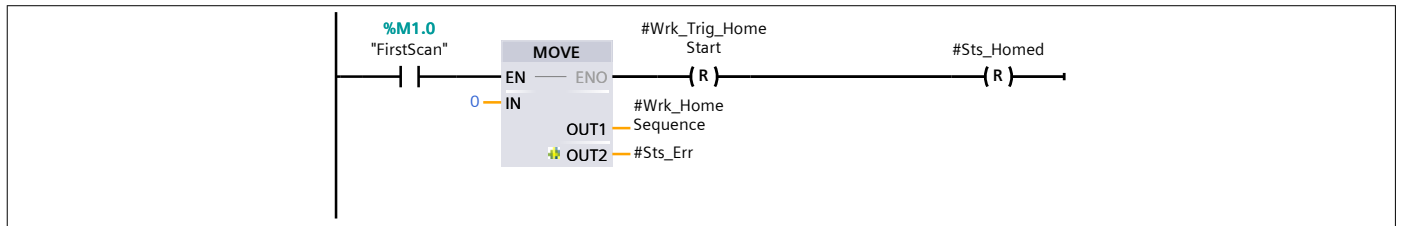
Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
Inp_MasterNoMotion	Bool	false	Non-retain
Cmd_Enable	Bool	false	Non-retain
Cmd_Disable	Bool	false	Non-retain
Cmd_FaultReset	Bool	false	Non-retain
Cmd_Home	Bool	false	Non-retain
Cmd_Abort	Bool	false	Non-retain
Cmd_Stop	Bool	false	Non-retain
▼ Output			
Sts_ER	Bool	false	Non-retain
Sts_PowerEnable	Bool	false	Non-retain
Sts_EnableDone	Bool	false	Non-retain
Sts_DisableDone	Bool	false	Non-retain
Sts_FaultResetDone	Bool	false	Non-retain
Sts_HomeDone	Bool	false	Non-retain
Sts_AbortDone	Bool	false	Non-retain
Sts_StopDone	Bool	false	Non-retain
Sts_AxisOk	Bool	false	Non-retain
Sts_NoMotion	Bool	false	Non-retain
Sts_Homed	Bool	false	Non-retain
▼ InOut			
Ref_Axis	TO_PositioningAxis		
▼ Static			
Cfg_UseVirtualMaster	Bool	false	Non-retain
Cfg_HomeEnabled	Bool	true	Non-retain
Cfg_StopEnabled	Bool	true	Non-retain
Cfg_AbortEnabled	Bool	true	Non-retain
Cfg_ZeroSpeedTolerance	Real	0.1	Non-retain
Cfg_AbortRamp	Real	100.0	Non-retain
Cfg_StopRamp	Real	20.0	Non-retain
Cfg_HomePosition	Real	0.0	Non-retain
Sts_Err	DInt	0	Non-retain
Err_MCBlockFault	Bool	false	Non-retain
Err_ErrorID	Word	16#0	Non-retain
Out_StatusWord	DWord	16#0	Non-retain

Name	Data type	Default value	Retain
Out_ErrorWord	DWord	16#0	Non-retain
Out_WarnWord	DWord	16#0	Non-retain
Wrk_DecelRate	Real	0.0	Non-retain
Wrk_TempR	Real	0.0	Non-retain
Wrk_Velocity	Real	0.0	Non-retain
Wrk_MaskedStillStatus	Real	0.0	Non-retain
Wrk_ER_Trig	Bool	false	Non-retain
Wrk_Trig_HomeStart	Bool	false	Non-retain
Wrk_AtHome1	Bool	false	Non-retain
Wrk_AtHome2	Bool	false	Non-retain
Wrk_AtHome3	Bool	false	Non-retain
Wrk_HomeSwPresent	Bool	false	Non-retain
Wrk_HomeJogFwd	Bool	false	Non-retain
Wrk_Trig_Init	Bool	false	Non-retain
Wrk_HaltEnable	Bool	false	Non-retain
Wrk_Sts_ER	Bool	false	Non-retain
Wrk_StatusWord_Homed	Bool	false	Non-retain
Wrk_Sts_NoMotion	Bool	false	Non-retain
Wrk_Trig_UnlatchHomeStatus	Bool	false	Non-retain
Wrk_FaultReset_Delay	TON_TIME		Non-retain
Wrk_DisabWatchdog	TON_TIME		Non-retain
Wrk_EnabWatchdog	TON_TIME		Non-retain
Wrk_HomeSequence	DInt	0	Non-retain
Wrk_HomeMode	Int	0	Non-retain
Wrk_StatusWord	DWord	16#0	Non-retain
Wrk_ErrorWord	DWord	16#0	Non-retain
Wrk_MC_Power	MC_POWER		
Wrk_MC_Halt	MC_HALT		
Wrk_MC_Home	MC_HOME		
Wrk_MC_Reset	MC_RESET		
Wrk_MC_Jog	MC_MOVEJOG		
Wrk_Jog_Velocity	LReal	0.0	Non-retain
MC_Power_Enable	Bool	false	Non-retain
Temp			
Constant			

Network 1: First scan initialization

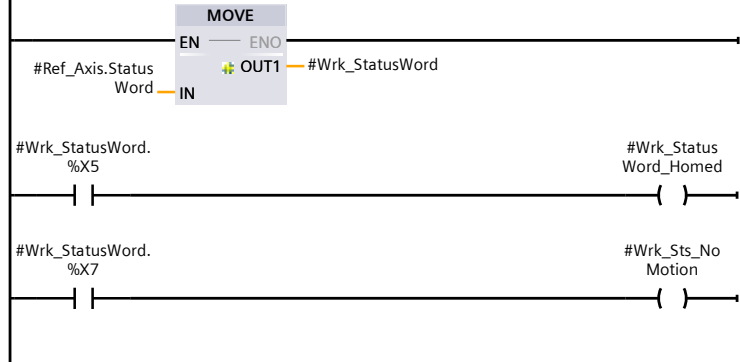
Initialize homing and fault errors



Network 2: Get axis state

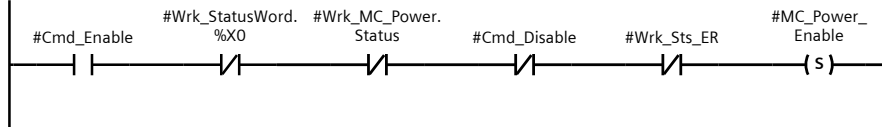
To avoid warning when using outputs as contacts keep some statuses in static area of DB

--	--	--



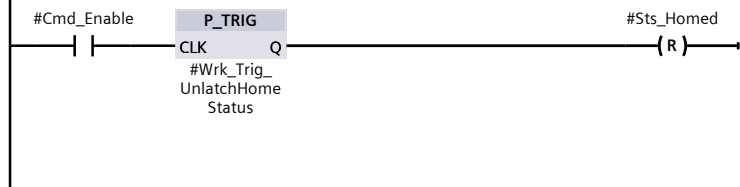
Network 3: - SECTION AXIS ENABLE

Enable axis indicated by result of MC_POWER

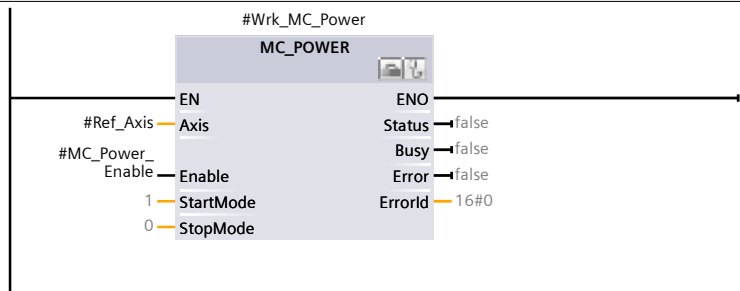


Network 4: Reset homed status

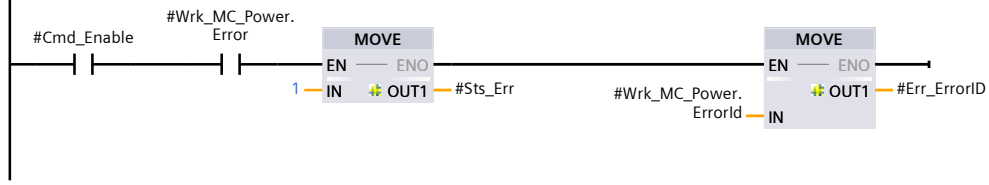
Reset when axis enabled



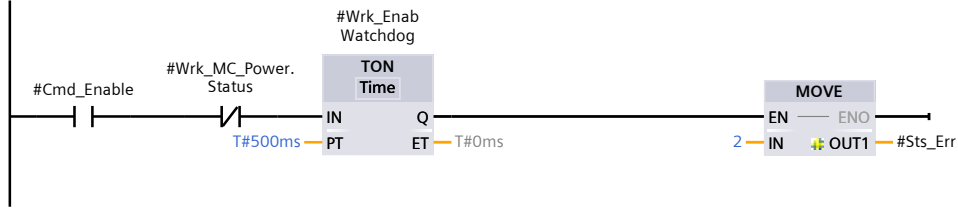
Network 5: MC_Power for axis



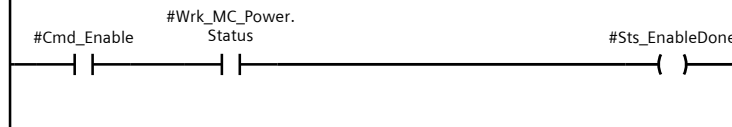
Network 6:



Network 7: Axis enable watchdog

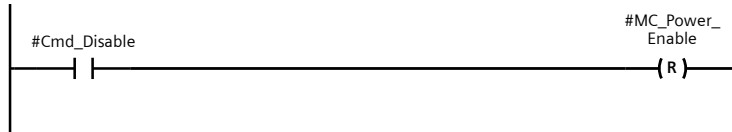


Network 8: Enable status for block output

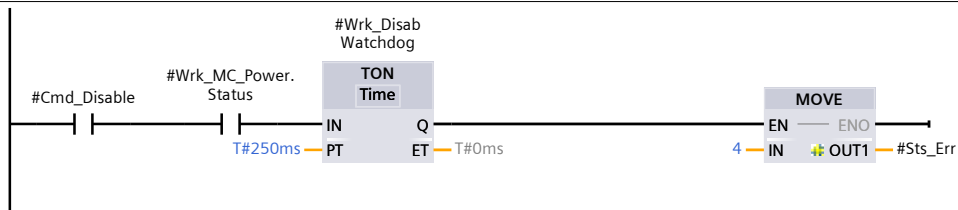


Network 9: - SECTION AXIS DISABLE

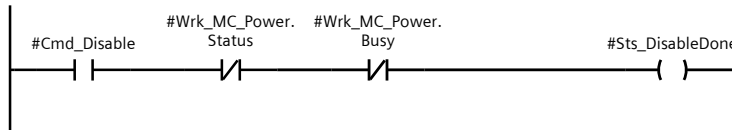
Turn off enable input to MC_POWER block



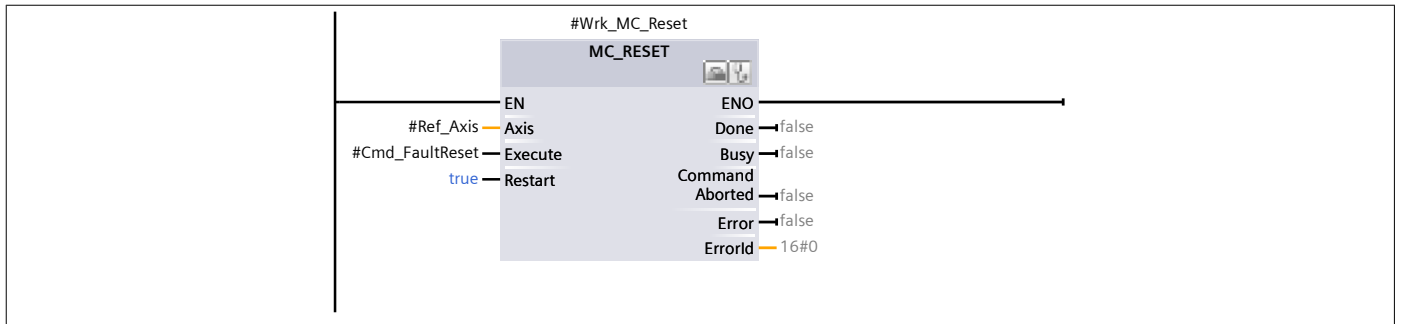
Network 10:



Network 11: Disable status for block output

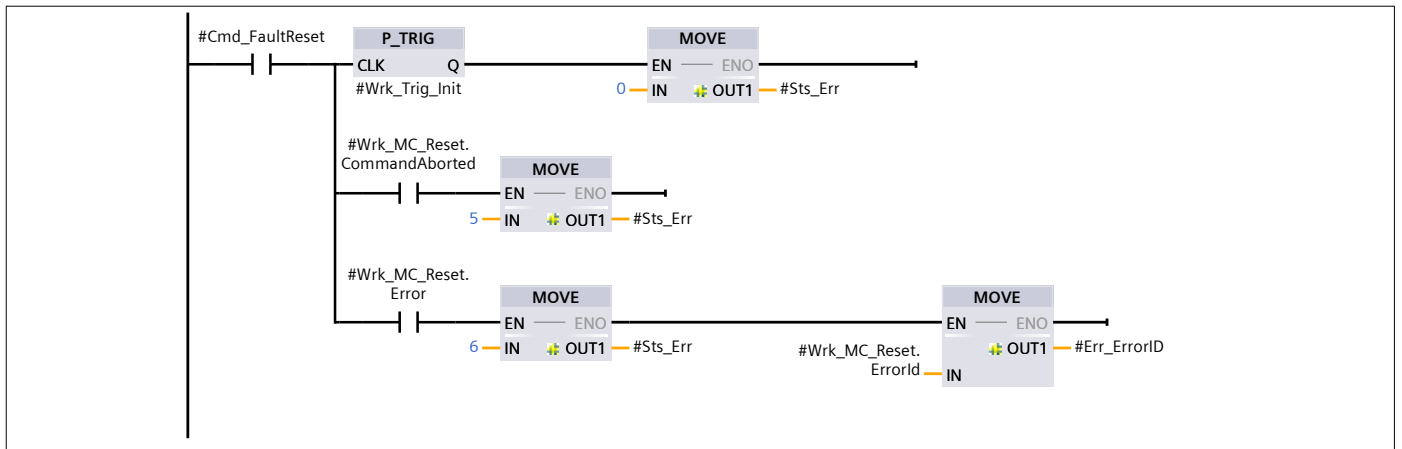


Network 12: - SECTION AXIS FAULT RESET

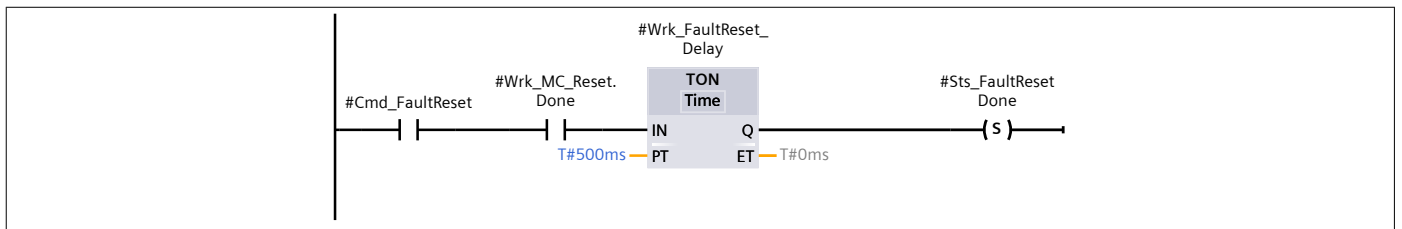


Network 13: Fault clearing indications

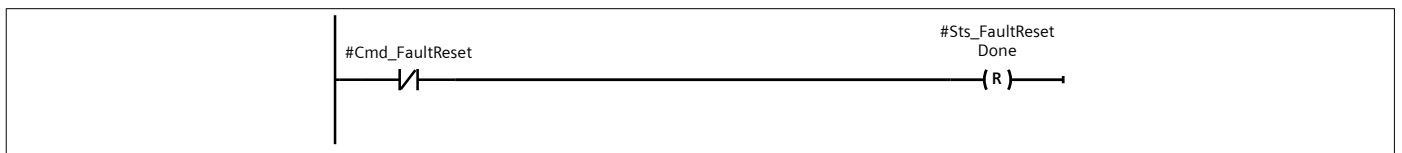
Error status output bits reset on earlier network.
Network reports errors from MC_Reset execution



Network 14: Fault Reset done / handshake

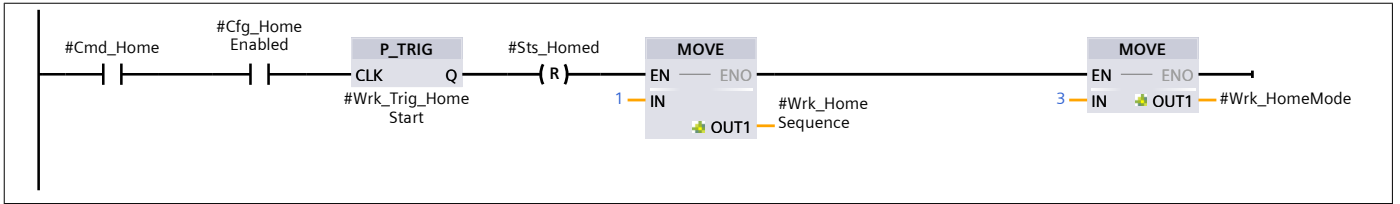


Network 15:

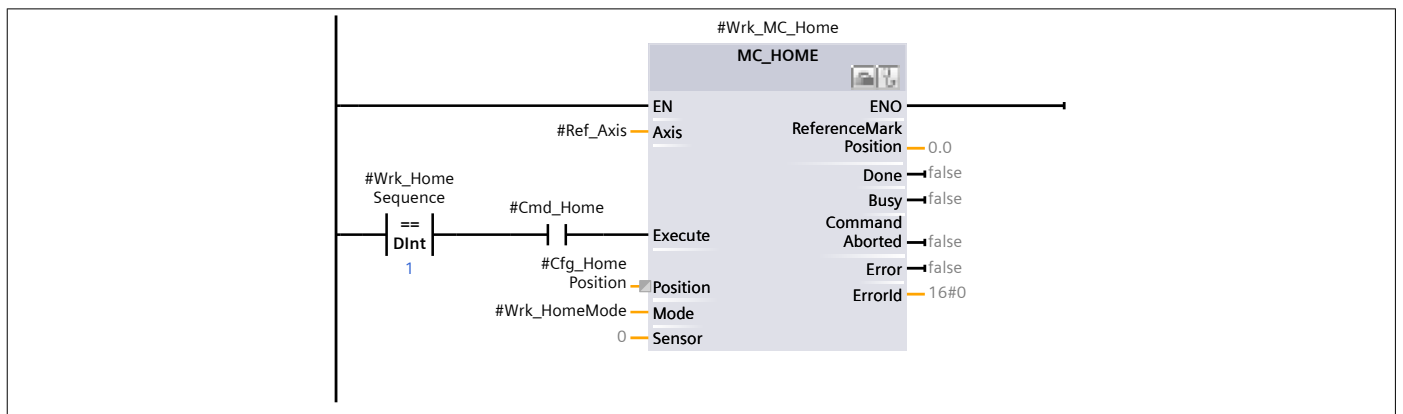


Network 16: - SECTION AXIS HOMING

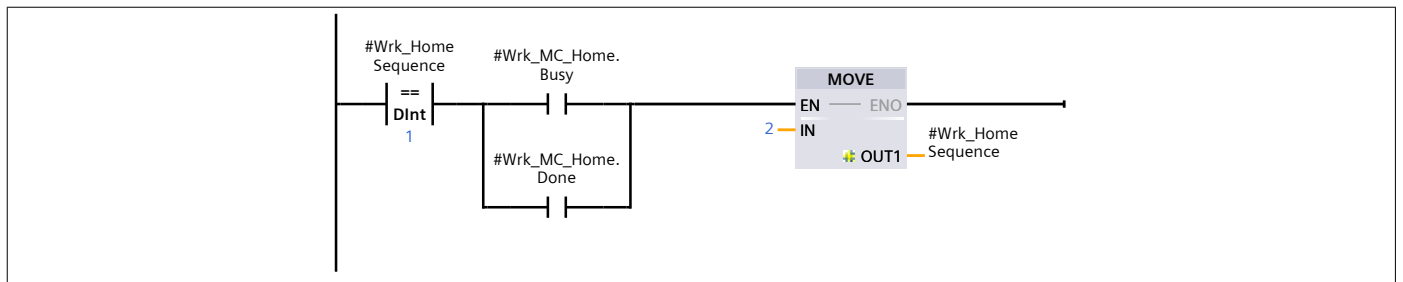
Home command, initiate sequence
Starting with V7.0, Mode=3 works for both incremental and absolute encoders.



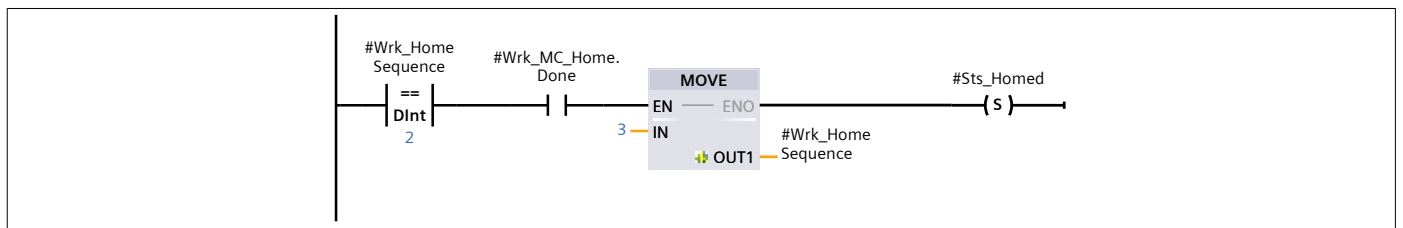
Network 17: Home sequence execution - Step 1 initiates Home command



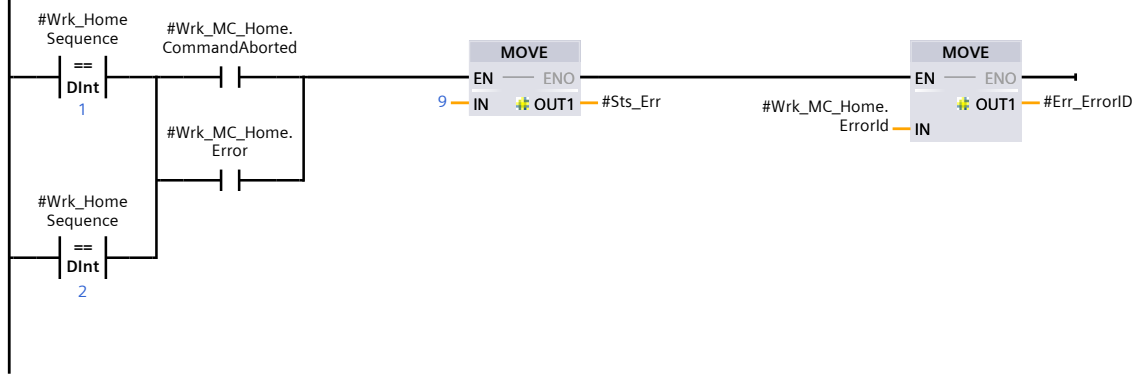
Network 18: Home sequence execution - When busy, go to step 2 and wait for finished



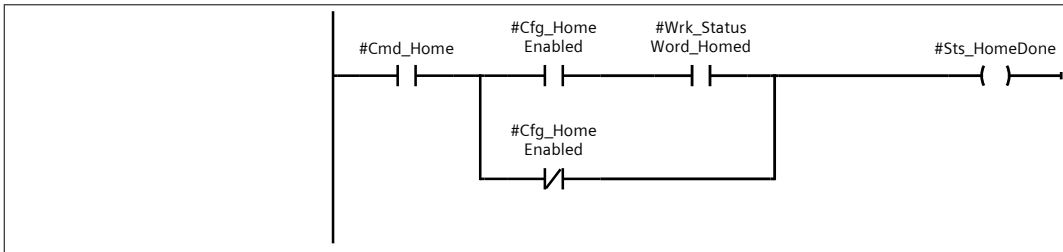
Network 19: Axis is homed



Network 20: Home error handling for MC_HOME



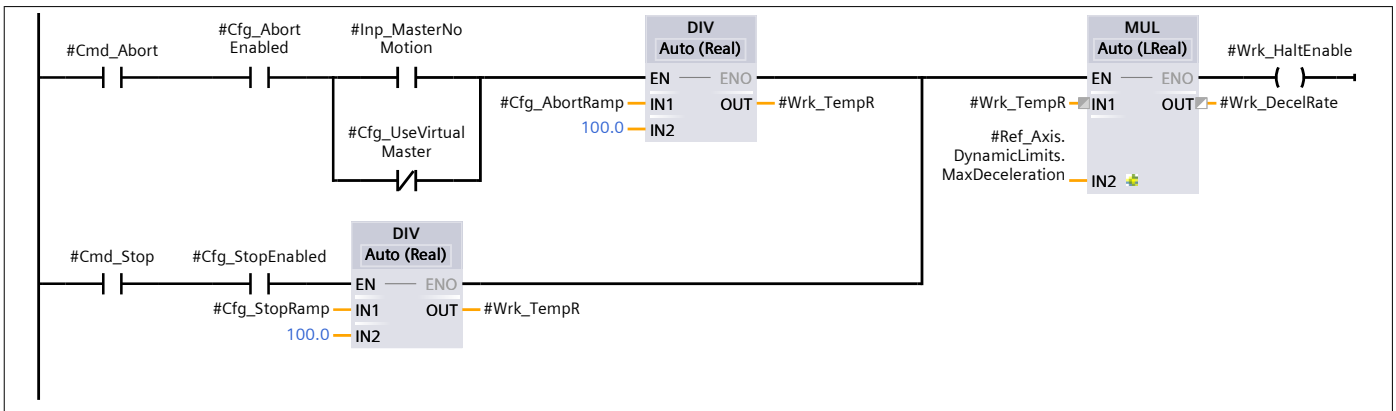
Network 21: Home done handshake



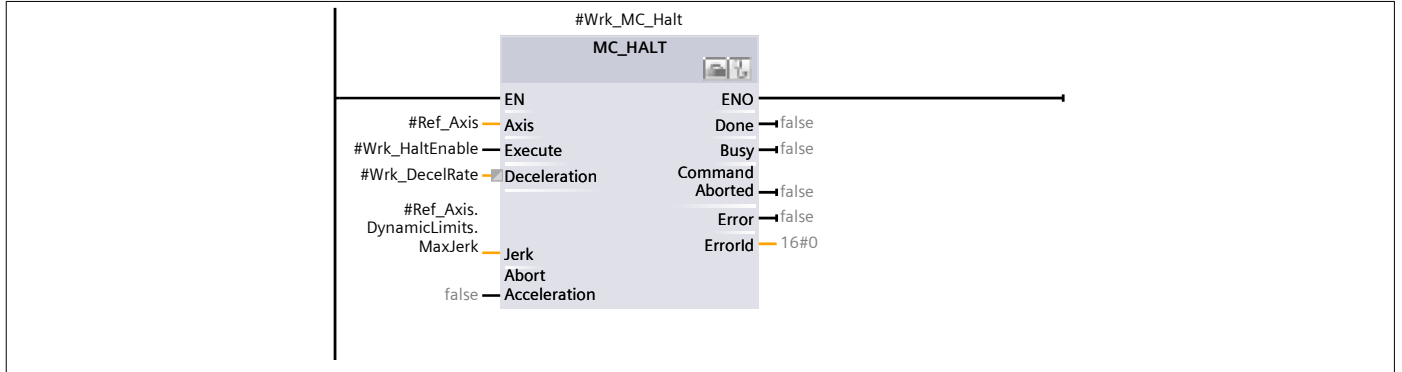
Network 22: - SECTION AXIS STOP

Whenever the machine enters either the stopping or aborting state, stop the axis
 -If stopping, all axes are stopped AFTER the virtual stops, providing a coordinated stop
 -Aborting causes all axes

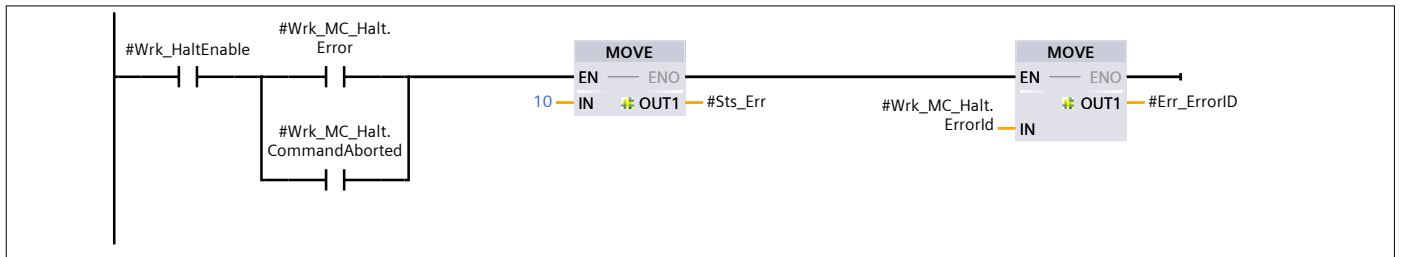
Deceleration as percent of maximum, obtained from axis structure
 Jerk set to max



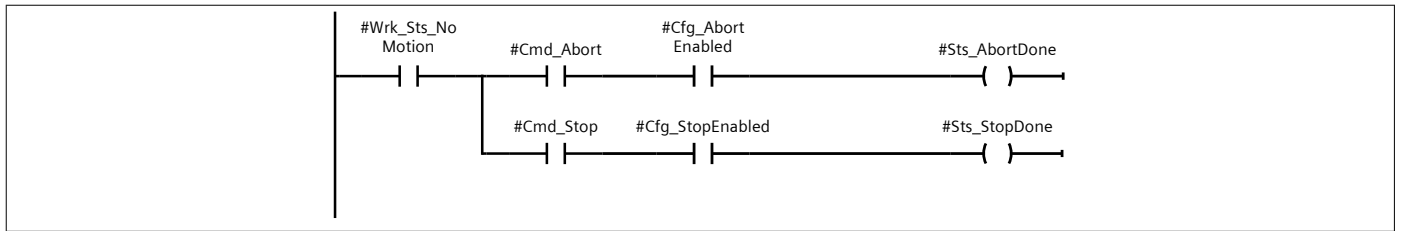
Network 23: MC_Halt block



Network 24:



Network 25: Stop done handshake



Network 26: - SECTION STATUS BIT UPDATE

Check for no axis movement

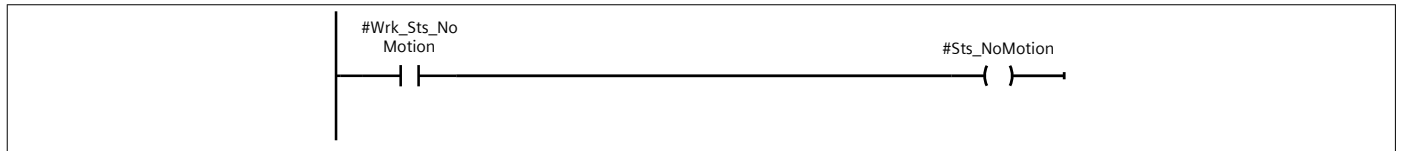
Status word relevant bits - x if relevant to no motion check

- Bit 00 1 if enabled
- Bit 01 1 error present
- Bit 02 1 restart active
- Bit 03 1 restart tags changed
- Bit 04 1 control panel active
- Bit 05 1 axis homed
- Bit 06 x 1 if no motion job running
- Bit 07 x 1 at standstill
- Bit 08 x 0 no positioning job running
- Bit 09 x 0 no jog job running
- Bit 10 x 0 no MoveVelocity running
- Bit 11 x 0 no Home job running
- Bit 12 x 1 axis running at constant velocity or at stanstill
- Bit 13 x 0 No acceleration

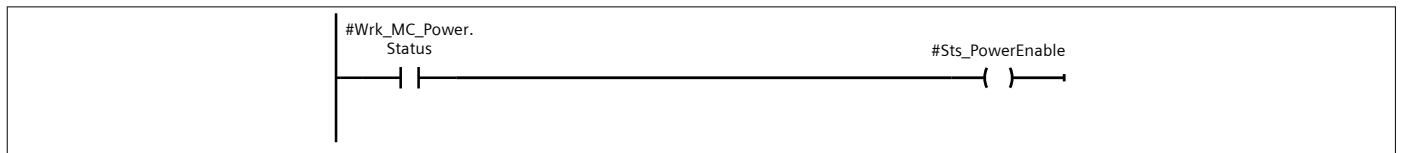
Bit 14 x 0 No deceleration
 Bit 21 x 0 Axis not being synchronized to leading value
 Bit 22 x 0 Axis is not moving synchronously
 Bit 23 x 0 No superimposed motion active
 Bit 24 x 0 No motion control instruction for leading value shift is active

Mask = DW#16#01E07FC0
 Value to check is = DW#16#000010C0

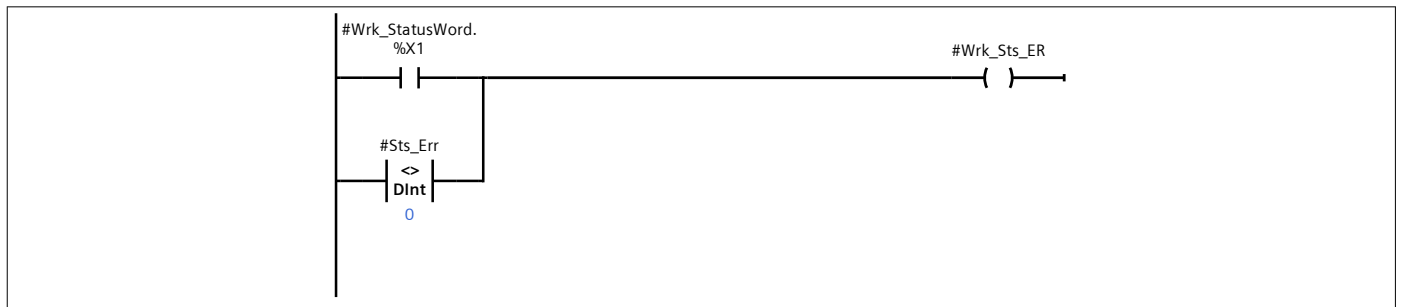
- Err_Sts Values
- 1 = MC_Power error
 - 2 = more than 500 msec for axis to enable
 - 4 = more than 250 msec for axis to disable
 - 5 = MC_Reset command aborted
 - 6 = MC_Reset error
 - 7 = MC_MOVEJOG when homing error
 - 8 = MC_HOME error
 - 9 = MC_HALT error



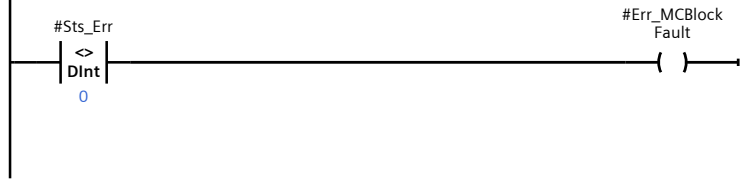
Network 27:



Network 28: General error

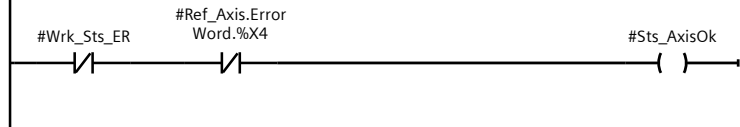


Network 29: Summary of motion block errors

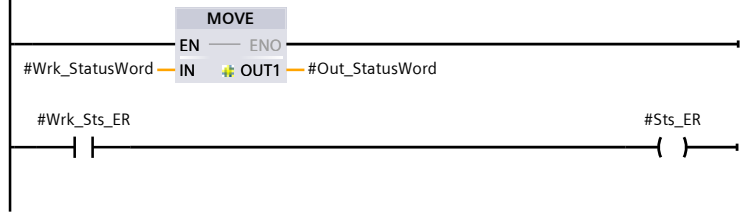


Network 30: Axis is okay and ready

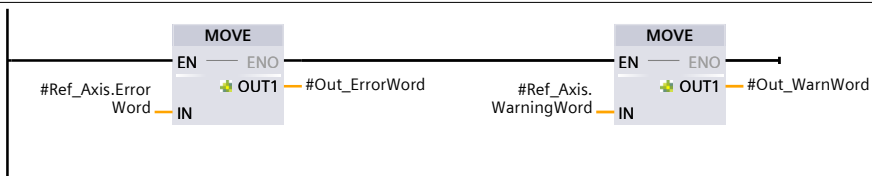
Bit 4 in ErrorWord is drive fault



Network 31: Copy working statuses to block outputs



Network 32: Copy axis error and warnings to outputs



Program blocks / EM02_Axis02_K

EM02_00_Main [FB300]

EM02_00_Main Properties

General

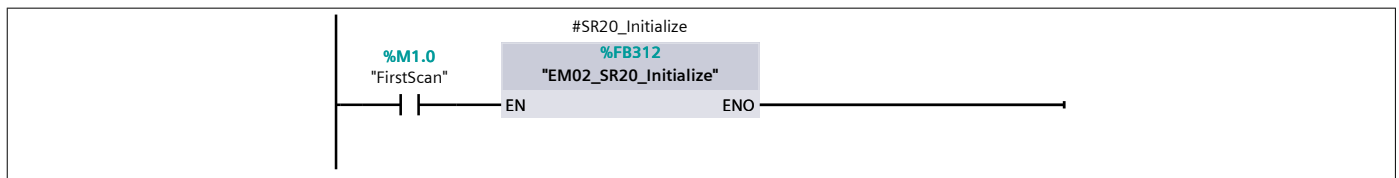
Name	EM02_00_Main	Number	300	Type	FB
Language	LAD	Numbering	Manual		

Information

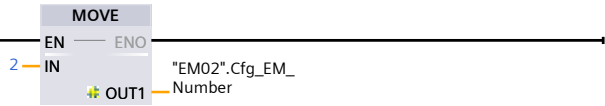
Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Cfg_EM_Number	Int	0	Non-retain
Par_Jog_Accel	Real	0.0	Non-retain
Par_Jog_Decel	Real	0.0	Non-retain
Par_Jog_Speed	Real	0.0	Non-retain
CM00_EM_Procedure	"EM02_CM00_Procedure"		
CM01_EMConditions	"EM02_CM01_EM-Conditions"		
CM02_02_Knife	"EM02_CM02_ServoAxisObject"		
CM03_02_AxisJog_Knife	"EM02_CM03_ServoAxisJog"		
SR03_FaultHandler	"EM02_SR03_Fault Handler"		
SR20_Initialize	"EM02_SR20_Initialize"		
Wrk_Temp_Bit	Bool	false	Non-retain
Temp			
Constant			

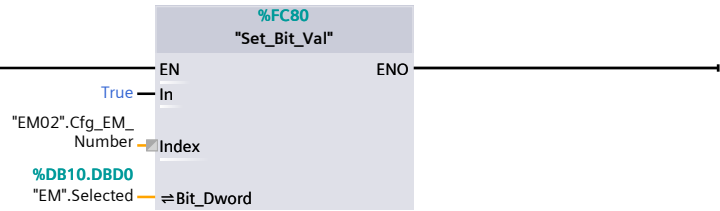
Network 2:



Network 3: Set EM number



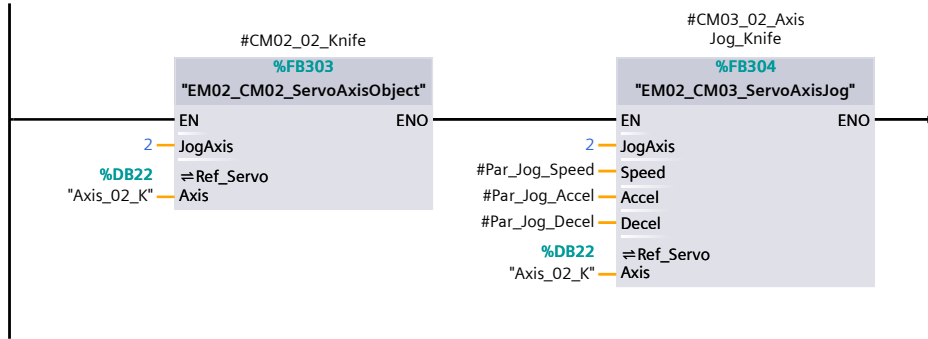
Network 4: Equipment module is selected and active



Network 5:



Network 6:



Network 7:

#CM01_EM
Conditions

#SR03_Fault
Handler

%FB302

%FB311

"EM02_CM01_EMConditions"

"EM02_SR03_FaultHandler"

EN

ENO

EN

ENO

Program blocks / EM02_Axis02_K

EM02_CM00_Procedure [FB301]

EM02_CM00_Procedure Properties

General

Name	EM02_CM00_Procedure	Number	301	Type	FB
Language	LAD	Numbering	Manual		

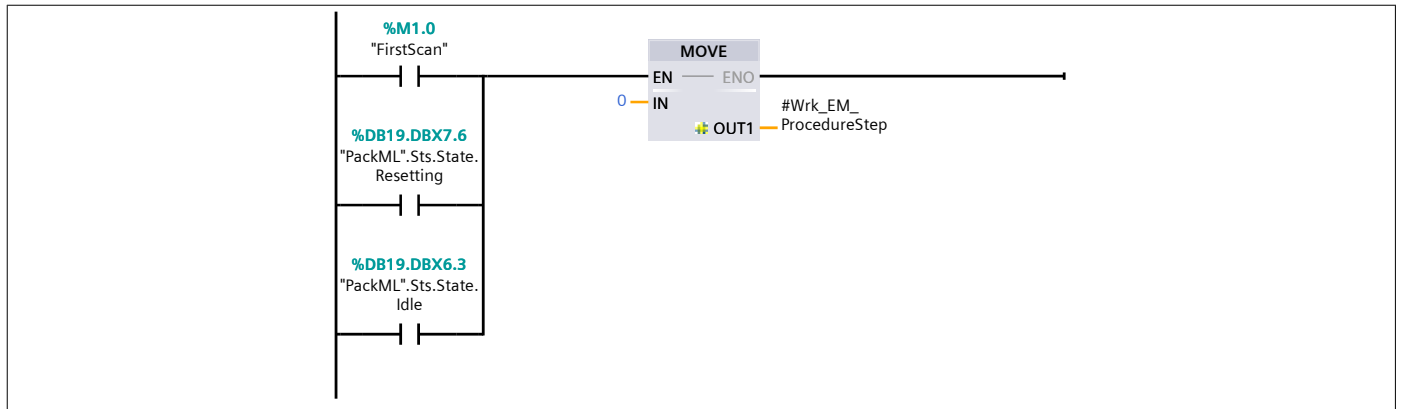
Information

Title	EM Procedure Sequence	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
ErrorID	Word	16#0	Non-retain
Sts_Err	Int	0	Non-retain
Wrk_EM_ProcedureStep	DInt	0	Non-retain
Wrk_EM_ProcedureNoStepsActive	Bool	false	Non-retain
Wrk_EM_ProcedureDone	Bool	false	Non-retain
Wrk_EM_ProcedureFault	Bool	false	Non-retain
Wrk_MC_InterpCam	MC_INTERPOLATE-CAM		
Wrk_MC_MoveVel	MC_MOVEVELOCITY		
Wrk_MC_Halt	MC_HALT		
Wrk_MC_Home	MC_HOME		
Wrk_MC_MeasuringIn	MC_MEASURINGINPUT		
Wrk_MC_CamIn	MC_CAMIN		
Wrk_CamIn_MasterOffset	LReal	0.0	Non-retain
Knife_Homed	Bool	false	Non-retain
End_Of_Profile	Bool	false	Non-retain
Wrk_Trans_InterpCam	Bool	false	Non-retain
Wrk_Trans_MoveVel	Bool	false	Non-retain
Wrk_Trans_Halt	Bool	false	Non-retain
Wrk_Trans_Home	Bool	false	Non-retain
Wrk_Trans_CamIn	Bool	false	Non-retain
Wrk_Trans_MeasIn	Bool	false	Non-retain
Web_RegistrationOffset	LReal	0.0	Non-retain
Saved_Regist_Position	LReal	0.0	Non-retain
Wrk_Push	Bool	false	Non-retain
MC_Meas_Error	Word	16#0	Non-retain
Wrk_MC_AbortMeas	MC_ABORTMEASURINGINPUT		
Temp			
Constant			

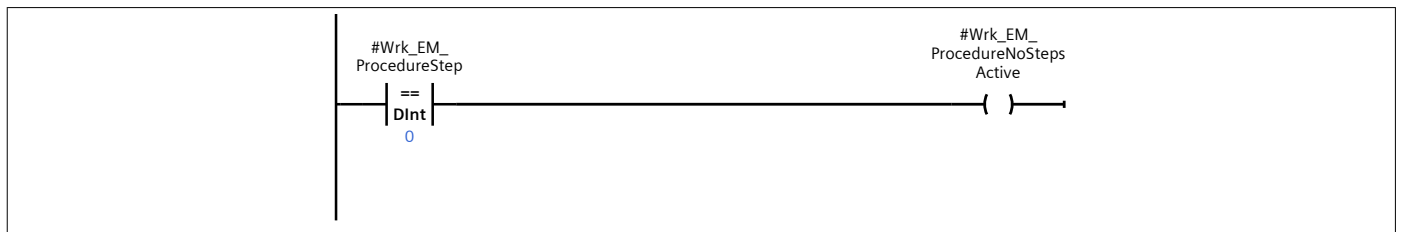
Network 1: Reset step number

First scan or resetting or idle - clear all steps



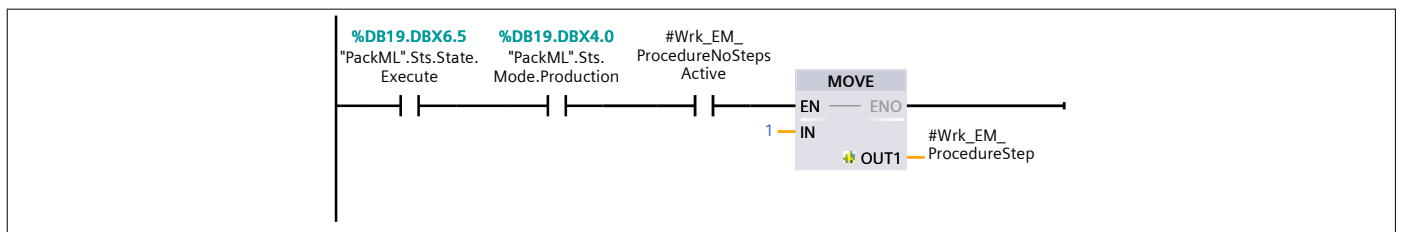
Network 2: Indication that all steps reset

Used by RESETTING state and initial execute step

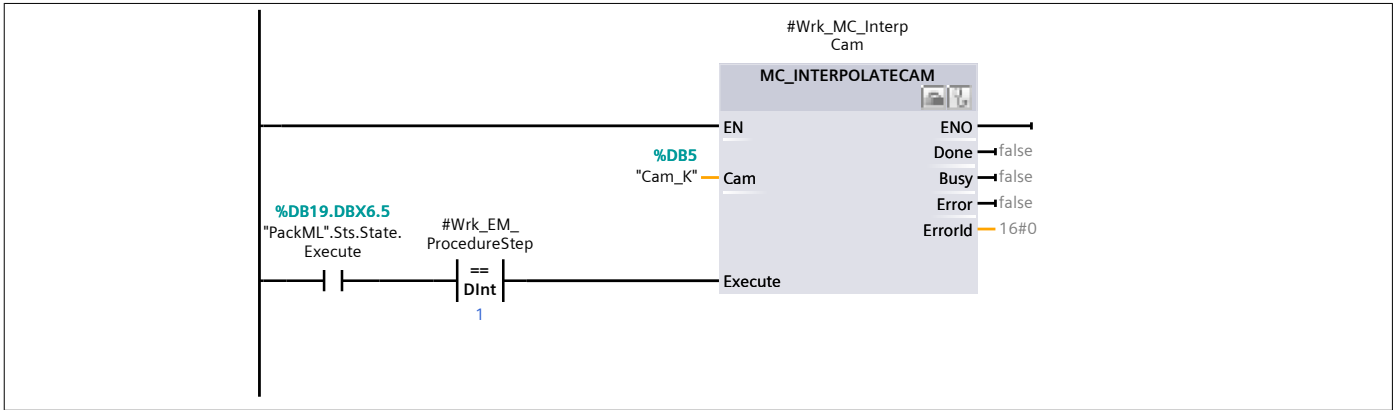


Network 3: EXECUTE -

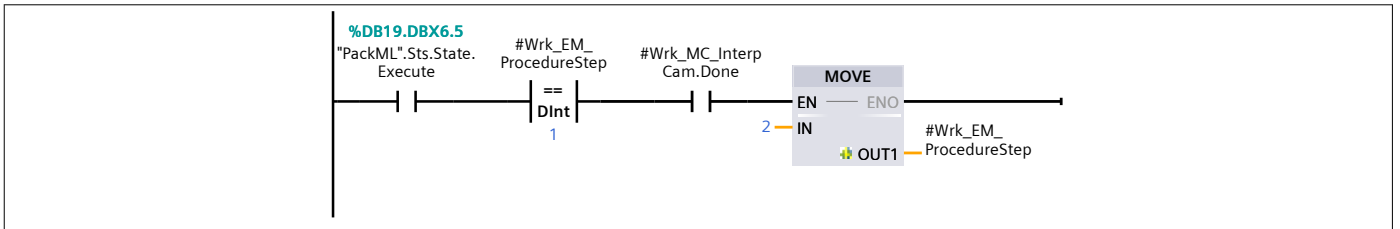
If no steps active go to step 1



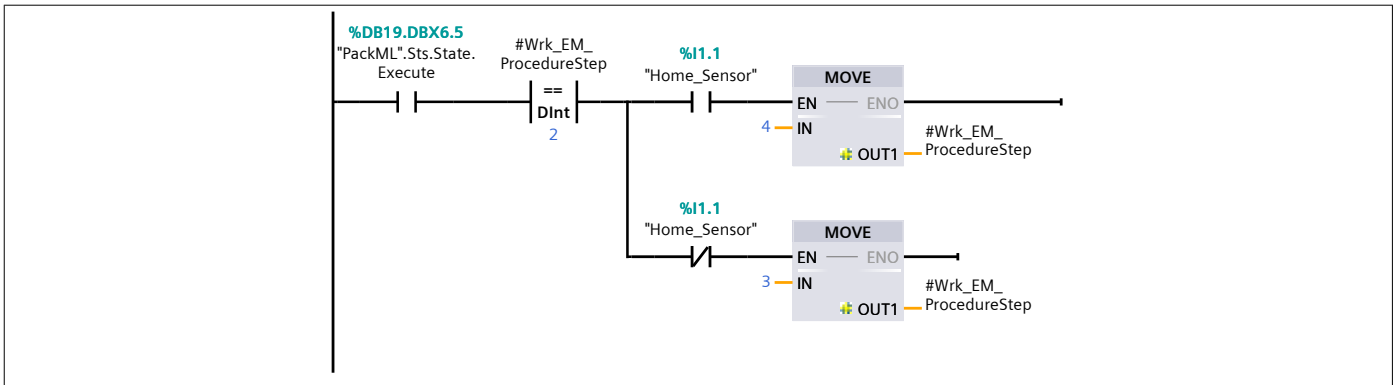
Network 4: Step 1 - Generate Cam



Network 5: Step 1 transition when interpolation done

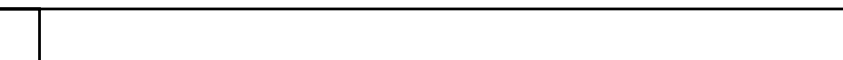


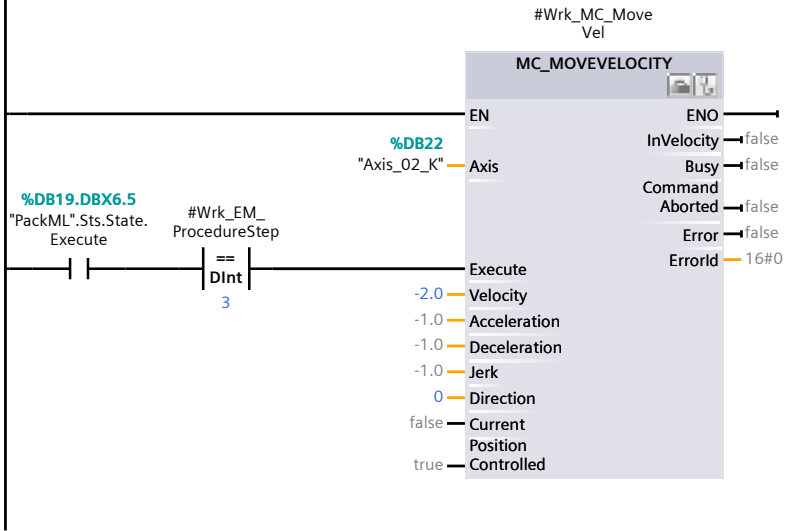
Network 6: Step 2 - Check for at home. If at home sensor, skip to halt axis



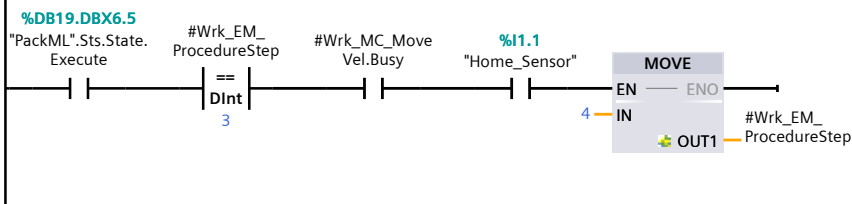
Network 7: Step 3 - Move knife until get to home sensor

Home velocity of 2 rad/sec will not overshoot home sensor

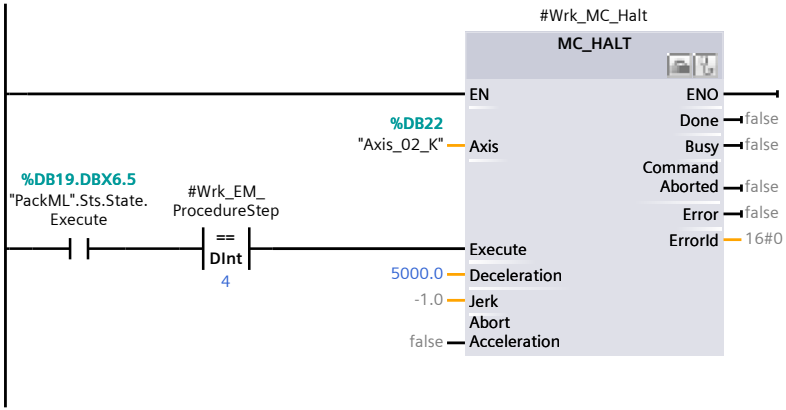




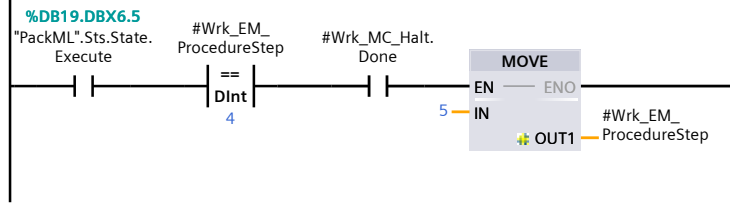
Network 8: Wait for home sensor to come on



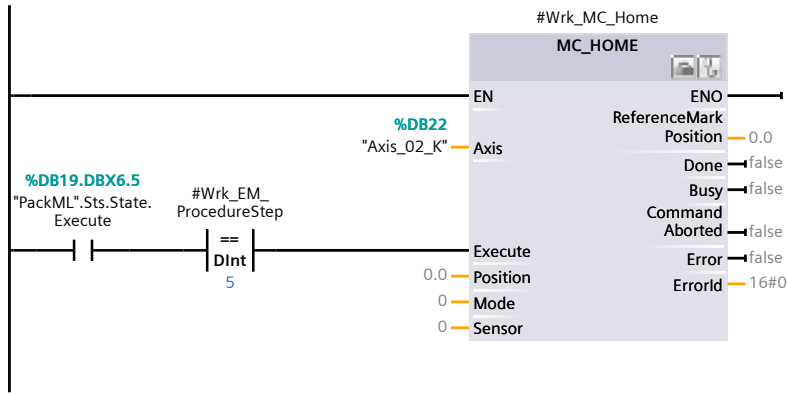
Network 9: Step 4 - Stop with quick decel



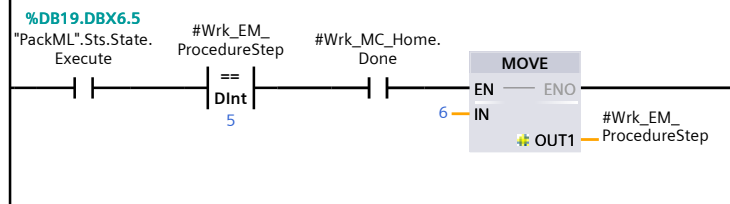
Network 10: Transition when halt done



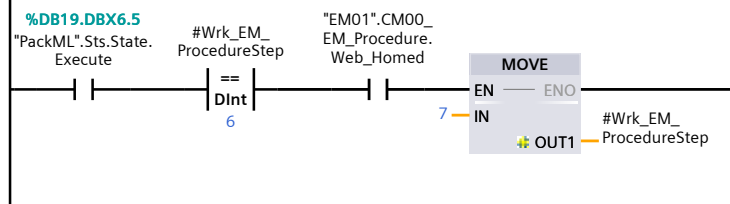
Network 11: Step 5 - Home axis



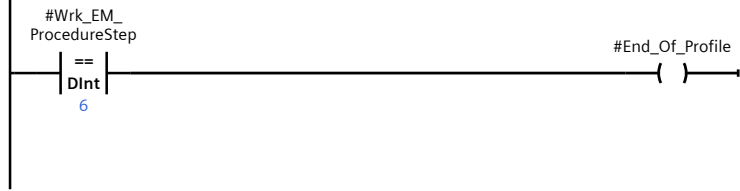
Network 12: Step 5 - Home axis



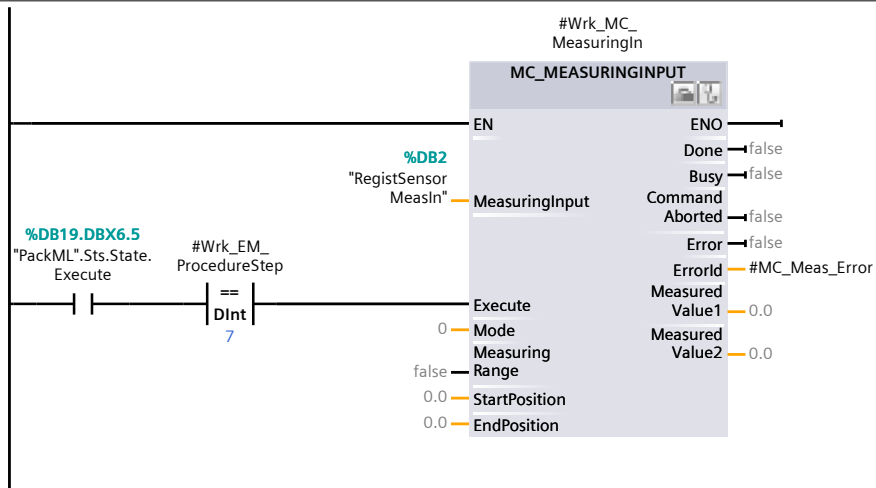
Network 13: Step 6 - Wait for web axis homed.



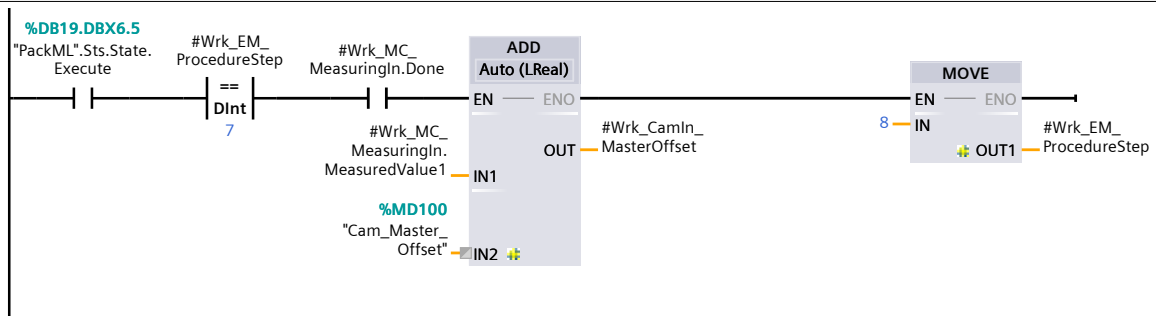
Network 14: Set end of profile indication on for web axis



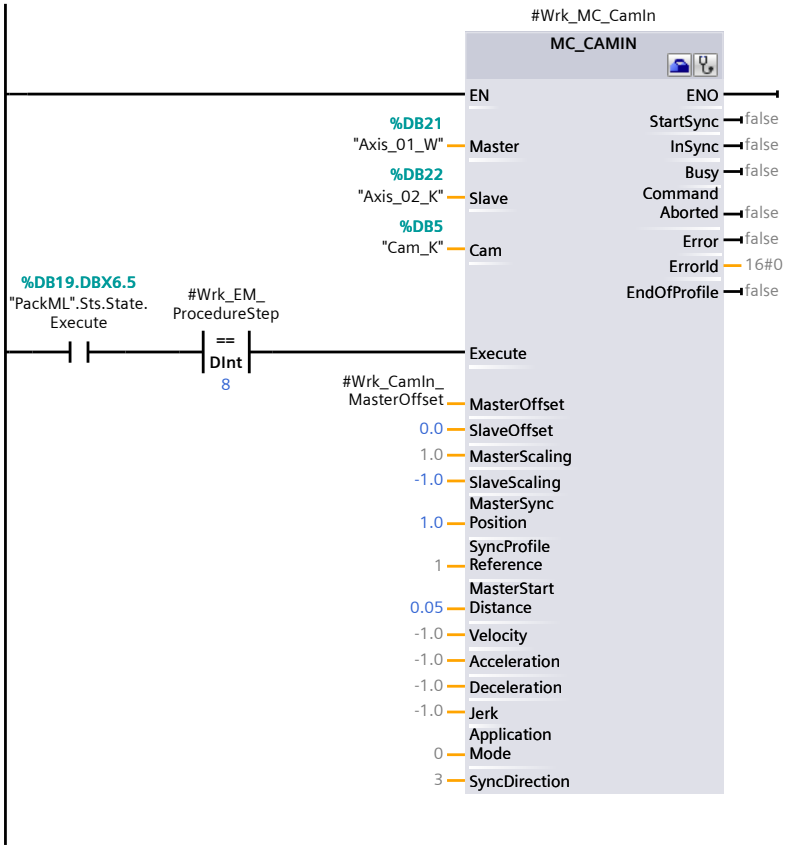
Network 15: Step 7 - Wait for registration sensor



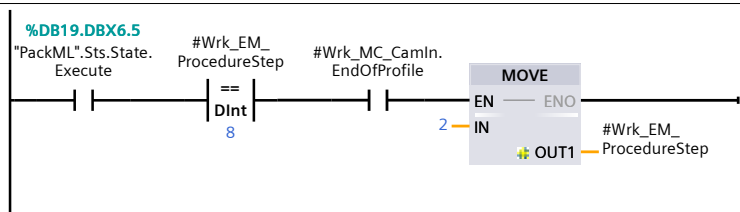
Network 16: Transition when registration sensed. Calculate master offset for cam



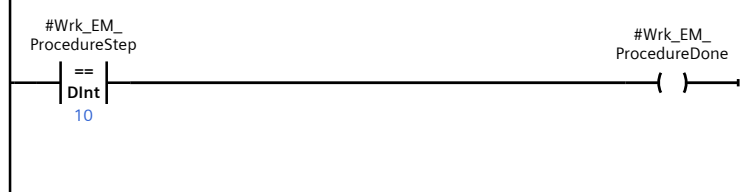
Network 17: Step 8 - Start cam



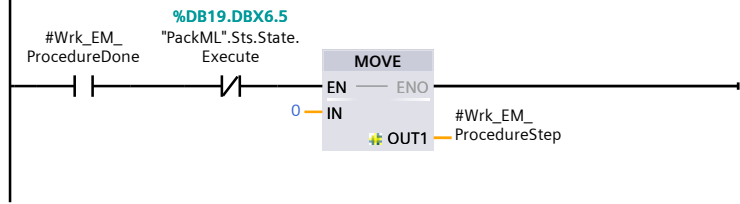
Network 18: Transition to next step when at end of profile



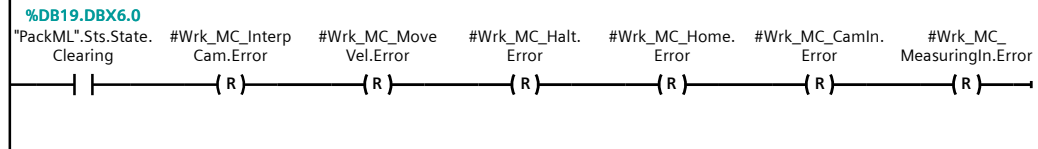
Network 19: Last step - set done



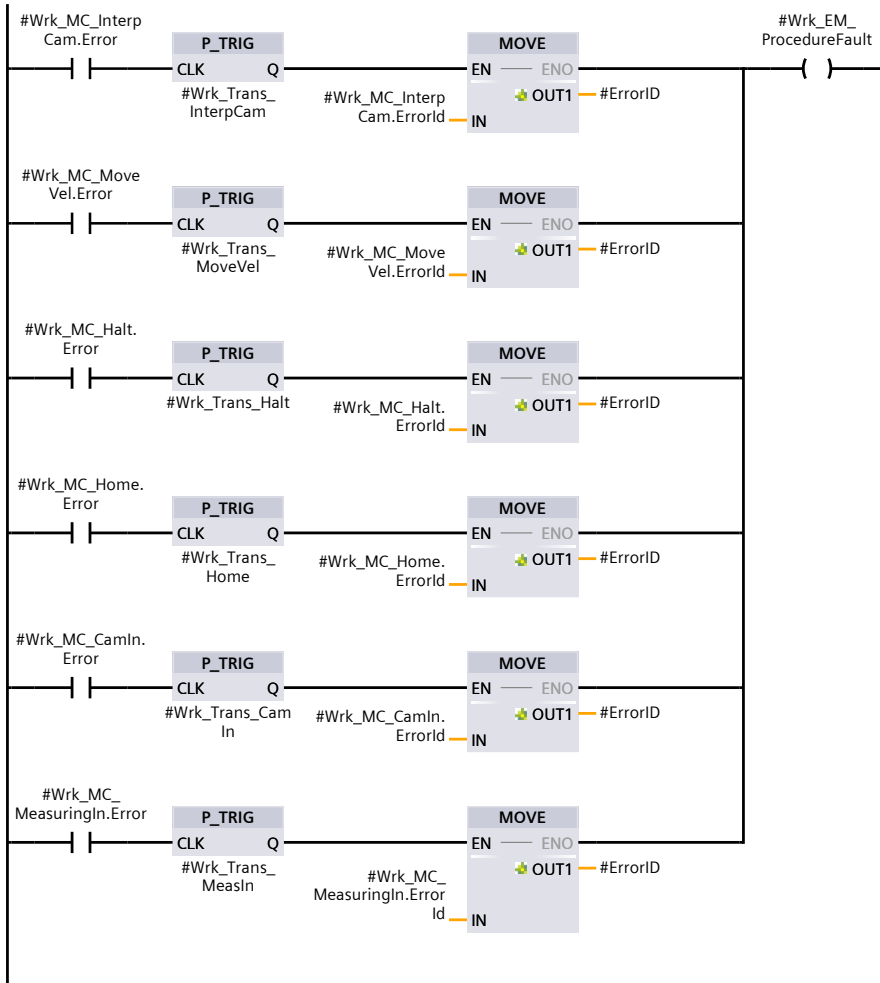
Network 20: When done, wait for out of execute step, then set step back to zero so can restart without resetting



Network 21: Clearing state resets any errors

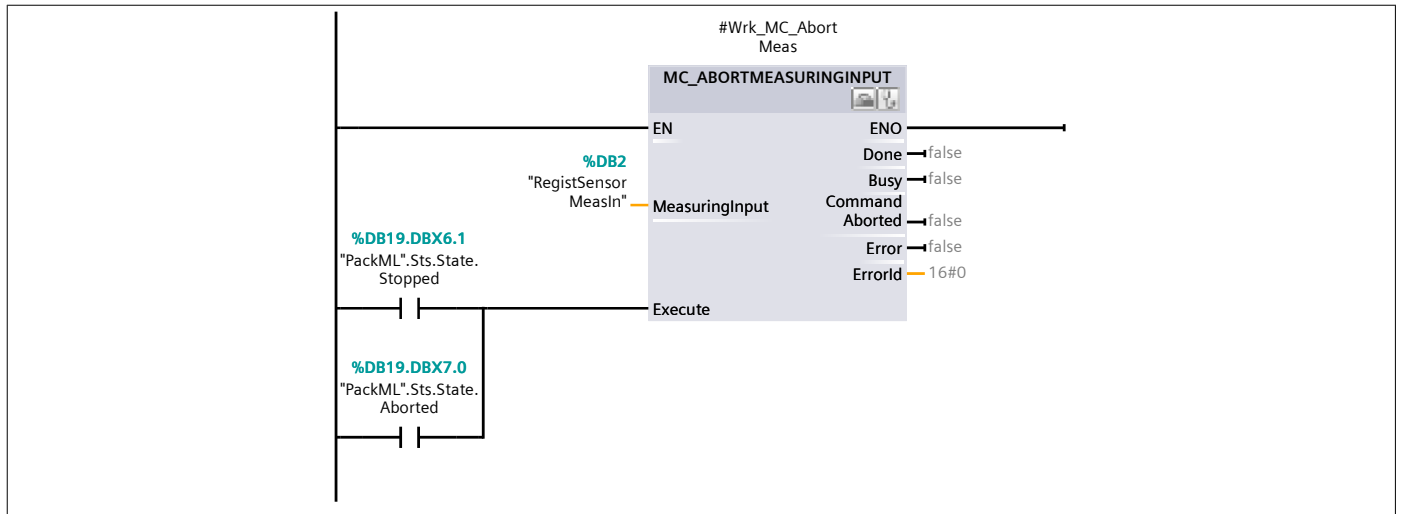


Network 22: Put any error handling here

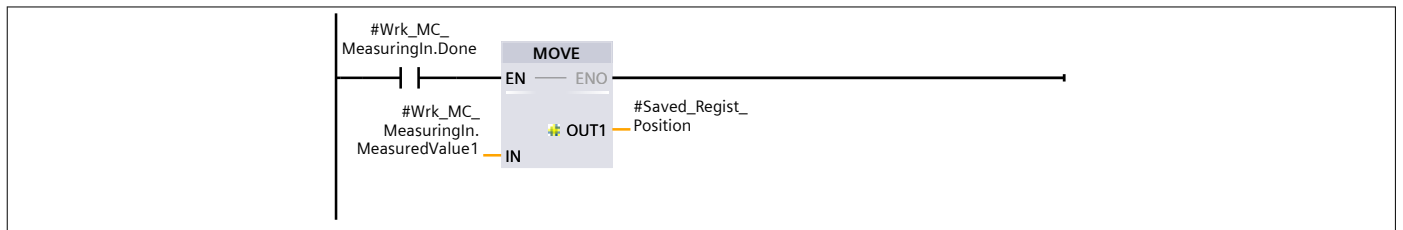


Network 23: If stopped or aborted, cancel measurement

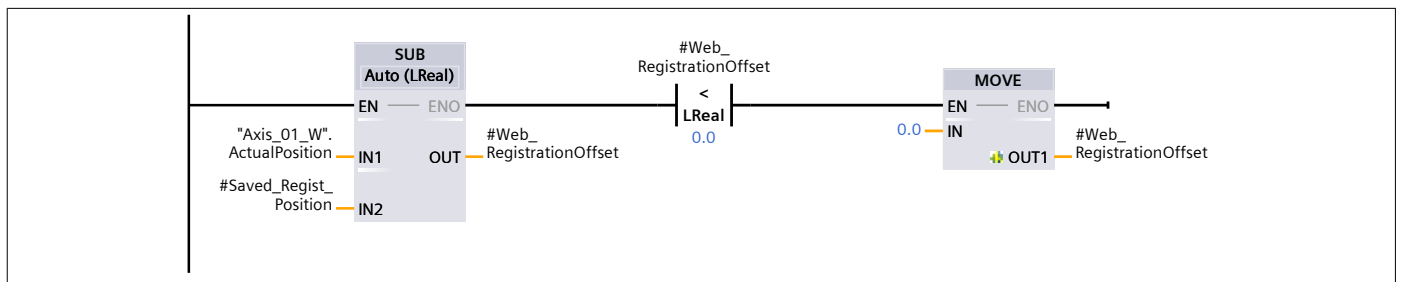
If not cancelled, the only way to clear the error is to stop the processor and restart it.



Network 24:



Network 25: Track Web Registration position offset for trace



Program blocks / EM02_Axis02_K

EM02_CM02_ServoAxisObject [FB303]

EM02_CM02_ServoAxisObject Properties

General

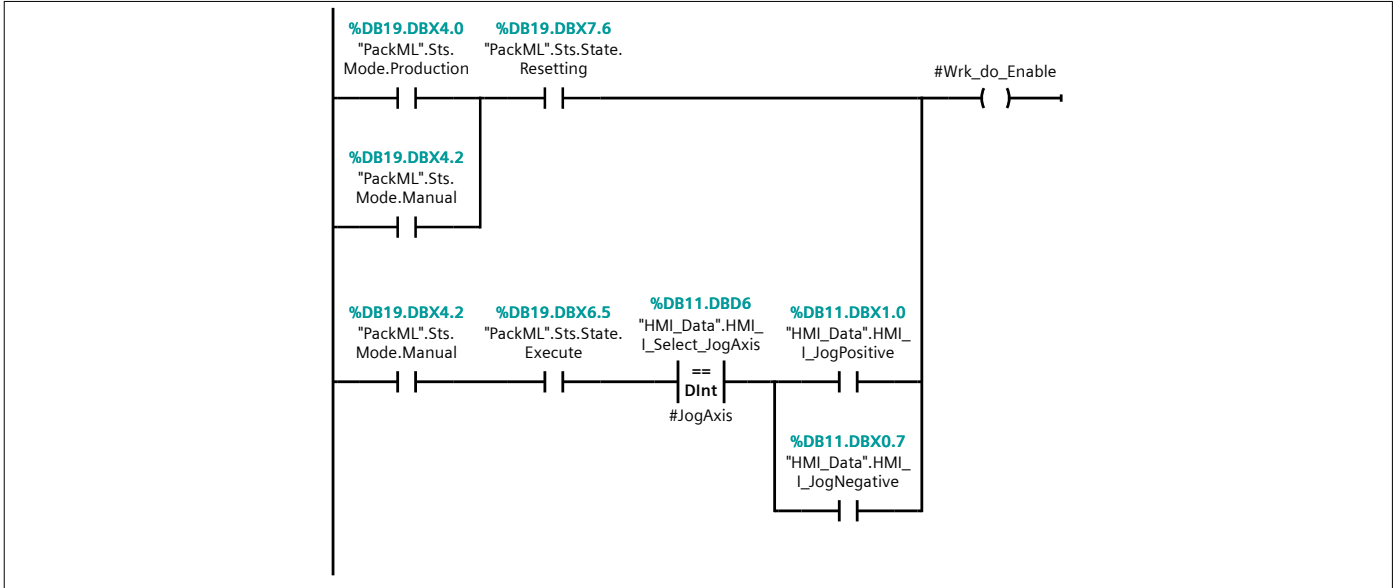
Name	EM02_CM02_ServoAxisObject	Number	303	Type	FB
Language	LAD	Numbering	Manual		

Information

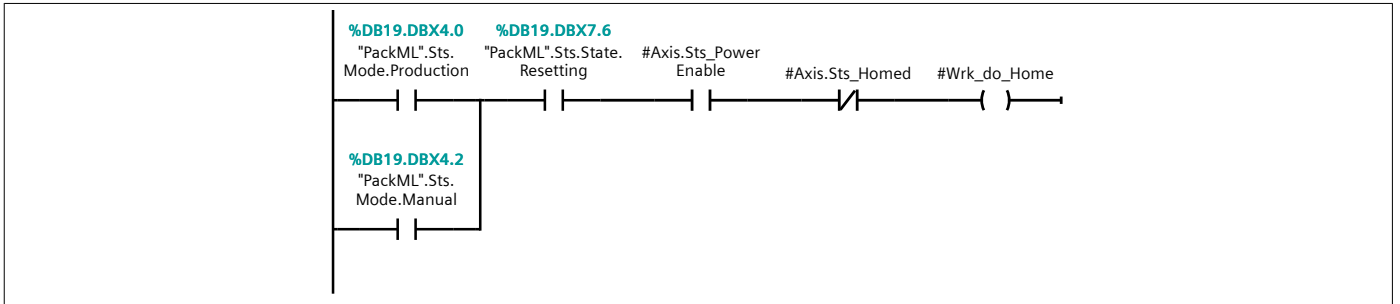
Title	Servo Axis object for Knife axis	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
JogAxis	DInt	0	Non-retain
Output			
▼ InOut			
Ref_ServoAxis	TO_PositioningAxis		
▼ Static			
Axis	"Axis_ObjectPOS"		
Wrk_DisableDelay	TON_TIME		Non-retain
Wrk_DisableDelay_IN	Bool	false	Non-retain
Wrk_do_Enable	Bool	false	Non-retain
Wrk_do_Home	Bool	false	Non-retain
Wrk_do_Stop	Bool	false	Non-retain
Wrk_do_Disable	Bool	false	Non-retain
Wrk_Temp_Bit	Bool	false	Non-retain
Temp			
Constant			

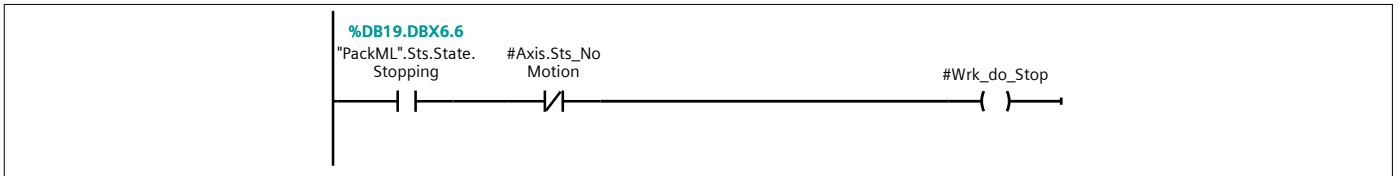
Network 1: Do_Enable logic



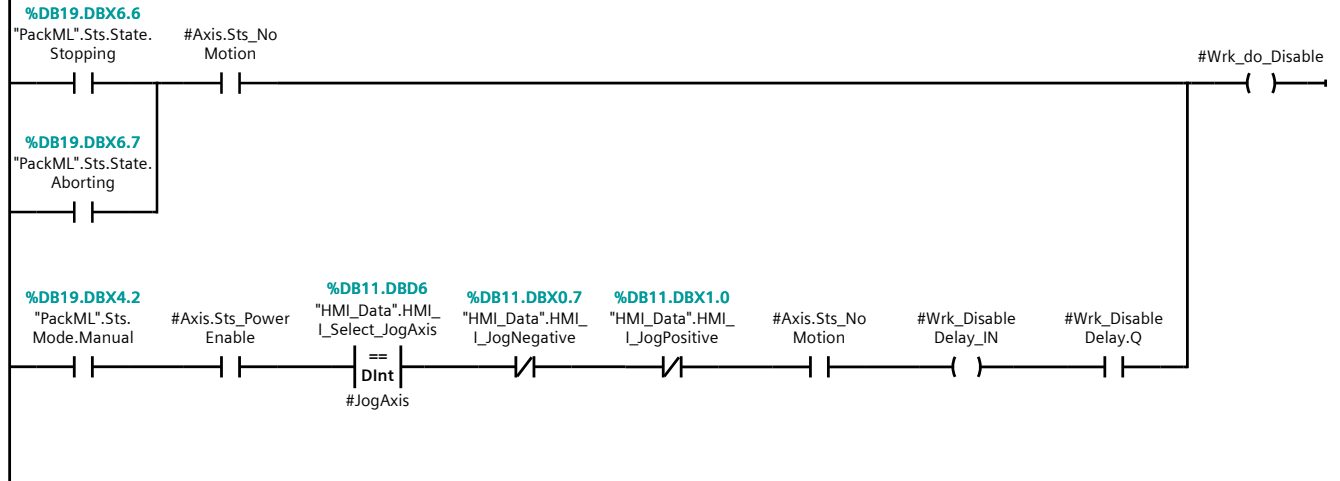
Network 2: Do_Home logic



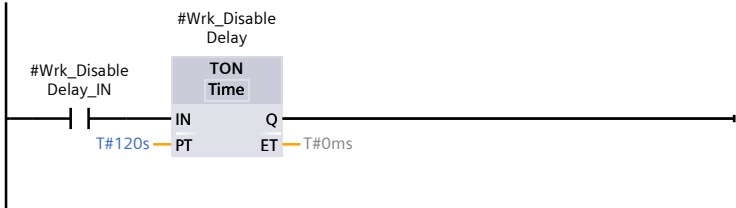
Network 3: Do_Stop logic



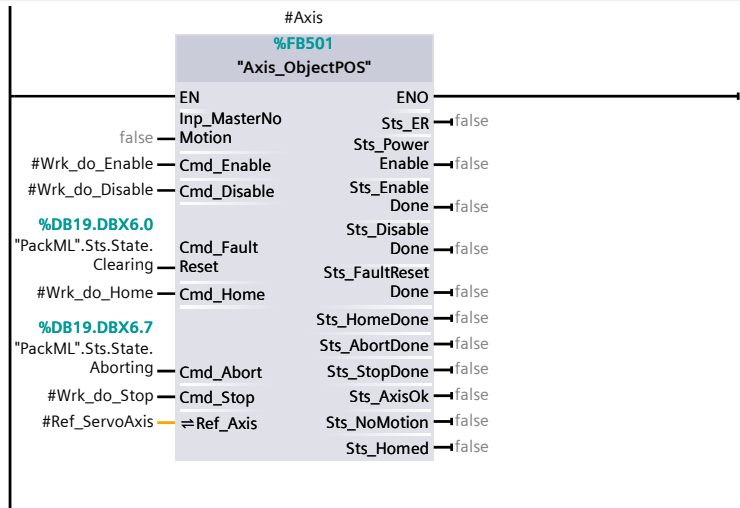
Network 4: Do_Disable logic



Network 5: Disable delay timer for manual mode - if no jogging for 2 min, disable axis



Network 6:



Program blocks / EM02_Axis02_K

EM02_SR20_Initialize [FB312]

EM02_SR20_Initialize Properties

General

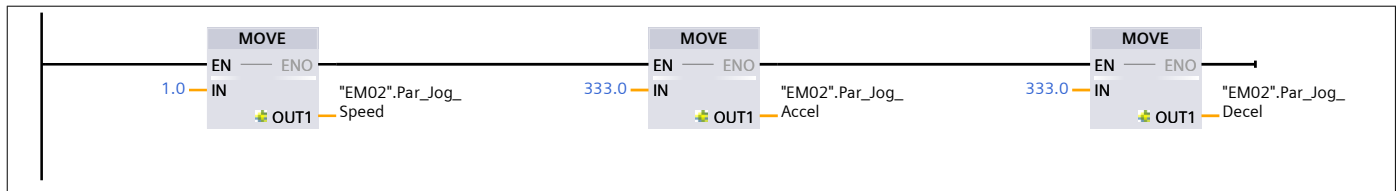
Name	EM02_SR20_Initialize	Number	312	Type	FB
Language	LAD	Numbering	Manual		

Information

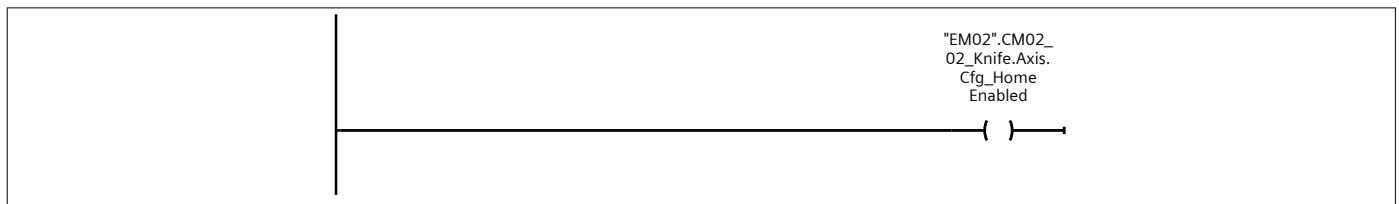
Title	Initialization for EM	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Temp_Dint	DInt	0	Non-retain
Temp			
Constant			

Network 2:



Network 3: Home enabled



Program blocks / EM02_Axis02_K

EM02 [DB301]

EM02 Properties

General

Name	EM02	Number	301	Type	DB
Language	DB	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Start value	Retain
Input			
Output			
InOut			
▼ Static			
Cfg_EM_Number	Int	0	False
Par_Jog_Accel	Real	0.0	False
Par_Jog_Decel	Real	0.0	False
Par_Jog_Speed	Real	0.0	False
CM00_EM_Procedure	"EM02_CM00_Procedure"		False
CM01_EMConditions	"EM02_CM01_EMConditions"		False
CM02_02_Knife	"EM02_CM02_ServoAxisObject"		False
CM03_02_AxisJog_Knife	"EM02_CM03_ServoAxisJog"		False
SR03_FaultHandler	"EM02_SR03_FaultHandler"		False
SR20_Initialize	"EM02_SR20_Initialize"		False
Wrk_Temp_Bit	Bool	false	False

Program blocks / EM02_Axis02_K

EM02_CM03_ServoAxisJog [FB304]

EM02_CM03_ServoAxisJog Properties

General

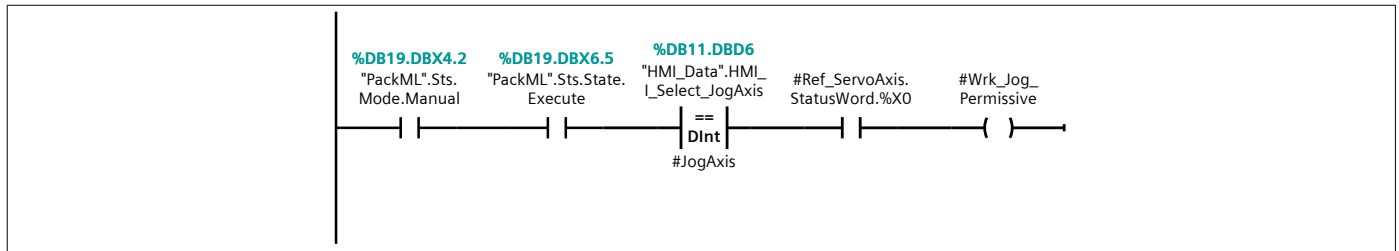
Name	EM02_CM03_ServoAxis-Jog	Number	304	Type	FB
Language	LAD	Numbering	Manual		

Information

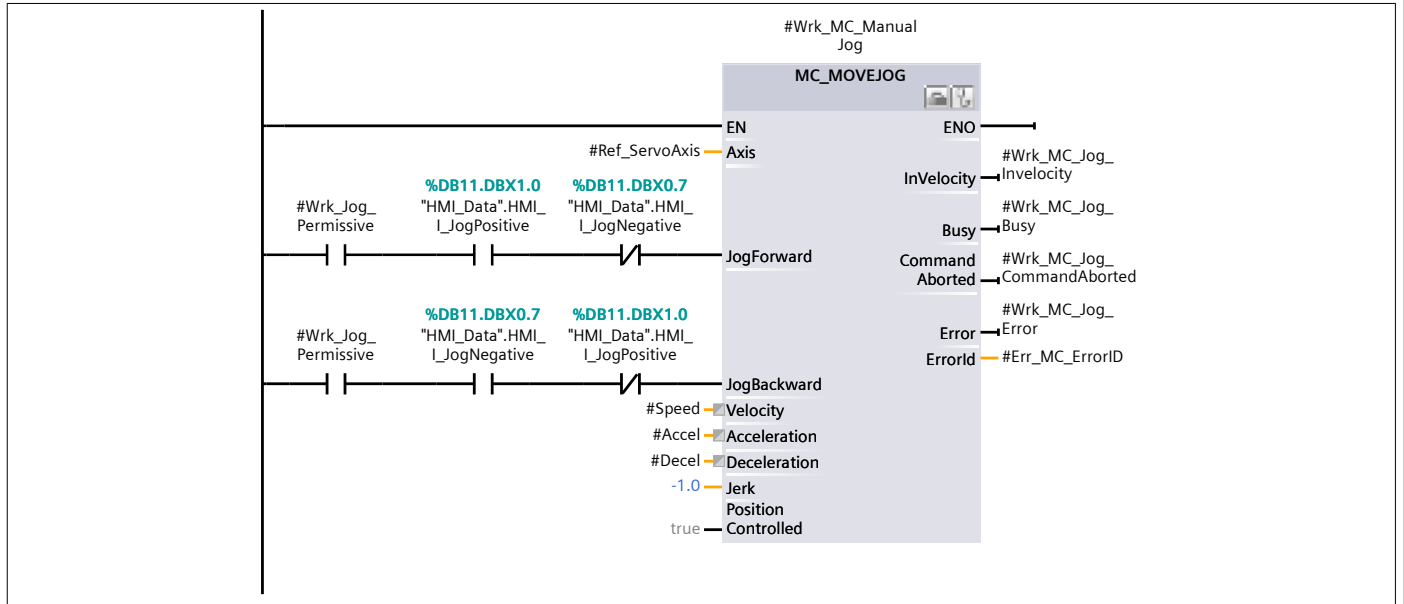
Title	Jog servo axis in manual mode	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
▼ Input			
JogAxis	DInt	0	Non-retain
Speed	Real	0.0	Non-retain
Accel	Real	0.0	Non-retain
Decel	Real	0.0	Non-retain
Output			
▼ InOut			
Ref_ServoAxis	TO_PositioningAxis		
▼ Static			
Sts_JogFault	Bool	false	Non-retain
Err_MC_ErrorID	Word	16#0	Non-retain
Wrk_Jog_Permissive	Bool	false	Non-retain
Wrk_MC_ManualJog	MC_MOVEJOG		
Wrk_MC_Jog_Invelocity	Bool	false	Non-retain
Wrk_MC_Jog_Busy	Bool	false	Non-retain
Wrk_MC_Jog_CommandAborted	Bool	false	Non-retain
Wrk_MC_Jog_Error	Bool	false	Non-retain
Temp			
Constant			

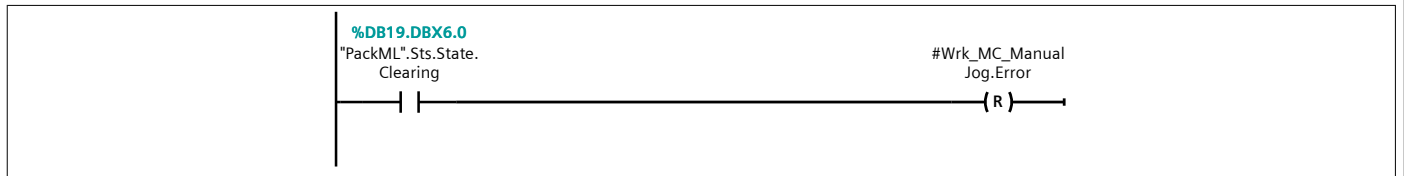
Network 2: RELEASE JOG FUNCTION (-> SELECT EM NUMBER VIA HMI)



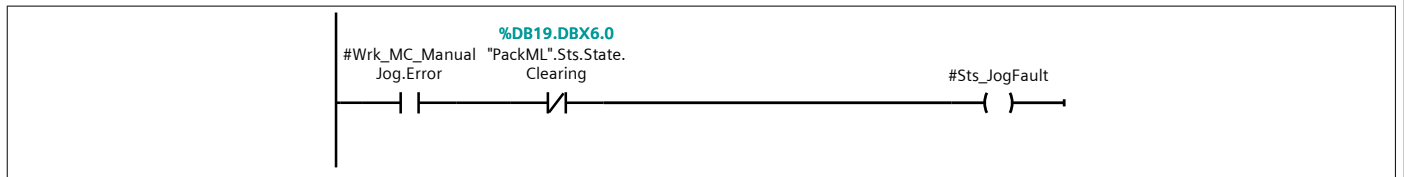
Network 3: JOG SERVO AXIS



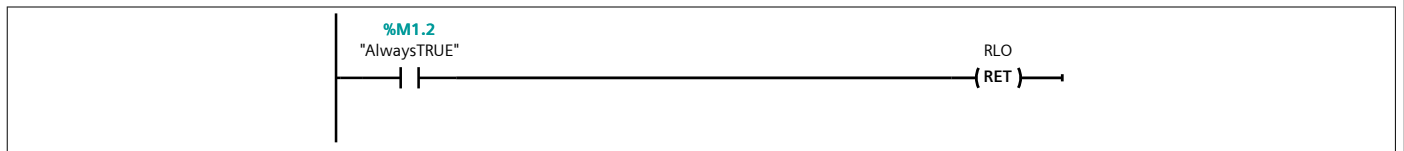
Network 4: Fault Reset



Network 5:



Network 6: END: Update the ENO Output. (DO NOT REMOVE. Must be last rung)



Program blocks / EM02_Axis02_K

EM02_CM01_EMConditions [FB302]

EM02_CM01_EMConditions Properties

General

Name	EM02_CM01_EMConditions	Number	302	Type	FB
------	------------------------	--------	-----	------	----

Language	LAD	Numbering	Manual
----------	-----	-----------	--------

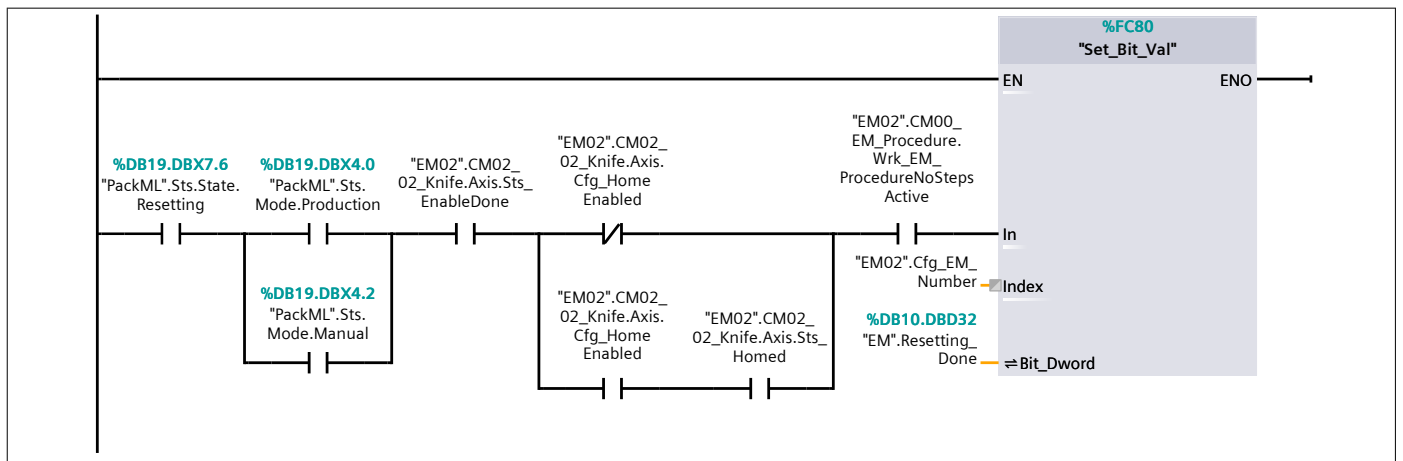
Information

Title		Author		Comment	
-------	--	--------	--	---------	--

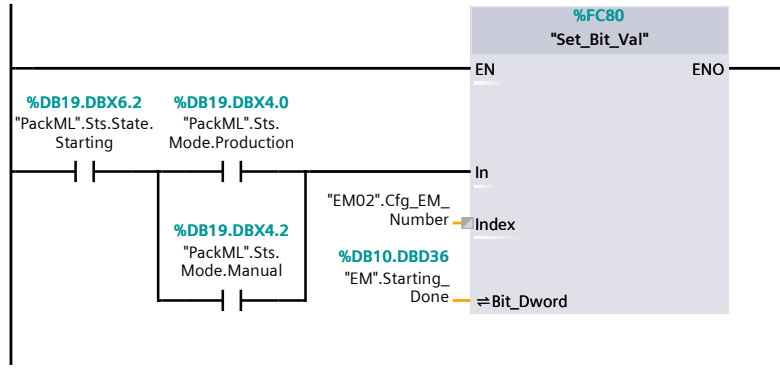
Family		Version	0.1	User-defined ID	
--------	--	---------	-----	-----------------	--

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Wrk_Temp	DInt	0	Non-retain
Temp			
Constant			

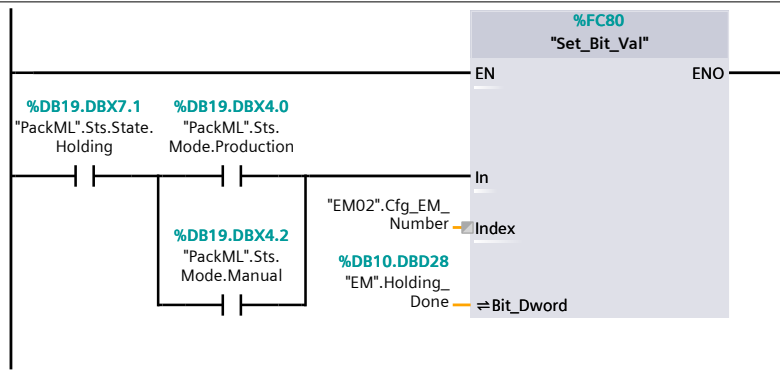
Network 1: Resetting state



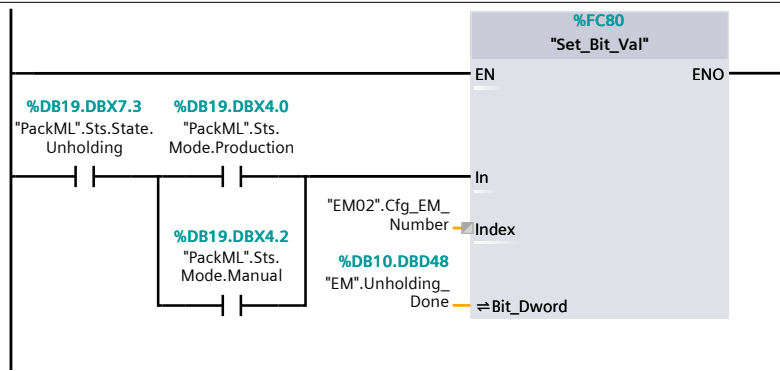
Network 2: Starting state



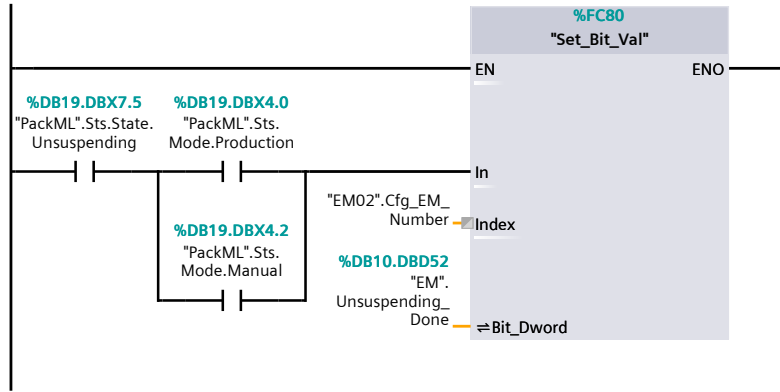
Network 3: Holding state



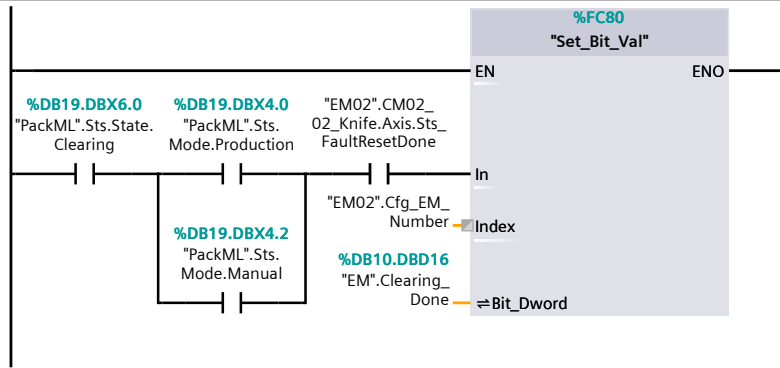
Network 4: Unholding



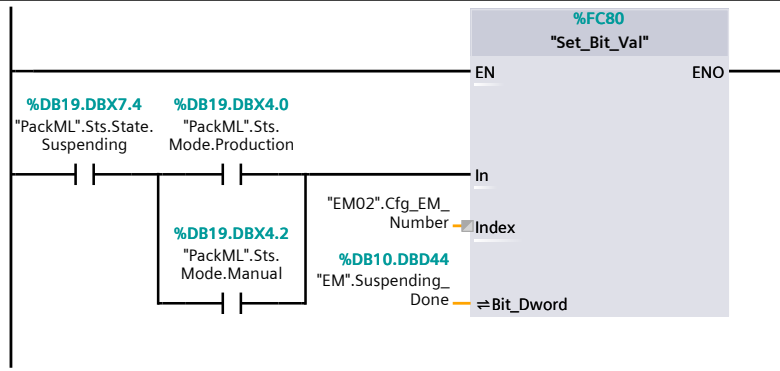
Network 5: Unsuspending



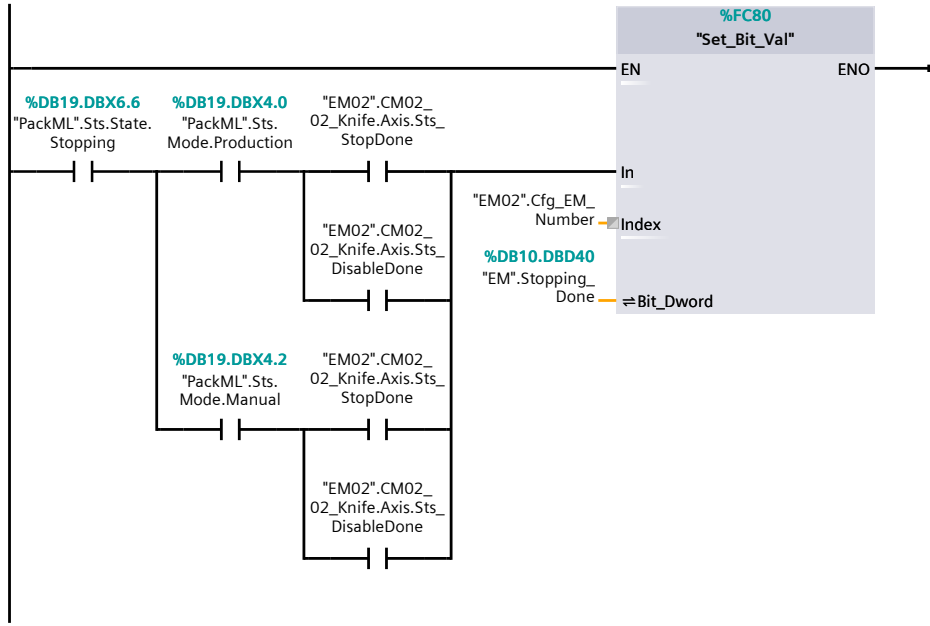
Network 6: Clearing state



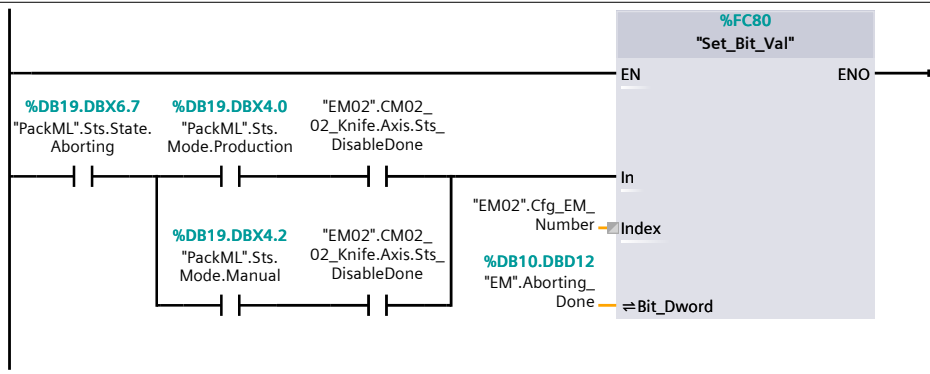
Network 7: Suspending state



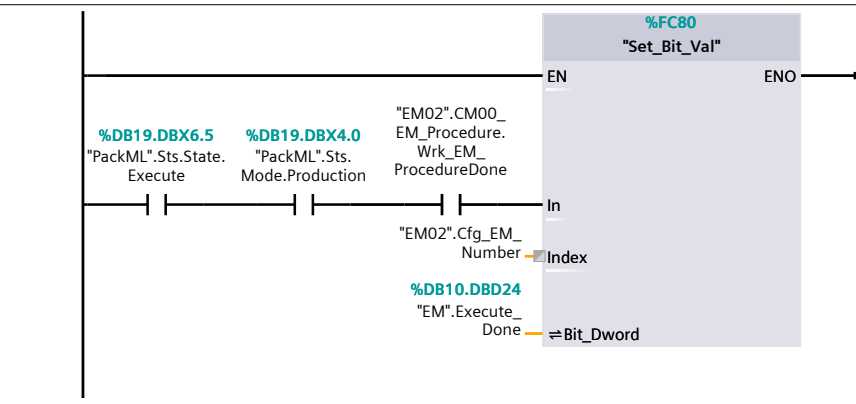
Network 8: Stopping state



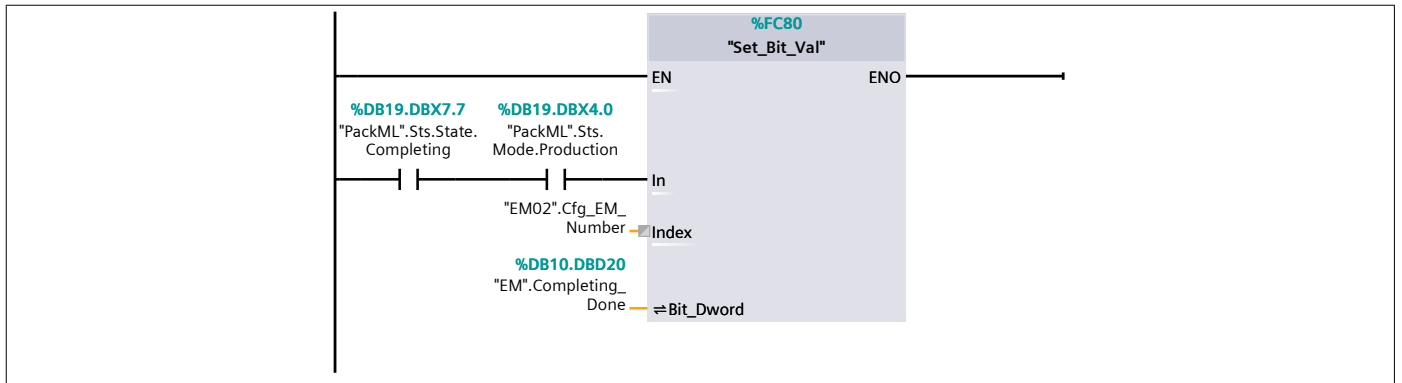
Network 9: Aborting state



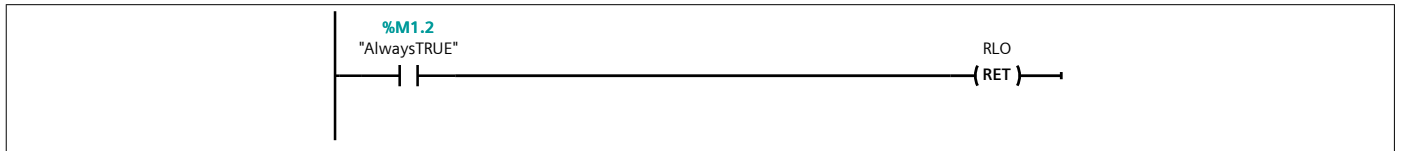
Network 10: Execute state



Network 11: Completing State



Network 12:



Program blocks / EM02_Axis02_K

EM02_SR03_FaultHandler [FB311]

EM02_SR03_FaultHandler Properties

General

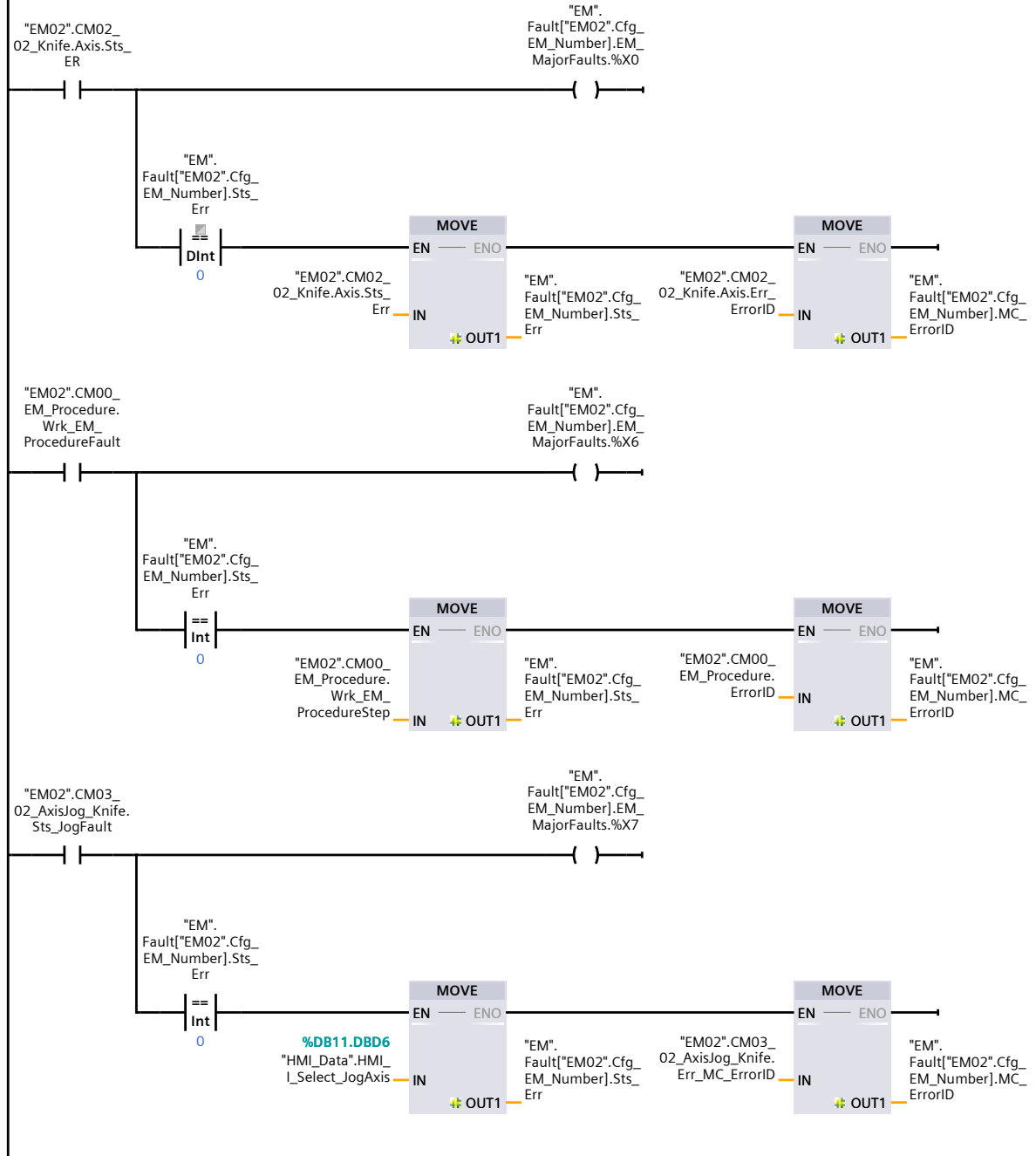
Name	EM02_SR03_FaultHandler	Number	311	Type	FB
Language	LAD	Numbering	Manual		

Information

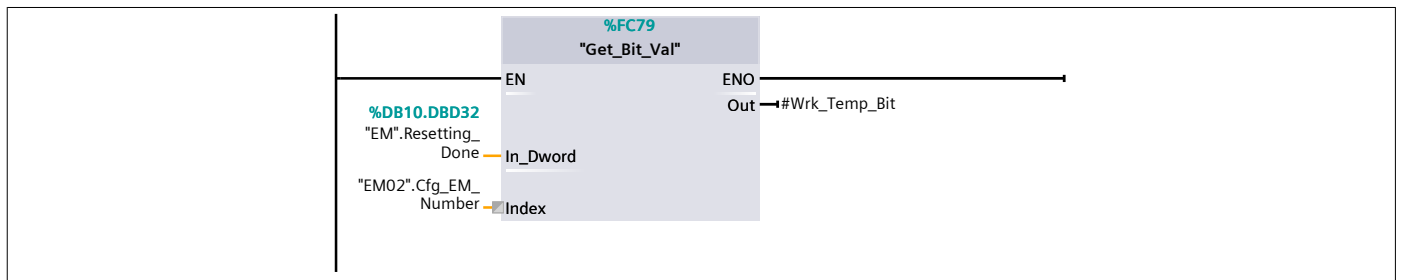
Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value	Retain
Input			
Output			
InOut			
▼ Static			
Wrk_Temp	DInt	0	Non-retain
Wrk_Temp_Bit	Bool	false	Non-retain
Temp			
Constant			

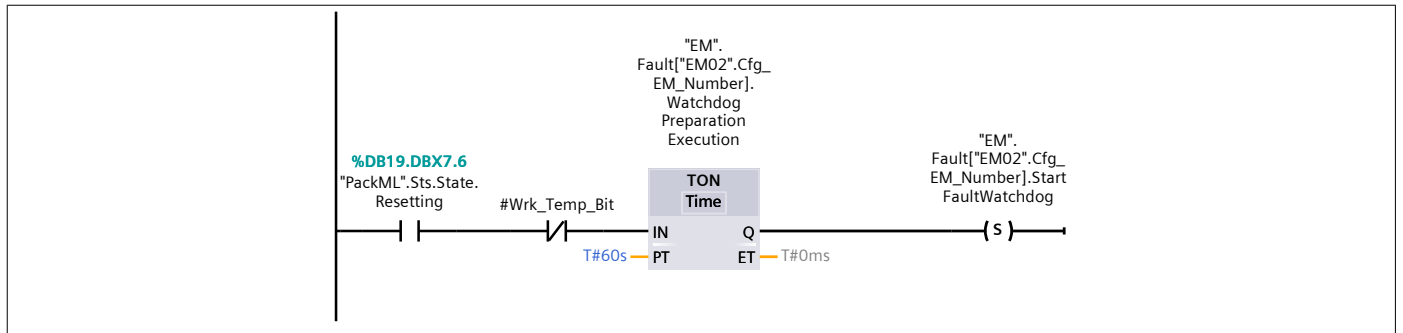
Network 1: Map axis errors, procedure motion errors, and jogging errors into bits of the EM_Major-Faults



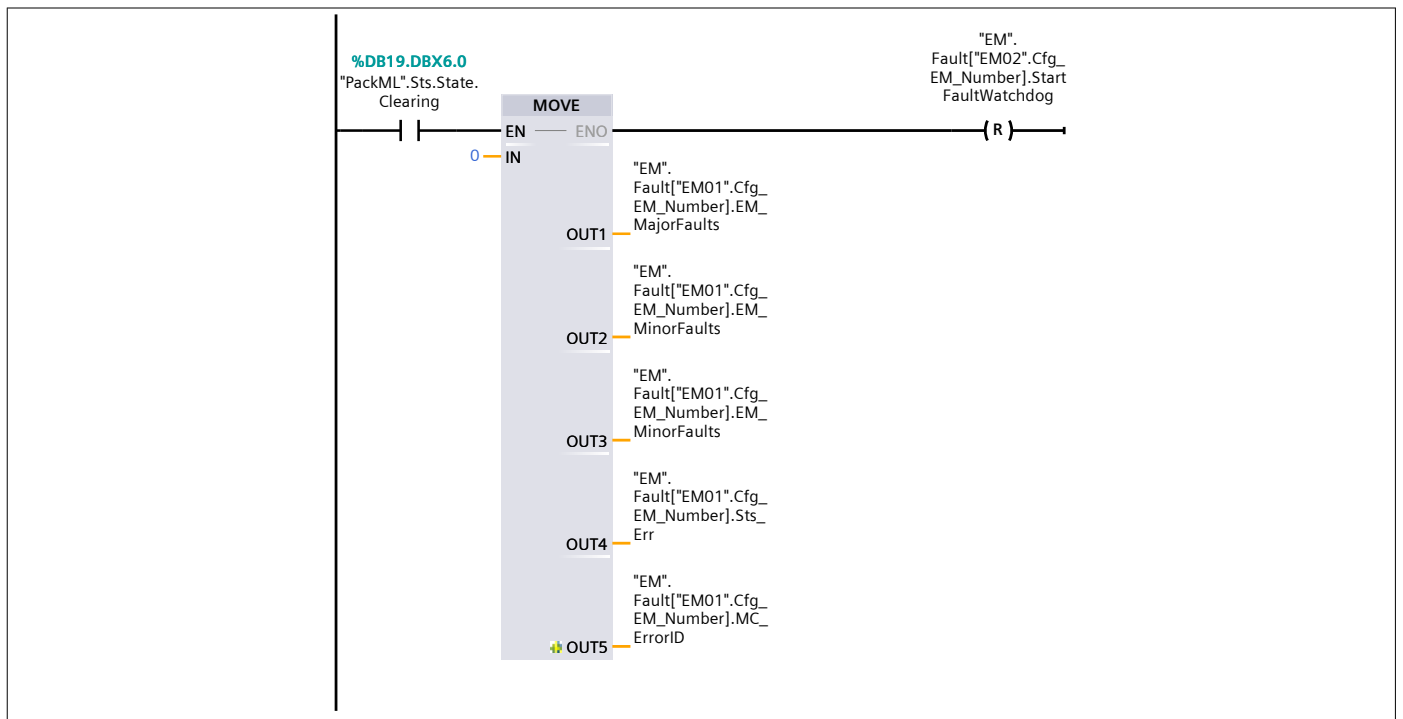
Network 2: Get resetting status for this EM



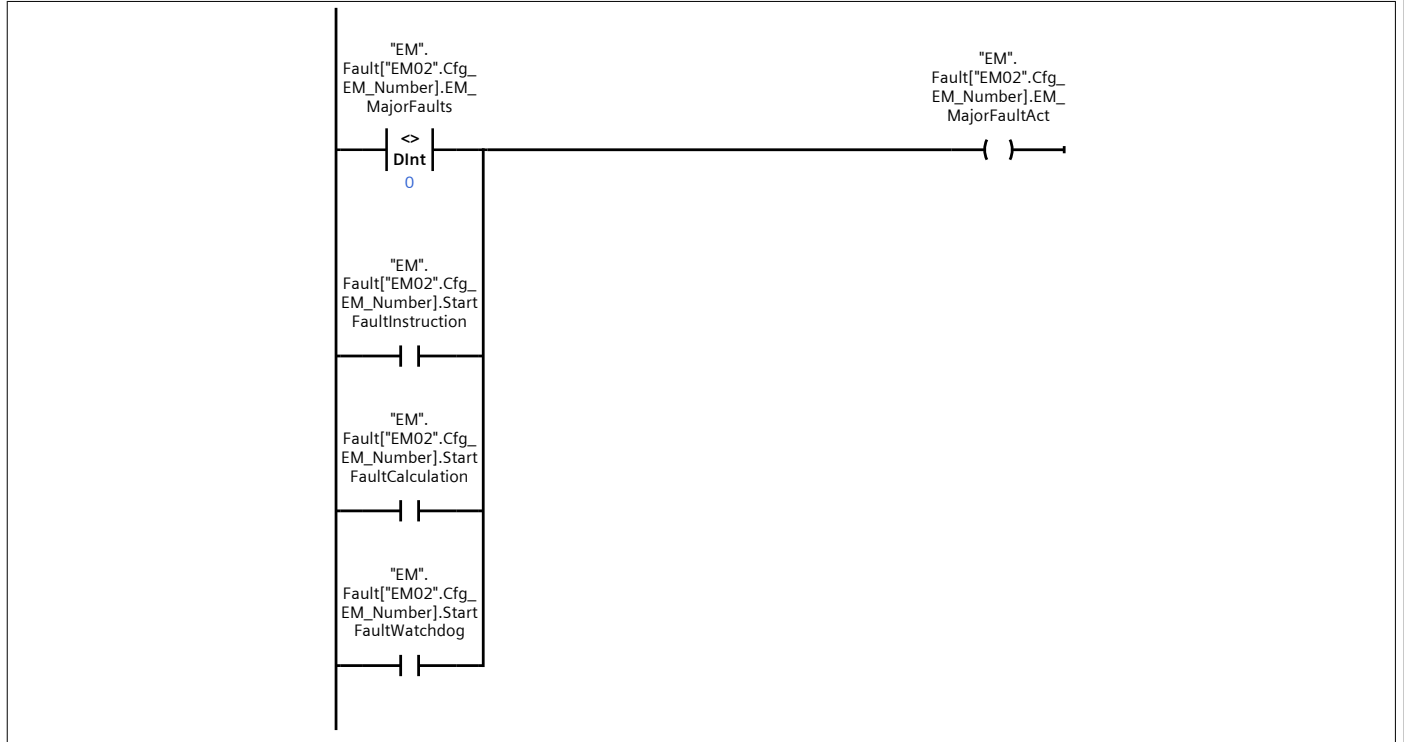
Network 3: Watchdog for resetting - CREATE AN ERROR IF TIMEOUT OCCURS



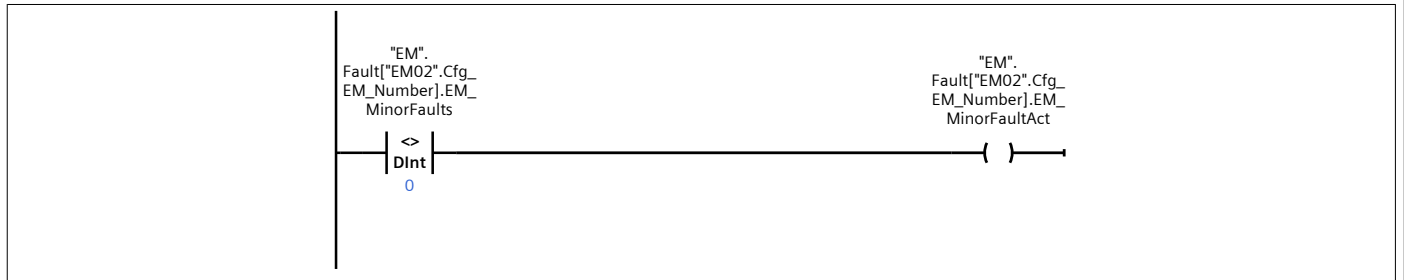
Network 4: Clear watchdog and instruction fault error details



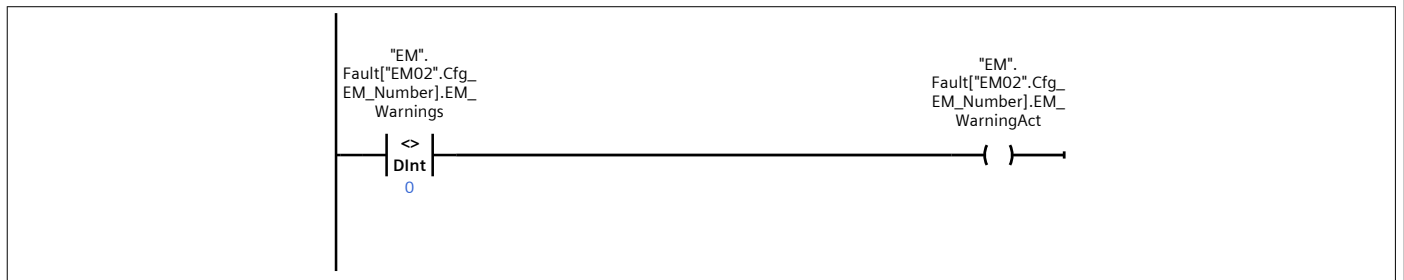
Network 5: Summary of EM faults



Network 6:



Network 7:



Program blocks / Misc Blocks

Axis_Status_Decode [FC70]

Axis_Status_Decode Properties

General

Name	Axis_Status_Decode	Number	70	Type	FC
Language	SCL	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value
▼ Input		
Status_Word	DWord	
Error_Word	DWord	
Warning_Word	DWord	
▼ Output		
Decoded_Status	"Servo_StatusWord_Type"	
Decoded_Error	"Servo_ErrorWord_Type"	
Decoded_Warning	"Servo_Warning-Word_Type"	
InOut		
Temp		
Constant		
▼ Return		
Axis_Status_Decode	Void	

```

0001 // Decodes servo status words into booleans.
0002 //
0003 // Inputs are the 3 status Dwords from the technology object.
0004 // Outputs are data types that have the DWord and the individual booleans.
0005 //
0006 #Decoded_Status.TO_DWord           := #Status_Word;
0007 #Decoded_Status.Enable             := #Decoded_Status.TO_DWord.%X0;
0008 #Decoded_Status.Error              := #Decoded_Status.TO_DWord.%X1;
0009 #Decoded_Status.RestartActive      := #Decoded_Status.TO_DWord.%X2;
0010 #Decoded_Status.OnlineStartValuesChanged := #Decoded_Status.TO_DWord.%X3;
0011 #Decoded_Status.ControlPanelActive := #Decoded_Status.TO_DWord.%X4;
0012 #Decoded_Status.HomingDone        := #Decoded_Status.TO_DWord.%X5;
0013 #Decoded_Status.Done               := #Decoded_Status.TO_DWord.%X6;
0014 #Decoded_Status.Standstill         := #Decoded_Status.TO_DWord.%X7;
0015 #Decoded_Status.PositioningCommand := #Decoded_Status.TO_DWord.%X8;
0016 #Decoded_Status.JogCommand         := #Decoded_Status.TO_DWord.%X9;
0017 #Decoded_Status.VelocityCommand    := #Decoded_Status.TO_DWord.%X10;
0018 #Decoded_Status.HomingCommand      := #Decoded_Status.TO_DWord.%X11;
0019 #Decoded_Status.ConstantVelocity    := #Decoded_Status.TO_DWord.%X12;
0020 #Decoded_Status.Accelerating        := #Decoded_Status.TO_DWord.%X13;
0021 #Decoded_Status.Decelerating       := #Decoded_Status.TO_DWord.%X14;
0022 #Decoded_Status.SWLimitMinActive   := #Decoded_Status.TO_DWord.%X15;
0023 #Decoded_Status.SWLimitMaxActive   := #Decoded_Status.TO_DWord.%X16;

```

```

0024 #Decoded_Status.HWLimitMinActive           := #Decoded_Status.TO_DWord.%X17;
0025 #Decoded_Status.HWLimitMaxActive           := #Decoded_Status.TO_DWord.%X18;
0026 #Decoded_Status.Synchronizing              := #Decoded_Status.TO_DWord.%X21;
0027 #Decoded_Status.Synchronous                := #Decoded_Status.TO_DWord.%X22;
0028 #Decoded_Status.SuperimposedMotionCommand := #Decoded_Status.TO_DWord.%X23;
0029 //
0030 //
0031 #Decoded_Error.TO_DWord                      := #Error_Word;
0032 #Decoded_Error.SystemFault                  := #Decoded_Error.TO_DWord.%X0;
0033 #Decoded_Error.ConfigurationFault           := #Decoded_Error.TO_DWord.%X1;
0034 #Decoded_Error.UserFault                    := #Decoded_Error.TO_DWord.%X2;
0035 #Decoded_Error.CommandNotAccepted           := #Decoded_Error.TO_DWord.%X3;
0036 #Decoded_Error.DriveFault                   := #Decoded_Error.TO_DWord.%X4;
0037 #Decoded_Error.SensorFault                  := #Decoded_Error.TO_DWord.%X5;
0038 #Decoded_Error.DynamicError                 := #Decoded_Error.TO_DWord.%X6;
0039 #Decoded_Error.CommunicationFault           := #Decoded_Error.TO_DWord.%X7;
0040 #Decoded_Error.SWLimit                      := #Decoded_Error.TO_DWord.%X8;
0041 #Decoded_Error.HWLimit                      := #Decoded_Error.TO_DWord.%X9;
0042 #Decoded_Error.HomingFault                  := #Decoded_Error.TO_DWord.%X10;
0043 #Decoded_Error.FollowingErrorFault          := #Decoded_Error.TO_DWord.%X11;
0044 #Decoded_Error.PositioningFault             := #Decoded_Error.TO_DWord.%X12;
0045 #Decoded_Error.PeripheralError             := #Decoded_Error.TO_DWord.%X13;
0046 #Decoded_Error.SynchronousError            := #Decoded_Error.TO_DWord.%X14;
0047 //
0048 //
0049 #Decoded_Warning.TO_Dword                    := #Warning_Word;
0050 #Decoded_Warning.ConfigurationFault          := #Decoded_Warning.TO_Dword.%X1;
0051 #Decoded_Warning.CommandNotAccepted         := #Decoded_Warning.TO_Dword.%X3;
0052 #Decoded_Warning.DynamicError               := #Decoded_Warning.TO_Dword.%X6;
0053 #Decoded_Warning.FollowingErrorWarning      := #Decoded_Warning.TO_Dword.%X11;

```

Program blocks / Misc Blocks

Get_Bit_Val [FC79]

Get_Bit_Val Properties

General

Name	Get_Bit_Val	Number	79	Type	FC
Language	SCL	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value
▼ Input		
In_Dword	DWord	
Index	DInt	
▼ Output		
Out	Bool	
InOut		
Temp		
Constant		
▼ Return		
Get_Bit_Val	Void	

```

0001 // Gets value of bit of Dword
0002 CASE #Index OF
0003     0:
0004         #Out := #In_Dword.%X0;
0005     1:
0006         #Out := #In_Dword.%X1;
0007     2:
0008         #Out := #In_Dword.%X2;
0009     3:
0010         #Out := #In_Dword.%X3;
0011     4:
0012         #Out := #In_Dword.%X4;
0013     5:
0014         #Out := #In_Dword.%X5;
0015     6:
0016         #Out := #In_Dword.%X6;
0017     7:
0018         #Out := #In_Dword.%X7;
0019     8:
0020         #Out := #In_Dword.%X8;
0021     9:
0022         #Out := #In_Dword.%X9;
0023    10:
0024         #Out := #In_Dword.%X10;
0025    11:
0026         #Out := #In_Dword.%X11;
0027    12:

```

```
0028     #Out := #In_Dword.%X12;  
0029 13:  
0030     #Out := #In_Dword.%X13;  
0031 14:  
0032     #Out := #In_Dword.%X14;  
0033 15:  
0034     #Out := #In_Dword.%X15;  
0035 16:  
0036     #Out := #In_Dword.%X16;  
0037 17:  
0038     #Out := #In_Dword.%X17;  
0039 18:  
0040     #Out := #In_Dword.%X18;  
0041 19:  
0042     #Out := #In_Dword.%X19;  
0043 20:  
0044     #Out := #In_Dword.%X20;  
0045 21:  
0046     #Out := #In_Dword.%X21;  
0047 22:  
0048     #Out := #In_Dword.%X22;  
0049 23:  
0050     #Out := #In_Dword.%X23;  
0051 24:  
0052     #Out := #In_Dword.%X24;  
0053 25:  
0054     #Out := #In_Dword.%X25;  
0055 26:  
0056     #Out := #In_Dword.%X26;  
0057 27:  
0058     #Out := #In_Dword.%X27;  
0059 28:  
0060     #Out := #In_Dword.%X28;  
0061 29:  
0062     #Out := #In_Dword.%X29;  
0063 30:  
0064     #Out := #In_Dword.%X30;  
0065 31:  
0066     #Out := #In_Dword.%X31;  
0067 END_CASE;
```

Program blocks / Misc Blocks

MEQ_DWORD [FC81]

MEQ_DWORD Properties

General

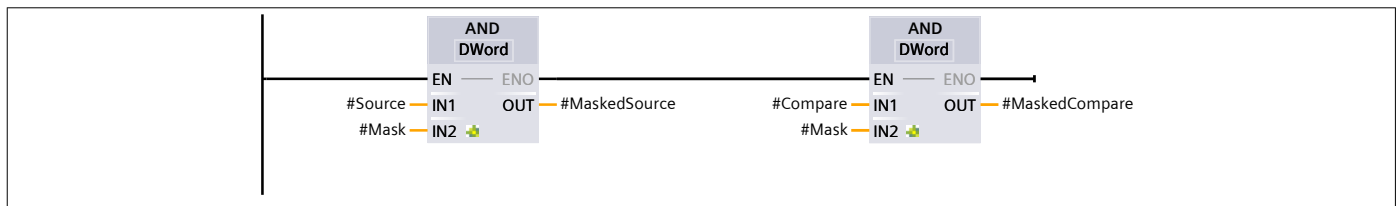
Name	MEQ_DWORD	Number	81	Type	FC
Language	LAD	Numbering	Manual		

Information

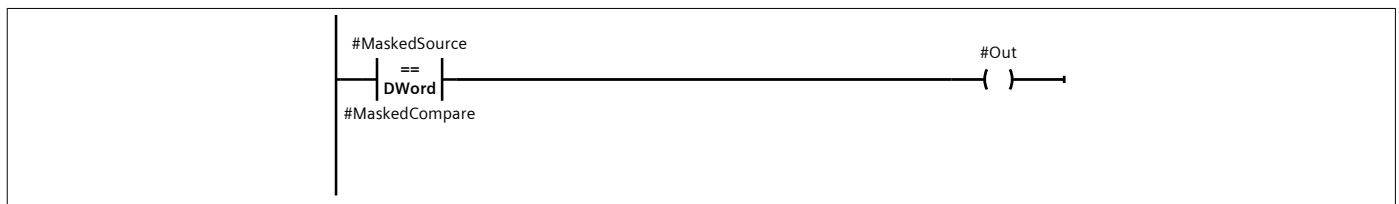
Title	Masked Equal for DWord	Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value
▼ Input		
Source	DWord	
Mask	DWord	
Compare	DWord	
▼ Output		
Out	Bool	
InOut		
▼ Temp		
MaskedSource	DWord	
MaskedCompare	DWord	
Constant		
▼ Return		
MEQ_DWORD	Void	

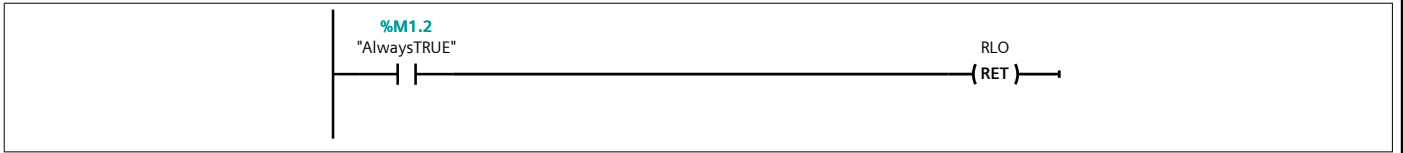
Network 1: Do masking



Network 2: Compare



Network 3: Always sets ENO true



Program blocks / Misc Blocks

Set_Bit_Val [FC80]

Set_Bit_Val Properties

General

Name	Set_Bit_Val	Number	80	Type	FC
Language	SCL	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value
▼ Input		
In	Bool	
Index	DInt	
Output		
▼ InOut		
Bit_Dword	DWord	
Temp		
Constant		
▼ Return		
Set_Bit_Val	Void	

```

0001 // Places boolean value in bit of Dword
0002 CASE #Index OF
0003     0:
0004         #Bit_Dword.%X0 := #In;
0005     1:
0006         #Bit_Dword.%X1 := #In;
0007     2:
0008         #Bit_Dword.%X2 := #In;
0009     3:
0010         #Bit_Dword.%X3 := #In;
0011     4:
0012         #Bit_Dword.%X4 := #In;
0013     5:
0014         #Bit_Dword.%X5 := #In;
0015     6:
0016         #Bit_Dword.%X6 := #In;
0017     7:
0018         #Bit_Dword.%X7 := #In;
0019     8:
0020         #Bit_Dword.%X8 := #In;
0021     9:
0022         #Bit_Dword.%X9 := #In;
0023    10:
0024         #Bit_Dword.%X10 := #In;
0025    11:
0026         #Bit_Dword.%X11 := #In;
0027    12:

```

```
0028     #Bit_Dword.%X12 := #In;
0029     13:
0030     #Bit_Dword.%X13 := #In;
0031     14:
0032     #Bit_Dword.%X14 := #In;
0033     15:
0034     #Bit_Dword.%X15 := #In;
0035     16:
0036     #Bit_Dword.%X16 := #In;
0037     17:
0038     #Bit_Dword.%X17 := #In;
0039     18:
0040     #Bit_Dword.%X18 := #In;
0041     19:
0042     #Bit_Dword.%X19 := #In;
0043     20:
0044     #Bit_Dword.%X20 := #In;
0045     21:
0046     #Bit_Dword.%X21 := #In;
0047     22:
0048     #Bit_Dword.%X22 := #In;
0049     23:
0050     #Bit_Dword.%X23 := #In;
0051     24:
0052     #Bit_Dword.%X24 := #In;
0053     25:
0054     #Bit_Dword.%X25 := #In;
0055     26:
0056     #Bit_Dword.%X26 := #In;
0057     27:
0058     #Bit_Dword.%X27 := #In;
0059     28:
0060     #Bit_Dword.%X28 := #In;
0061     29:
0062     #Bit_Dword.%X29 := #In;
0063     30:
0064     #Bit_Dword.%X30 := #In;
0065     31:
0066     #Bit_Dword.%X31 := #In;
0067     END_CASE;
0068
```

Program blocks / Misc Blocks

States_Disabled_Decode [FC82]

States_Disabled_Decode Properties

General

Name	States_Disabled_Decode	Number	82	Type	FC
Language	SCL	Numbering	Manual		

Information

Title		Author		Comment	
Family		Version	0.1	User-defined ID	

Name	Data type	Default value
▼ Input		
Disabled_States	DInt	
▼ Output		
ClearingEnabled	Bool	
StoppedEnabled	Bool	
StartingEnabled	Bool	
IdleEnabled	Bool	
SuspendedEnabled	Bool	
ExecuteEnabled	Bool	
StoppingEnabled	Bool	
AbortingEnabled	Bool	
AbortedEnabled	Bool	
HoldingEnabled	Bool	
HeldEnabled	Bool	
UnholdingEnabled	Bool	
SuspendingEnabled	Bool	
UnsuspendingEnabled	Bool	
ResettingEnabled	Bool	
CompletingEnabled	Bool	
CompleteEnabled	Bool	
InOut		
Temp		
Constant		
▼ Return		
States_Disabled_Decode	Void	

```

0001 // Decode StatesDisabled Dint output from LPMLV30_UnitModeStateManager FB
0002 // Disabled states are "1" bits in the Dint.
0003 // Decode into individual enable Booleans that are "1" if state is enabled
0004 //
0005 #ClearingEnabled := NOT #Disabled_States.%X1;
0006 #StoppedEnabled := NOT #Disabled_States.%X2;
0007 #StartingEnabled := NOT #Disabled_States.%X3;
0008 #IdleEnabled := NOT #Disabled_States.%X4;
0009 #SuspendedEnabled := NOT #Disabled_States.%X5;
0010 #ExecuteEnabled := NOT #Disabled_States.%X6;

```

Totally Integrated
Automation Portal

```
0011 #StoppingEnabled := NOT #Disabled_States.%X7;  
0012 #AbortingEnabled := NOT #Disabled_States.%X8;  
0013 #AbortedEnabled := NOT #Disabled_States.%X9;  
0014 #HoldingEnabled := NOT #Disabled_States.%X10;  
0015 #HeldEnabled := NOT #Disabled_States.%X11;  
0016 #UnholdingEnabled := NOT #Disabled_States.%X12;  
0017 #SuspendingEnabled := NOT #Disabled_States.%X13;  
0018 #UnsuspendingEnabled := NOT #Disabled_States.%X14;  
0019 #ResettingEnabled := NOT #Disabled_States.%X15;  
0020 #CompletingEnabled := NOT #Disabled_States.%X16;  
0021 #CompleteEnabled := NOT #Disabled_States.%X17;
```