OMRON

Sysmac Library
for Sysmac Studio
Automation Software

PackML Support Library Implementation Guide

SYSMAC-SE2□□□□

SYSMAC
always in control

W22I-E-01

NOTE

No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- Microsoft, Windows, Windows Vista, Excel, and Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.
- EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- ODVA, CIP, CompoNet, DeviceNet, and EtherNet/IP are trademarks of ODVA.
- The SD and SDHC logos are trademarks of SD-3C, LLC.

Other company names and product names in this document are the trademarks or registered trademarks of their respective companies.

Copyrights

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

Terms and Conditions Agreement

Warranty, Limitations of Liability

Warranties

- **Exclusive Warranty**

  Omron's exclusive warranty is that the Products will be free from defects in materials and workmanship for a period of twelve months from the date of sale by Omron (or such other period expressed in writing by Omron). Omron disclaims all other warranties, express or implied.

- **Limitations**

  OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE PRODUCTS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE.

  Omron further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Products or otherwise of any intellectual property right.

- **Buyer Remedy**

  Omron's sole obligation hereunder shall be, at Omron's election, to (i) replace (in the form originally shipped with Buyer responsible for labor charges for removal or replacement thereof) the non-complying Product, (ii) repair the non-complying Product, or (iii) repay or credit Buyer an amount equal to the purchase price of the non-complying Product; provided that in no event shall Omron be responsible for warranty, repair, indemnity or any other claims or expenses regarding the Products unless Omron's analysis confirms that the Products were properly handled, stored, installed and maintained and not subject to contamination, abuse, misuse or inappropriate modification. Return of any Products by Buyer must be approved in writing by Omron before shipment. Omron Companies shall not be liable for the suitability or unsuitability or the results from the use of Products in combination with any electrical or electronic components, circuits, system assemblies or any other materials or substances or environments. Any advice, recommendations or information given orally or in writing, are not to be construed as an amendment or addition to the above warranty.

See http://www.omron.com/global/ or contact your Omron representative for published information.

Limitation on Liability; Etc

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.

Further, in no event shall liability of Omron Companies exceed the individual price of the Product on which liability is asserted.

Application Considerations

Suitability of Use

Omron Companies shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Product in the Buyer's application or use of the Product. At Buyer's request, Omron will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Product. This information by itself is not sufficient for a complete determination of the suitability of the Product in combination with the end product, machine, system, or other application or use. Buyer shall be solely responsible for determining appropriateness of the particular Product with respect to Buyer's application, product or system. Buyer shall take application responsibility in all cases.

NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Programmable Products

Omron Companies shall not be responsible for the user's programming of a programmable Product, or any consequence thereof.

Disclaimers

Performance Data

Data presented in Omron Company websites, catalogs and other materials is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of Omron's test conditions, and the user must correlate it to actual application requirements. Actual performance is subject to the Omron's Warranty and Limitations of Liability.

Change in Specifications

Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions

Information presented by Omron Companies has been checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical or proofreading errors or omissions.

Cautions

⚠ Caution

Read all related manuals carefully before you use this library.	
Check the user program, data, and parameter settings for proper execution before you use them for actual operation.	
Keep the emergency stop switch in hand to prevent a sudden operation of the motor when you perform testing operation.	
The sample programming shows only the portion of a program that uses the function or function block from the library.	
When using actual devices, also program safety circuits, device interlocks, I/O with other devices, and other control procedures.	
Understand the contents of sample programming before you use the sample programming and create the user program.	

Precautions for Safe Use

Operation

- The Sysmac Library and manuals are assumed to be used by personnel that is given in Intended Audience in this manual. Otherwise, do not use them.

Precautions for Correct Use

Using the Library

- You cannot change the source code of the functions or function blocks that are provided in the Sysmac Library.

Operation

- Specify the input parameter values within the valid range.
- In the function or function block with an Enabled output variable, if the value of Enabled is FALSE, do not use the processing result of the function or function block as a command value to the control target.

Revision History

A manual revision code appears as a suffix to the catalog number on the front and back covers of the manual.

Revision code	Date	Revised content
00	July 2016	Original production - Q-160106 PackML Library External Specifications
01	August 2016	Added cover, precautions, terms & conditions, corrected errors

CONTENTS

1. Summary 1
1.1. PackML Summary 1
1.2. Scope 5
1.3. Library information 5
2. Device Configuration 6
2.1. Target Models 6
2.2. Hardware Configuration 6
3. PackML Mode/State Machine Control POUs 7
3.1. Overview 7
3.2. Structure Definition 9
3.3. PackMLModeStateMachine Function Block 11
3.4. PackMLModeStateTimer Function Block 22
3.5. PMLCtrlCmd_<transition name> Function Blocks 25
3.6. PMLState_Is <state name> Function Blocks 27
4. PackML Module Command POU 29
4.1. Structure Definition 29
4.2. PMLTransitionCmd_ResetAll Function 30
4.3. PMLTransitionCmd_ResetAllCmdSetAllSC Function 32
4.4. PMLTransitionCmd_Summarize Function 33
4.5. PMLTransitionCmd_SummarizePackTagCtrlCmd Function 35
5. Alarm Control POU 37
5.1. Overview 37
5.2. Structure Definition 38
5.3. Alarm Function Block 41
5.4. AlarmStatus_Update Function 43
5.5. AlarmSummation_Add Function 44
5.6. AlarmSummation_SortFilter Function Block 46
5.7. DT_TO_PackTagDINTarray Function 49
6. Sample Project 50
6.1. Sample HMI 50
6.2. Sample Program 53

1. Summary

1.1. PackML Summary

1.1.1. PackML

PackML (Packaging Machine Language) is the general specifications for packaging machine defined by Packaging Work Group in OMAC (Organization for Machine Automation and Control).

Adapted Standard: ISA TR88.00.02 Rev.2 (2015) known as PackML v3.0

1.1.2. Mode and State

-Mode-

We support the machine mode in PACKML (refer to Packaging Machine Language V3.0 Mode & States Definition Document)

1. PRODUCTION MODE
   This represents the mode which is utilized for routine production. The machine executes relevant logic in response to commands which are either entered directly by the operator or issued by another supervisory system.

2. MAINTENANCE MODE
   This mode allows, may allow suitably authorised personnel, the ability to run an individual machine independent of other machines in a production line. This mode would typically be used for faultfinding, machine trials or testing operational improvements. This mode would also allow the speed of the machine to be adjusted (where this feature is available).

3. MANUAL MODE
   This provides direct control of individual machine axes. This feature is available depending upon the mechanical constraints of the mechanisms being exercised. This feature would be typically used for the commissioning of individual drives, verifying the operation of synchronised drives, testing the drive as a result of modifying parameters etc.

-State-

We support the machine State in PACKML (refer to Packaging Machine Language V3.0 Mode & States Definition Document)

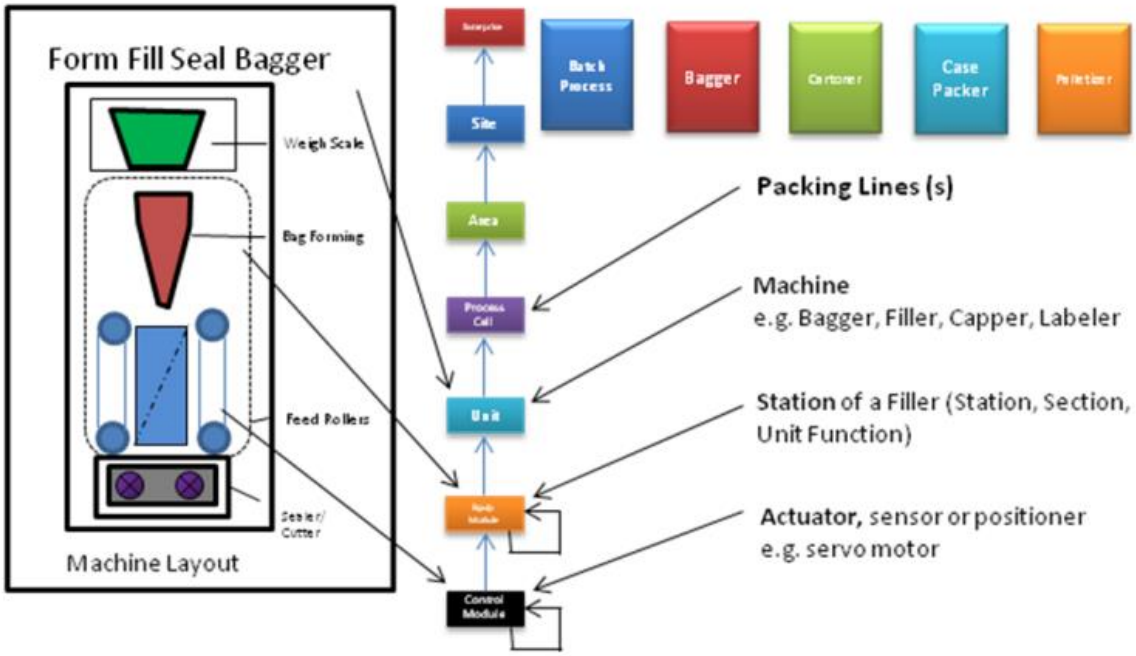
State Name	Description
STOPPED	**State Type: Wait** The machine is powered and stationary after completing the STOPPING state. All communications with other systems are functioning (if applicable). A *RESET* command will cause an exit from STOPPED to the RESETTING state.
STARTING	**State Type: Acting** The machine completes the steps needed to start. This state is entered as a result of a *STARTING* command (local or remote). Following this command the machine will begin to “execute”.
IDLE	**State Type: Wait** This is the state which indicates that RESETTING is complete. The machine will maintain the conditions which were achieved during the RESETTING state, and perform operations required when the machine is in IDLE.
SUSPENDING	**State Type: Acting** This state shall be used when EXTERNAL (outside this unit machine but usually on the same integrated production line) process conditions do not allow the machine to continue producing, that is, the machine leaves EXECUTE due to upstream or downstream conditions on the line. This is typically due to a blocked or starved event. This condition may be detected by a local machine sensor or based on a supervisory system external command. While in the SUSPENDING state, the machine is typically brought to a controlled stop and then transitions to SUSPENDED upon state complete. To be able to restart production correctly after the SUSPENDED state, all relevant process set-points and return status of the procedures at the time of receiving the *SUSPEND* command must be saved in the machine controller when executing the SUSPENDING procedure.
SUSPENDED	**State Type: Wait** Refer to SUSPENDING for when this state is used. In this state the machine shall not produce product. It will either stop running or continue to cycle without producing until external process conditions return to normal, at which time, the SUSPENDED state will transition to the UNSUSPENDING state, typically without any operator intervention.
UNSUSPENDING	**State Type: Acting** Refer to SUSPENDING for when this state is used. This state is a result of process conditions returning to normal. The UNSUSPENDING state initiates any required actions or sequences necessary to transition the machine from SUSPENDED back to EXECUTE. To be able to restart production correctly after the SUSPENDED state, all relevant process set-points and return status of the procedures at the time of receiving the SUSPEND command must be saved in the machine controller when executing the SUSPENDING procedure.
EXECUTE	**State Type: Acting** Once the machine is processing materials it is in the EXECUTE state. Different machine modes will result in specific types of EXECUTE activities. For example, if the machine is in the “Production” mode, the EXECUTE will result in products being produced, while in “Clean Out” mode the EXECUTE state refers to the action of cleaning the machine.
STOPPING	**State Type: Acting** This state is entered in response to a *STOP* command. While in this state the machine executes the logic which brings it to a controlled stop as reflected by the STOPPED state. Normal STARTING of the machine cannot be initiated unless RESETTING had taken place.
ABORTING	**State Type: Acting** The ABORTING state can be entered at any time in response to the *ABORT* command or on the occurrence of a machine fault. The aborting logic will bring the machine to a rapid safe stop.
ABORTED	**State Type: Wait** The machine maintains status information relevant to the ABORT condition. The machine can only exit the ABORTED state after an explicit CLEAR command, subsequently to manual intervention to correct and reset the detected machine faults.

HOLDING	**State Type: Acting** This state shall be used when INTERNAL (inside this unit machine and not from another machine on the production line) machine conditions do not allow the machine to continue producing, that is, the machine leaves EXECUTE due to internal conditions. This is typically used for routine machine conditions that requires minor operator servicing to continue production. This state can be initiated automatically or by an operator and can be easily recovered from. An example of this would be a machine that requires an operator to periodically refill a glue dispenser or carton magazine and due to the machine design, these operations cannot be performed while the machine is running. Since these types of tasks are normal production operations, it is not desirable to go through aborting or stopping sequences, and because these functions are integral to the machine they are not considered to be "external". While in the HOLDING state, the machine is typically brought to a controlled stop and then transitions to HELD upon state complete. To be able to restart production correctly after the HELD state, all relevant process set-points and return status of the procedures at the time of receiving the *HOLD* command must be saved in the machine controller when executing the HOLDING procedure.
HELD	**State Type: Wait** Refer to HOLDING for when this state is used. In this state the machine shall not produce product. It will either stop running or continue to dry cycle. A transition to the UNHOLDING state will occur when INTERNAL machine conditions change or an *UNHOLD* command is initiated by an operator.
UNHOLDING	**State Type: Acting** Refer to HOLDING for when this state is used. A machine will typically enter into UNHOLDING automatically when INTERNAL conditions, material levels, for example, return to an acceptable level. If an operator is required to perform minor servicing to replenish materials or make adjustments, then the *UNHOLD* command may be initiated by the operator.
COMPLETEING	**State Type: Acting** This state is an automatic response from the EXECUTE state. Normal operation has run to completion, i.e. processing of material at the infeed will stop.
COMPLETE	**State Type: Wait** The machine has finished the COMPLETING state and is now waiting for a *RESET* command before transitioning to the RESETTING state.
RESETTING	**State Type: Acting** This state is the result of a *RESET* command from the STOPPED or COMPLETE state. Faults and stop causes are reset. RESETTING will typically cause safety devices to be energized and place the machine in the IDLE state where it will wait for a *START* command. No hazardous motion should happen in this state.
CLEARING	**State Type: Acting** Initiated by a state command to clear faults that may have occurred when ABORTING, and are present in the ABORTED state before proceeding to a STOPPED state.

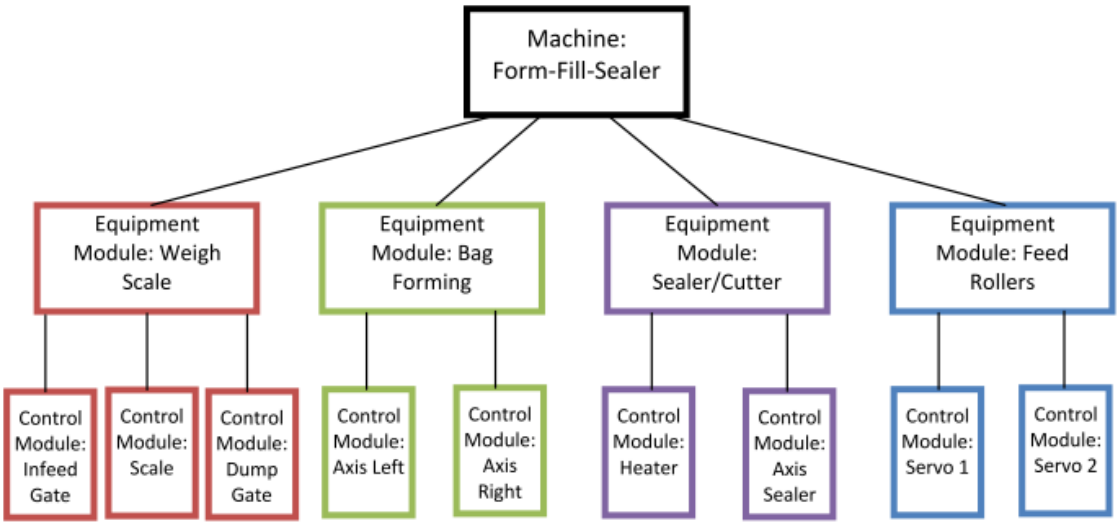
1.1.3. Hierarchy of machine module

ISA TR88 (the Based specifications of PackML) defines the hierarchy of machine module as follows.

We implement the sample program based on this hierarchy.



Example ISA88 Physical Hierarchy



1.2. Scope

We supply the following FBs, sample HMIs and Sample program.

Items	Refer
FBs or FUNs	
Manage PackML mode and State	Chapter.3
PackMLModeStateMachine Function Block	
PackMLModeStateTimer Function Block	
PMLCtrlCmd_<Transition name> Function Block	
PMLState_Is <State name> Function Block	
PackML Mode Command	Chapter.4
PMLTransitionCmd_ResetAll Function	
PMLTransitionCmd_ResetAllCmdSetAllSC Function	
PMLTransitionCmd_Summarize Function	
PMLTransitionCmd_SummarizePackTagCtrlCmd Function	
Alarm Control	Chapter.5
Alarm Function Block	
Alarm Status_Update Function	
AlarmSummation_Add Function Block	
AlarmSummation_SortFilter Function Block	
DT_TO_PackTagDINTarray	
Sample Project	Chapter.6
Include sample HMI and Program	

1.3. Library information

1.3.1. Library information

Items	
Library name	PackML30
Library file name	OmronLib_PackML30V1_0.slr
Name space	¥¥OmronLib¥PackML30
Source code	Non open

2. Device Configuration

2.1. Target Models

■Software

Name	Model	Version
Automation Software Sysmac Studio	SYSMAC-SE2□□□	V1.14 or later

■Device

Name	Model	Version
Machine Automation Controller NJ-Series CPU unit	NX701-□□□□ NJ101-□□□□	V1.10 or later
	NJ501-□□□□ NJ301-□□□□	V1.01 or later

2.2. Hardware Configuration

This FB processes only calculation in the CPU unit. No special units or external devices besides the CPU unit are required.

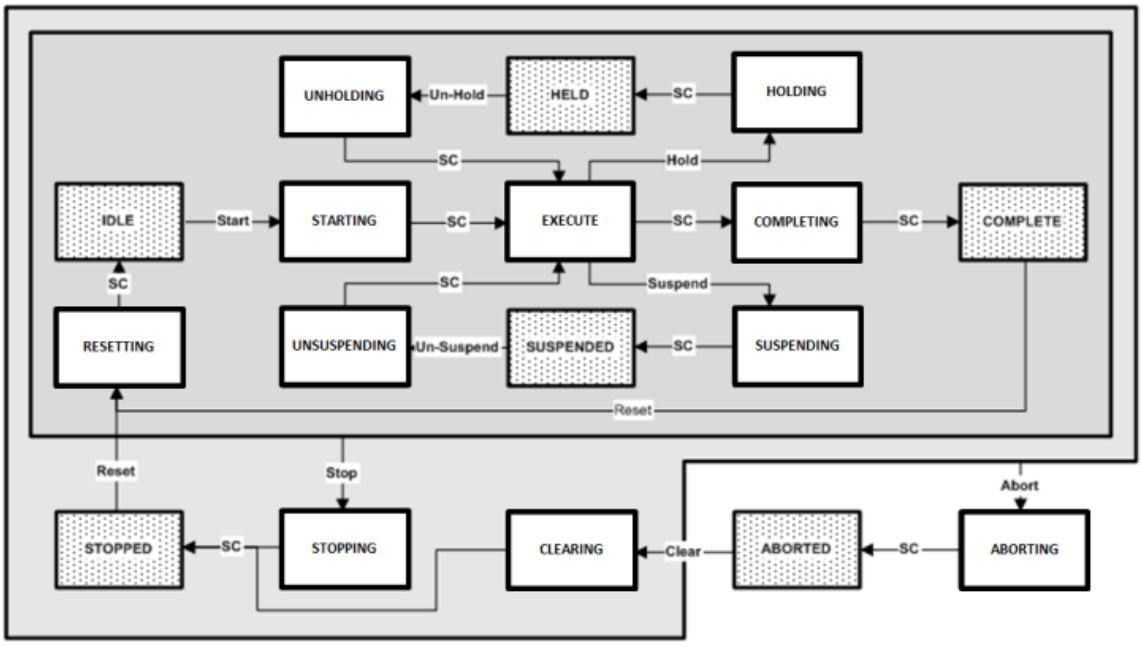
3. PackML Mode/State Machine Control POUs

3.1. Overview

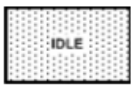
These POUs and STRUCTs provide the state machine per mode which PackML v3.0 (ISA TR8.00.02) specifies for packaging machines
Conforming to ISA TR8.00.02, the modes from 1 to 31 are supported.

ISA-TR88.00.02 Control Modes	
Value	**Unit / Machine Control Modes**
0	Invalid
1	Production
2	Maintenance
3	Manual
04 – 31	User Definable

The state machine diagram stipulated by PackML v3.0 (ISA TR88.00.02), each state number, and its meaning are as follows:



SC = State Complete



= Wait State

= Acting State

	State name	State Type	
1	Clearing	Acting	In the *Aborted* state, entering the *AbortClear* command transitions the state to *Clearing*.
2	Stopping	Wait	In the *Stopping* state, entering the *StateComplete* command transitions the state to *Stopped*.e.g.) A machine stops while power is supplied.
3	Starting	Acting	In the *Idle* state, entering the *Start* command transitions the state to *Starting*.
4	Idle	Wait	In the *Resetting* state, entering the *StateComplete* command transitions the state to *Idle*. e.g.) Operation has been prepared.
5	Suspended	Acting	In the *Suspending* state, entering the *StateComplete* command transitions the state to *Suspended*. e.g.) Automatic stop at upper and lower phase.
6	Execute	Dual State	The state is transitioned to *Execute* by the following ways: - In the *Starting* state, enter the *StateComplete* command. - In the *UnHolding* state, enter the *StateComplete* command. - In the *UnSuspending* state, enter the *StateComplete* command. e.g.) A machine is producing products. Operation differs depending on the current mode.
7	Stopping	Acting	Entering the *Stop* command transitions the state to *Stopping*.
8	Aborting	Acting	Entering the *Abort* command transitions the state to *Aborting*.
9	Aborted	Wait	In the *Aborting* state, entering the *StateComplete* command transitions the state to *Aborted*.
10	Holdng	Acting	In the *Execute* state, entering *Hold* command transitions the state to *Holding*.
11	Held	Wait	In the *Holding* state, entering the *StateComplete* command transitions the state to *Held*. e.g.) A machine is stopped temporarily for a test run of the machine, or for solving a machine trouble.
12	UnHolding	Acting	In the *Held* state, entering the *UnHold* command transitions the state to *UnHolding*.
13	Suspending	Wait	In the *Execute* state, entering the *Suspend* command s transitions the state to *Suspending*.
14	UnSuspending	Wait	In the *Suspended* state, entering the *UnSuspend* command transitions the state to *UnSuspending*.
15	Reseting	Acting	In the *Complete* state, entering the *Reset* command transitions the state to *Resetting*.
16	Completing	Acting	In the *Execute* state, entering the *StateComplete* command transitions the state to *Completing*.
17	Complete	Wait	In the *Completing* state, entering the *StateComplete* command transitions the state to *Complete*.e.g.) A cycle has stopped.

3.2. Structure Definition

3.2.1. sPACKML_STATES_FLAG

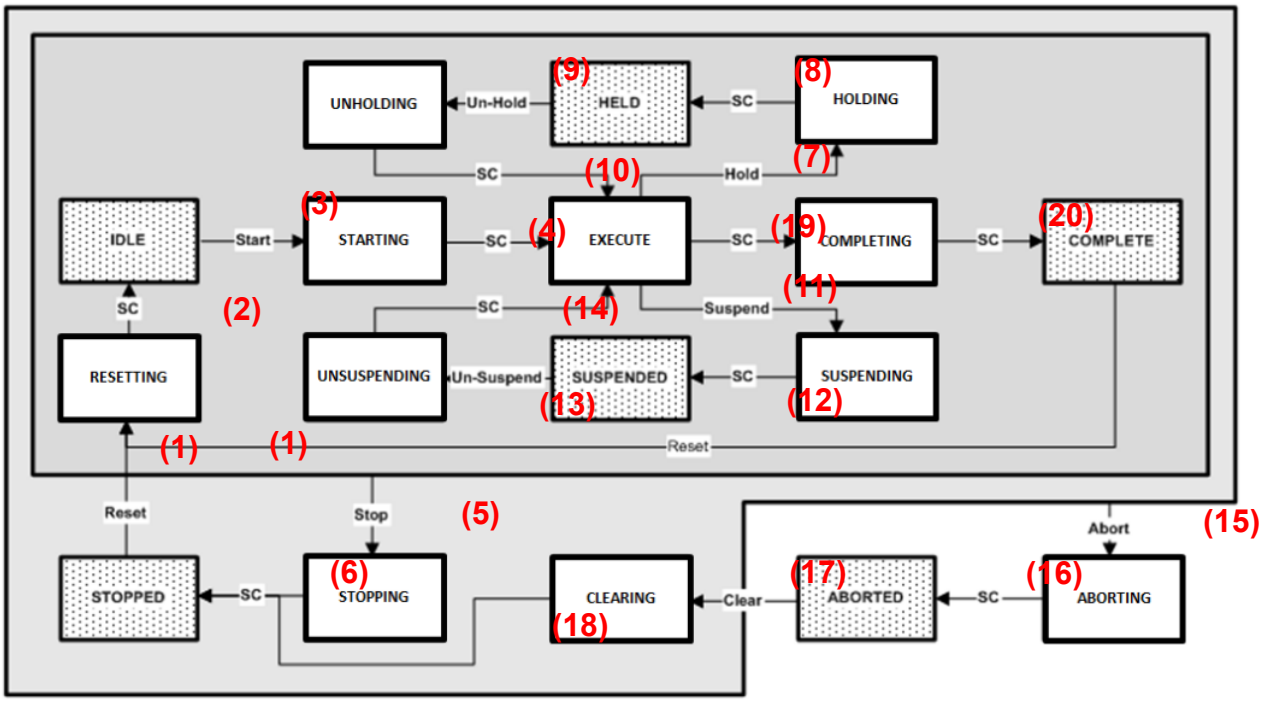
The following tables list the structures to be used for settings in each state for the PackML mode/state machine control function block that is described later.

Name		Data Type	Description
sPACKML_STATES_FLAG		STRUCT	The structure that retains a setting flag per state.
	CleaningState	BOOL	
	StoppedState	BOOL	
	StartingState	BOOL	
	IdleState	BOOL	
	SuspendedState	BOOL	
	ExecuteState	BOOL	
	StoppingState	BOOL	
	AbortingState	BOOL	
	AbortedState	BOOL	
	HoldingState	BOOL	
	HeldState	BOOL	
	UnholdingState	BOOL	
	SuspendingState	BOOL	
	UnssupendingState	BOOL	
	ResettingState	BOOL	
	CompletingState	BOOL	
	CompleteState	BOOL	

3.2.2. sPACKML_TRANSITION_COMMAND

Name		Data Type	Description
sPACKML_TRANSITION_COMMAND		STRUCT	The structure that indicates a transition for the PackML state machine.
	Cmd_Reset	BOOL	The command to execute the state transition from *Stopped* or *Complete* to *Resetting*. (1)
	Sts_Resetting_SC	BOOL	The request to execute the state transition from *Resetting* to *Idle*. (2)
	Cmd_Start	BOOL	The command to execute the state transition from *Idle* to *Starting*. (3)
	Sts_Starting_SC	BOOL	The request to execute the state transition from *Starting* to *Execute*. (4)

	Name	Data Type	Description
	Cmd_Stop	BOOL	The command to execute the state transition from *Idle, Resetting, Starting, Execute, Completing, Complete, Holding, Held, Unholding, Suspending, Suspended*, or *Unssupending* to *Stopping*. (5)
	Sts_Stopping_SC	BOOL	The request to execute the state transition from *Stopping* to *Stopped*. (6)
	Cmd_Hold	BOOL	The command to execute the state transition from *Execute* to *Holding*. (7)
	Sts_Holding_SC	BOOL	The request to execute the state transition from *Holding* to *Held*. (8)
	Cmd_UnHold	BOOL	The command to execute the state transition from *Held* to *UnHolding*. (9)
	Sts_UnHolding_SC	BOOL	The request to execute the state transition from *UnHolding* to *Execute*. (10)
	Cmd_Suspend	BOOL	The command to execute the state transition to *Execute* to *Suspending*. (11)
	Sts_Suspending_SC	BOOL	The request to execute the state transition from *Suspending* to *Suspended*. (12)
	Cmd_UnSuspend	BOOL	The command to execute the state transition from *Suspended* to *UnSuspending*. (13)
	Sts_UnSuspending_SC	BOOL	The request to execute the state transition from *UnSuspending* to *Execute*. (14)
	Cmd_Abort	BOOL	The command to execute the state transition from the state except *Aborting* and *Aborted* state, to *Aborting*. (15)
	Sts_Aborting_SC	BOOL	The request to execute the state transition from *Aborting* to *Aborted*. (16)
	Cmd_Clear	BOOL	The command to execute the state transition from *Aborted* to *Clearing*. (17)
	Sts_Clearing_SC	BOOL	The request to execute the state transition from *Clearing* to *Stopped*. (18)
	Sts_Execute_SC	BOOL	The command to execute the state transition from *Execute* to *Completing*. (19)
	Sts_Completing_SC	BOOL	The request to execute the state transition from *Completing* to *Complete*. (20)



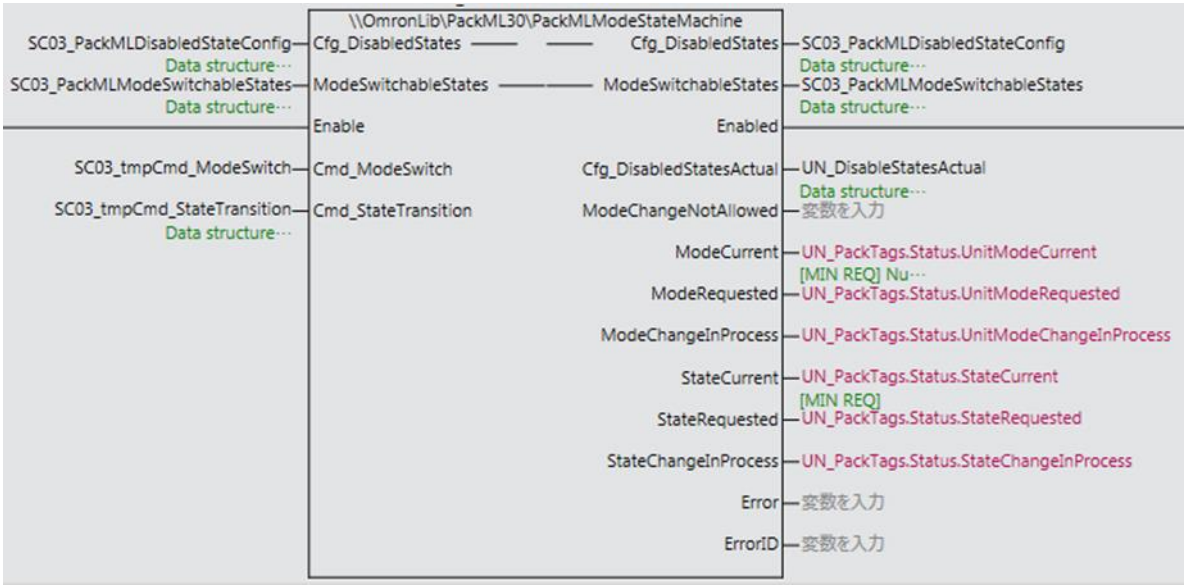
3.3. PackMLModeStateMachine Function Block

3.3.1. Provided Function

Based on the mode/state machine stipulated by PackML, the function block outputs the current mode and state according to the mode change/state transition command.

- Accepts unused state setting per mode specified.
- Sets the state in which the mode is switched.
- Switches the mode to the specified one if the internal state allows.
- Executes the specified state transition if the current state allows it. When more than one state transition commands are received at the same time, the state transitions are executed according to the proper priority.

■ Appearance of the function block (call representation example in ladder language)



3.3.2. Input Variable/Output Variable/Input and Output Variable

Name	In/Out	Data Type	Default	Description
Enable	Input	BOOL	FALSE	FB-enabled flag. Enables the FB function. Executing this in FALSE will clear the settings entered in the following.
Cfg_DisabledStates	In/Out	ARRAY[1..31] OF OmronLib¥PackML¥sPACKML_STATES_FLAG	-	Disabled-state settings. The unused states are specified per mode. The array index represents the mode number. When the set value is not adequate, the value is overwritten and corrected automatically. Set True to non-use states in each Mode
ModeSwitchablepStates	In/Out	ARRAY[1..31] OF OmronLib¥PackML¥sPACKML_STATES_FLAG	-	Mode Switch Permit Settings. The states where switching mode is permitted are specified per mode. The array index represents the mode number. Set true to the states chaning mode available.
Cmd_ModeSwitch	Input	DINT(0..31)	0	Mode Switch Command. The mode number is specified in order to change the mode. Nothing is executed when 0, the same number as the current mode,

				or the number out of range is specified.
Cmd_StateTransition	Input	OmronLib¥PackML¥sPACKML_TRANSITION_COMMAND	FALSE (All member)	Transition Command. The transition request to the state machine is specified. More than one transaction is specifiable. However, only an executable transaction for the current state is executed. When more than one transaction is executable, one transaction is executed according to the existing priority. When all transitions is set false, state-transition is not occurred.
Enabled	Output	BOOL	-	FB-enabled Flag Output. When TRUE, the effective values for the output variable of this FB are output.
Cfg_DisabledStatesActual	OutPut	ARRAY[1..31] OF OmronLib¥PackML¥sPACKML_STATES_FLAG	-	Non-use StateSetting Output Indicate the Non-use State setting in FB True is output in the Non-use state bits.
ModeChangeNotAllowed	Output	BOOL	-	Mode Change Prohibited Flag. The flag becomes TRUE when the condition to switch the mode is not met although the valid-mode-change command between 1 and 31 is entered.
ModeCurrent	Output	DINT(0..31)	-	The current mode number is output.
ModeRequested	Output	BOOL	-	The mode number that is currently requested to switch is output. *1

ModeChangeInProgress	Output	BOOL	-	It becomes TRUE only for 250ms after the mode is switched. *1
StateCurrent	Output	DINT(1..17)		The current state number is output. *2
StateRequested	Output	DINT(1..17)	-	The state number that is currently requested to execute the state transition is output. *1,2
StateChangeInProgress	Output	BOOL	-	It becomes TRUE only for 250ms after the state transition. *1
Error	Output	BOOL	-	Error flag. It is always FALSE because internal error never occurs in this FB.
ErrorIDEx	Output	DWORD	-	Error ID It is always 0 (normal) because internal error never occurs in this FB.

3.3.3. Mode/State Machine Settings

<Mode/State Machine Settings>

- Configure the PackML mode/State machine based on the specifications of the packaging machine application you create. Specify it by using the In/Out variables as follows:
  `Cfg_ModeStateConfiguration: ARRAY[1..31] OF OmronLib¥PackML¥sPACKML_MODE_STATE_CONFIG`
  `The above array index represents the mode number.`
  - Per mode, set the unused states.
  - Per mode, set the states in which the mode is permitted to be switched.

- The In/Out variable, Cfg_ModeStateConfiguration, of setting the mode/state machine is evaluated and set only in the rising edge of Enable Input. Afterwards, the mode/state machine settings are retained regardless of the In/Out variable value while the cyclic execution keeps ongoing. To change the settings, you need to reset the Enable Input of the function block to FALSE, and then execute it again after setting TRUE for Enable.

- When the specified settings are not adequate, the settings are modified to be adequate inside the function block, and then the modified settings are updated into the mode/state machine. The modification result is written back to the In/Out variable, Cfg_ModeStateConfiguration.

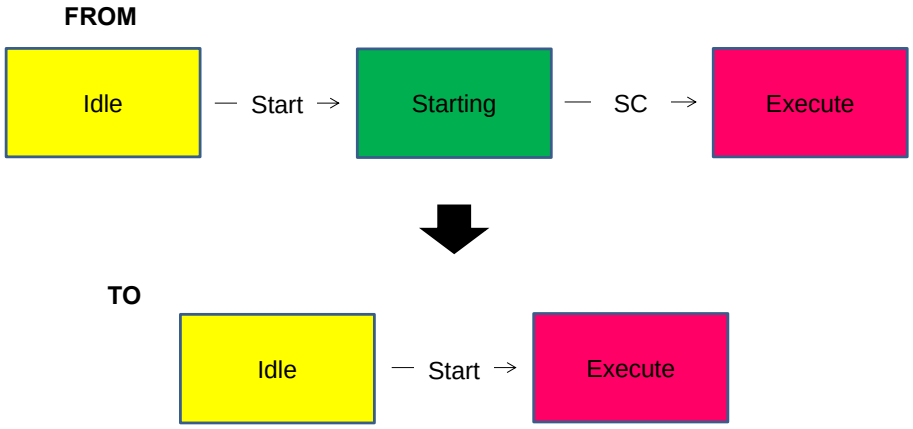
<Unused State Settings>

To configure the state to be used per mode, set TRUE for the BOOL member of the unused state by using Cmd_StateTransition[modeNumber]. StatesDisabled.

- Because the following states are essential for PackML v3.0, you cannot set them the unused state. The In/Out variables are modified automatically.

  - Stopped
  - Idle
  - Execute
  - Aborted

- When a state is not used, the command that transitions a state to the unused state will transition the state to the next state unconditionally. The following example shows the state transition when the *Starting* state is not used.



- When the Wait states except for *Stopped*, *Idle, Execute,* and *Aborted* are set as the unused states, the related *Acting* (…ing) states are automatically set as the unused states accordingly.

State set as “DisabeleState”	State as “unused State” automatically
Resetting	-
Starting	-
Suspending	-
Unsuspending	-
Holding	-
UnHolding	-
Completing	-
Aborting	-
Cleaning	-
Idle	-
Held	Holding,UnHolding
Suspend	Suspending,Unsuspending
Complete	Completing
Stopped	-
Aborted	-
Execute	-

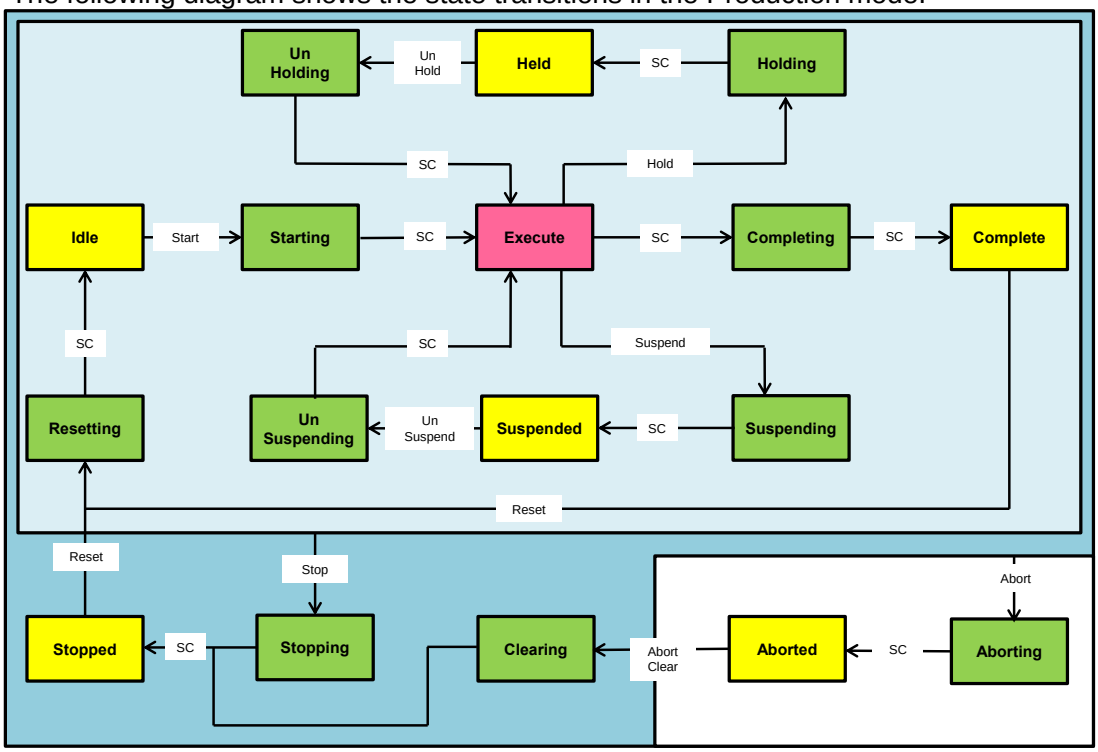
Here describes the typical modes such *as “Production Mode”, “Manual Mode”, “Maintenance Mode”* defined by Pack_ML.

【Example of Mode Settings】

① Production Mode (Mode Number: 1)

Production Mode is for production that is repeated by a machine. In this mode, the command that is directly entered by an operator or output by other monitoring-system starts up a machine. (e.g., Running/Normal Operating)

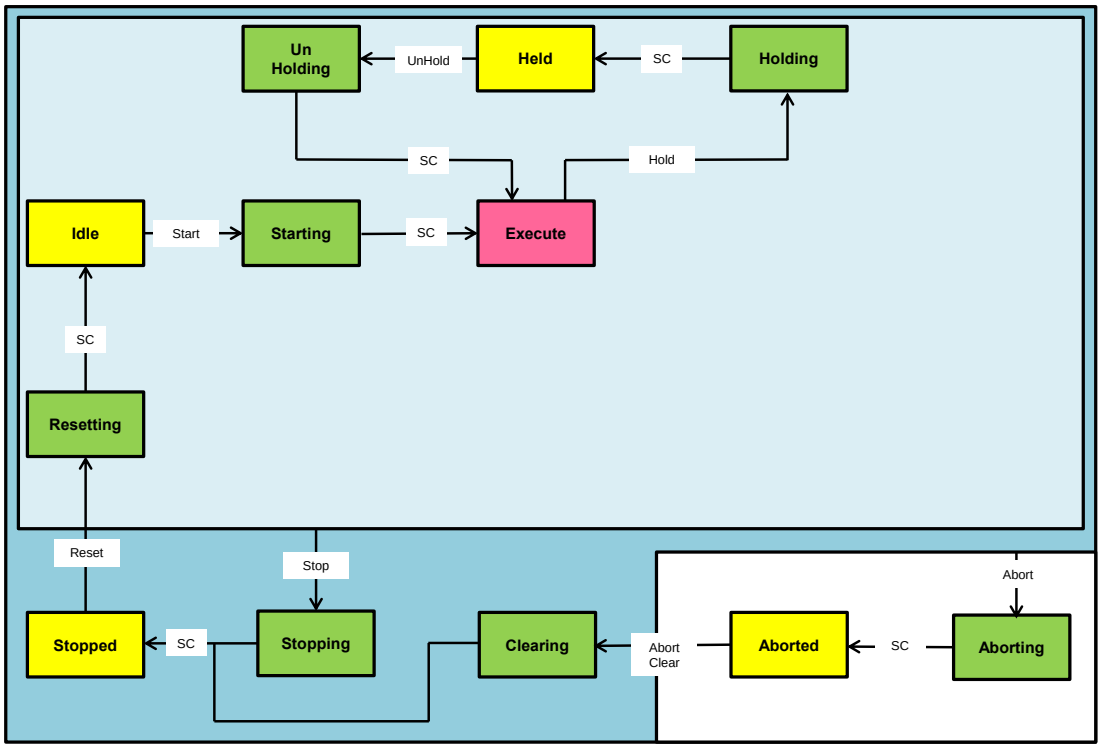
The following diagram shows the state transitions in the Production mode.



	State Commands									
Current State	Start	Reset	Hold	UnHold	Suspend	UnSuspend	AbortClear	Stop	Abort	State Complete
Idle	Staring	-	-	-	-	-	-	Stopping	Aborting	-
Starting	-	-	-	-	-	-	-	Stopping	Aborting	Execute
Execute	-	-	Holding	-	Suspending	-	-	Stopping	Aborting	Completing
Completing	-	-	-	-	-	-	-	Stopping	Aborting	Complete
Complete	-	Resetting	-	-	-	-	-	Stopping	Aborting	-
Resetting	-	-	-	-	-	-	-	Stopping	Aborting	Idle
Holding	-	-	-	-	-	-	-	Stopping	Aborting	Held
Held	-	-	-	UnHolding	-	-	-	Stopping	Aborting	-
UnHolding	-	-	-	-	-	-	-	Stopping	Aborting	Execute
Suspending	-	-	-	-	-	-	-	Stopping	Aborting	Suspended
Suspended	-	-	-	-	-	UnSuspending	-	Stopping	Aborting	-
UnSuspending	-	-	-	-	-	-	-	Stopping	Aborting	Execute
Stopping	-	-	-	-	-	-	-	-	Aborting	Stopped
Stopped	-	Resetting	-	-	-	-	-	-	Aborting	-
Aborting	-	-	-	-	-	-	-	-	-	Aborted
Aborted	-	-	-	-	-	-	Clearing	-	-	-
Clearing	-	-	-	-	-	-	-	-	Aborting	Stopped

* “-“ should be ignored.

②Maintenance Mode



In this mode, only permitted persons are allowed to operate individual machines in the production line. This mode is effective to detect machine defects, make a trial run, and test a machine.

The following table shows the state transitions in the Maintenance Mode.

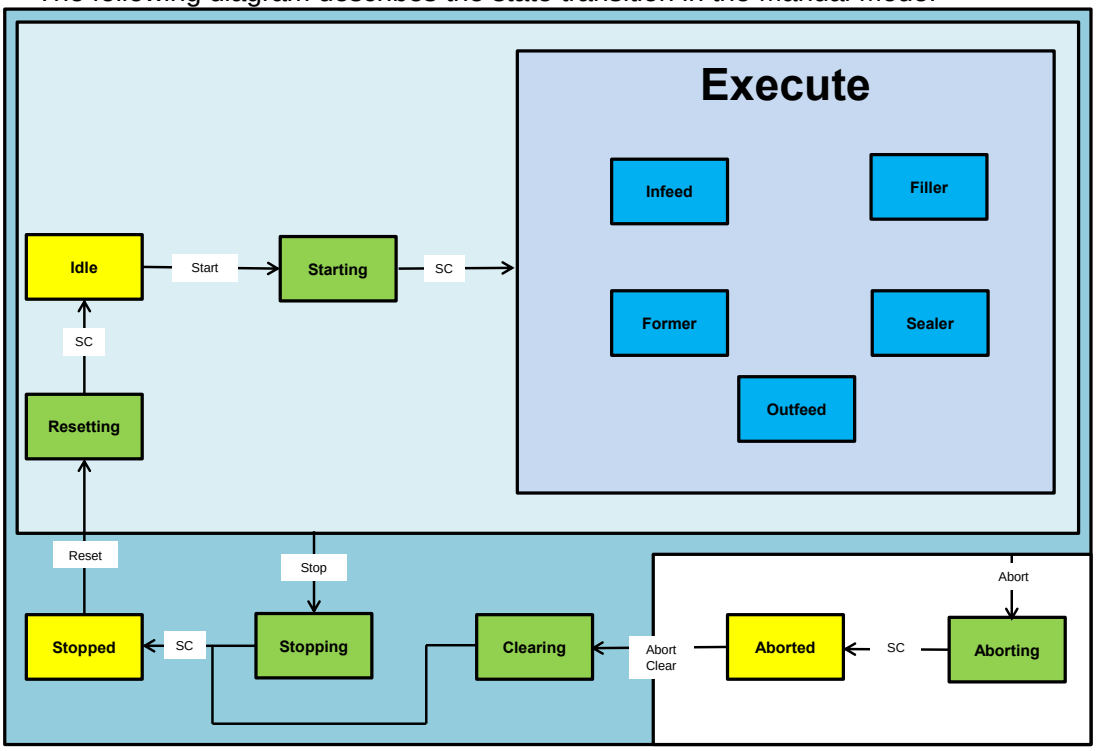
	State Commands									
Current State	Start	Reset	Hold	UnHold	Suspend	UnSuspend	AbortClear	Stop	Abort	State Complete
Idle	Staring	-	-	-	-	-	-	Stopping	Aborting	-
Starting	-	-	-	-	-	-	-	Stopping	Aborting	Execute
Execute	-	-	Holding	-	-	-	-	Stopping	Aborting	Completing
Completing	-	-	-	-	-	-	-	-	-	-
Complete	-	-	-	-	-	-	-	-	-	-
Resetting	-	-	-	-	-	-	-	Stopping	Aborting	Idle
Holding	-	-	-	-	-	-	-	Stopping	Aborting	Held
Held	-	-	-	UnHolding	-	-	-	Stopping	Aborting	
UnHolding	-	-	-	-	-	-	-	Stopping	Aborting	Execute
Suspending	-	-	-	-	-	-	-	-	-	-
Suspended	-	-	-	-	-	-	-	-	-	-
UnSuspending	-	-	-	-	-	-	-	-	-	-
Stopping	-	-	-	-	-	-	-	-	Aborting	Stopped
Stopped	-	Resetting	-	-	-	-	-	-	Aborting	-
Aborting	-	-	-	-	-	-	-	-	-	Aborted
Aborted	-	-	-	-	-	-	Clearing	-	-	-
Clearing	-	-	-	-	-	-	-	-	Aborting	Stopped

* “-“ should be ignored.

③Manual Mode

In this mode, only permitted persons are allowed to operate a machine directly (e.g., jogging).

The following diagram describes the state transition in the Manual Mode.



∗ The states in *Execute* are defined by the user. They are not processing to be done by this FB.

	State Commands									
Current State	Start	Reset	Hold	UnHold	Suspend	UnSuspend	AbortClear	Stop	Abort	State Complete
Idle	Staring	-	-	-	-	-	-	Stopping	Aborting	-
Starting	-	-	-	-	-	-	-	Stopping	Aborting	Execute
Execute	-	-	-	-	-	-	-	Stopping	Aborting	-
Completing	-	-	-	-	-	-	-	-	-	-
Complete	-	-	-	-	-	-	-	-	-	-
Resetting	-	-	-	-	-	-	-	Stopping	Aborting	Idle
Holding	-	-	-	-	-	-	-	-	-	-
Held	-	-	-	-	-	-	-	-	-	-
UnHolding	-	-	-	-	-	-	-	-	-	-
Suspending	-	-	-	-	-	-	-	-	-	-
Suspended	-	-	-	-	-	-	-	-	-	-
UnSuspending	-	-	-	-	-	-	-	-	-	-
Stopping	-	-	-	-	-	-	-	-	Aborting	Stopped
Stopped	-	Resetting	-	-	-	-	-	-	Aborting	-
Aborting	-	-	-	-	-	-	-	-	-	Aborted
Aborted	-	-	-	-	-	-	Clearing	-	-	-
Clearing	-	-	-	-	-	-	-	-	Aborting	Stopped

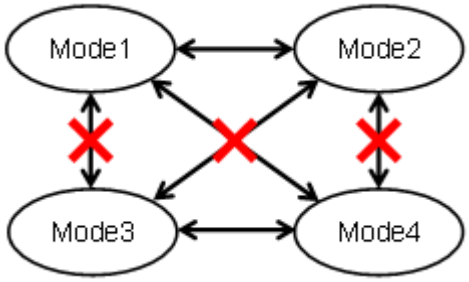
* "-" should be ignored.

<Permit Mode Switch Setting >

To specify the permit mode switch, set TRUE for the BOOL member of the state, which is permitted to switch, by using Cmd_StateTransition[modeNumber].StatesModeSwitchable

- When this flag is in the TRUE state in the current mode, the mode is switchable to the mode which flag is TRUE.

- With this function block, the mode is switchable to all of the modes in which the permit mode switch flag is set. A function that allows a mode to be switched between specific modes is not supported.
  Add required interlock logic outside of this function block if you need an application that allows a mode to be switched between specific modes in a specific state, and to control a mode to be switched between the specific modes as the following diagram shows.



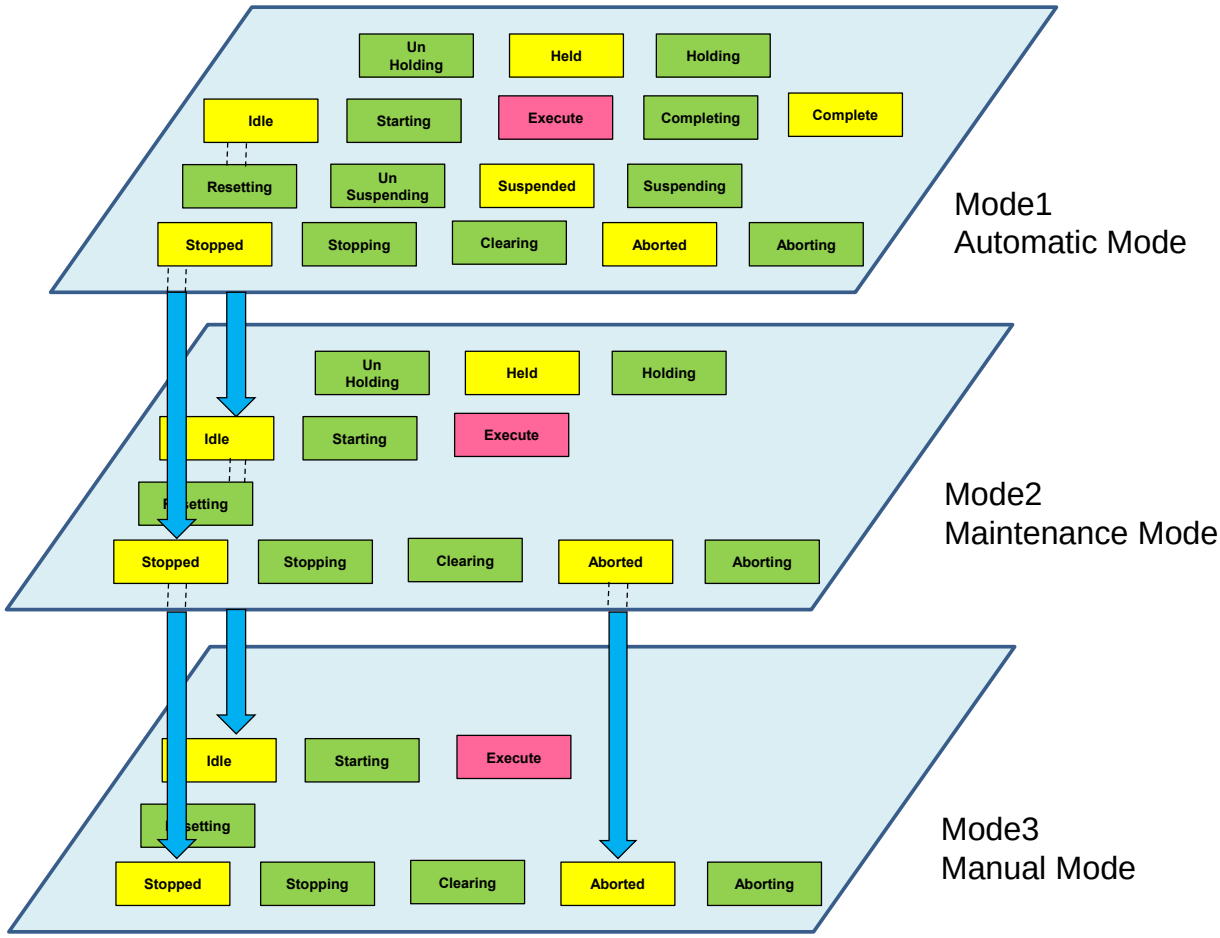
There are two ways of interlocking.

① Based on the current mode, change dynamically the settings of the In/Out variable, *ModeSwitchableStates*.

② Based on a specific mode, limit the values to be set for the In/Out variable, *Cmd_StateTransition*.

The following diagram shows an example of mode switch settings.

Mode1 to 3 support the foregoing *"Production", "Maintenance"*, and *"Manual"* modes respectively. The states in which the mode is switched between Mode 1 and 2 are set for "*Idle*" and "*Stopped*". The states in which the mode is switched between Mode2 and 3 are set for *"Idle", "Stopped",* and *"Aborted"*.



ModeSwitchableStates, which sets the above, is set as follows:
(The initial value of each member of sPACKML_STATES_FLAG shall be FALSE.)

```
ModeSwitchableStates[1].StoppedState := TRUE;
ModeSwitchableStates[2].StoppedState := TRUE;
ModeSwitchableStates[3].StoppedState := TRUE;

ModeSwitchableStates[1].IdleState := TRUE;
ModeSwitchableStates[2].IdleState := TRUE;
ModeSwitchableStates[3].IdleState := TRUE;

ModeSwitchableStates[2].AbortedState := TRUE;
ModeSwitchableStates[3].AbortedState := TRUE;
```

3.3.1. State Transition

- In the state transition command, Cmd_StateTransition, more than one transaction request flags can be set TRUE. However, transition requests which is not executable for the current state is ignored. Even if no executable transaction request is included, it is simply ignored without any error.
- When the state transition command, Cmd_StateTransition, includes more than one

executable transaction request, only one state transition is executed in the order of Cmd_Abort, Cmd_Stop, Cmd_xxx, and Sts_xxx_SC.

- When a valid mode switch command and a valid state transaction command are entered at the same time, the state transition command is executed first.

3.3.2. Mode Switch Function

- When the value of the mode switch command, Cmd_ModeSwitch, is not the valid value (1 to 31), it is considered as a request of switching the mode.
- When 0 or a mode number out of the range (32 or larger) is specified, the number is ignored, but it is not an error.
- When the mode switch command and a valid state transition command are given at the same time, the state transition is executed first.

3.3.3. Error and Error Code

This function block does not output error.

Even if invalid values are given to the input variables with Enable input TRUE, this function block ignores such inputs, always outputs the valid values and always set Enabled output TRUE.

3.4. PackMLModeStateTimer Function Block

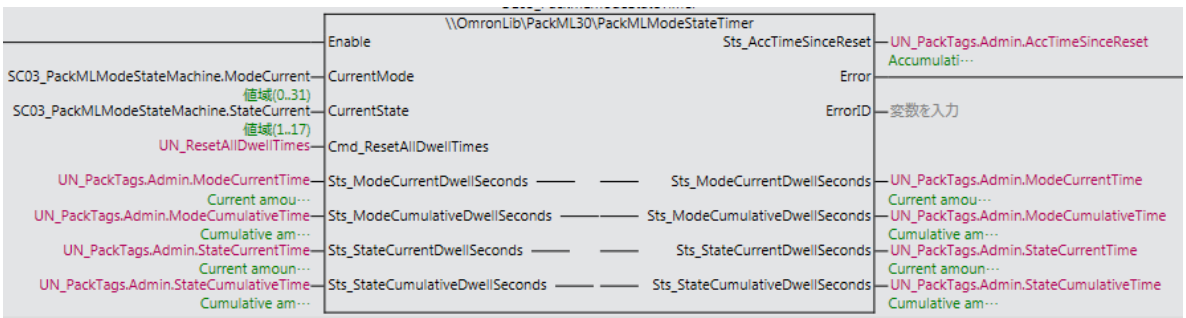
3.4.1. Provided Function

The function block measures and outputs a dwell time (second) in each state and mode of the mode/state machine stipulated by PackML.

The function is mainly used for calculating the following tags of PackTag.

- Admin.ModeCurrentTime
- Admin.StateCurrentTime
- Admin.AccTimeSinceReset
- Admin.ModeCumulativeTime
- Admin.StateumlativeTime

■Appearance of Function Block (call representation example in ladder language)



3.4.2. Input Variable/Output Variable/ In/Out Variable

Name	In/Out	Data Type	Default	Description
Enable	Input	BOOL	FALSE	FB-enabled flag. Enables this FB.
CurrentMode	Input	DINT(0..31)	1	Specifies the current mode number.
CurrentState	Input	DINT(1..17)	1	Specifies the current state number.
Cmd_ResetAllDwellTimes	Input	BOOL	FALSE	The command to reset the accumulated time. When it is executed at TRUE, accumulated dwell seconds in each mode and state are reset to 0.
Sts_ModeCurrentDwellSeconds	In/Out	ARRAY[1..31] OF DINT	-	Outputs dwell seconds in the current mode. Outputs seconds that have elapsed after the mode was switched to the current mode. Actually it is output but the variable is the In/Out variable in consideration of performance at the time of execution. For the array that has the mode number in the index, the values of the modes except for the current mode are 0.
Sts_ModeCumulativeDwellSeconds	In/Out	ARRAY[0..31] OF DINT	-	Outputs accumulated dwell seconds in each mode. Outputs accumulated dwell seconds in the current mode after the last reset. Actually it is output but the variable is the In/Out variable in consideration of performance at the time of execution. The array that has the mode number in the index.

Sts_StateCurrentDwellSeconds	In/Out	ARRAY[1..31,1..17] OF DINT	-	Outputs accumulated dwell seconds in the current state. Outputs seconds that have elapsed after the mode was transitioned to the current mode. Actually it is output but the variable is the In/Out variable on consideration of performance at the time of execution. For the array that has the mode number in the index. the values of the modes except for the current state are 0.
Sts_StateCumulativeDwellSeconds	In/Out	ARRAY[[1..31,1..17] OF DINT	-	Outputs the accumulated dwell seconds in the current state. Outputs the seconds that have elapsed after the state was transitioned to the current state. Actually it is output but the variable is the I/O variable in consideration of performance at the time of execution. For the array that has the state number in the index, the values of the modes except for the current state are 0.
Sts_AccTimeSinceReset	Output	DINT	-	The seconds that have elapsed after the last reset.
TimeRollOverWarning	Output	BOOL	-	When the seconds that have elapsed after the last reset exceed 2,147,483,647 seconds, it becomes TRUE. When the value is TRUE, no valid value is output to *Sts_AccTimeSinceReset.*
Error	Output	BOOL	-	Error flag. It is always FALSE because internal

				error never occurs in this FB.
ErrorIDEx	Output	DWORD	-	Error ID It is always 0 (normal) because internal error never occurs in this FB.

3.4.3. Error and ErrorIDEx

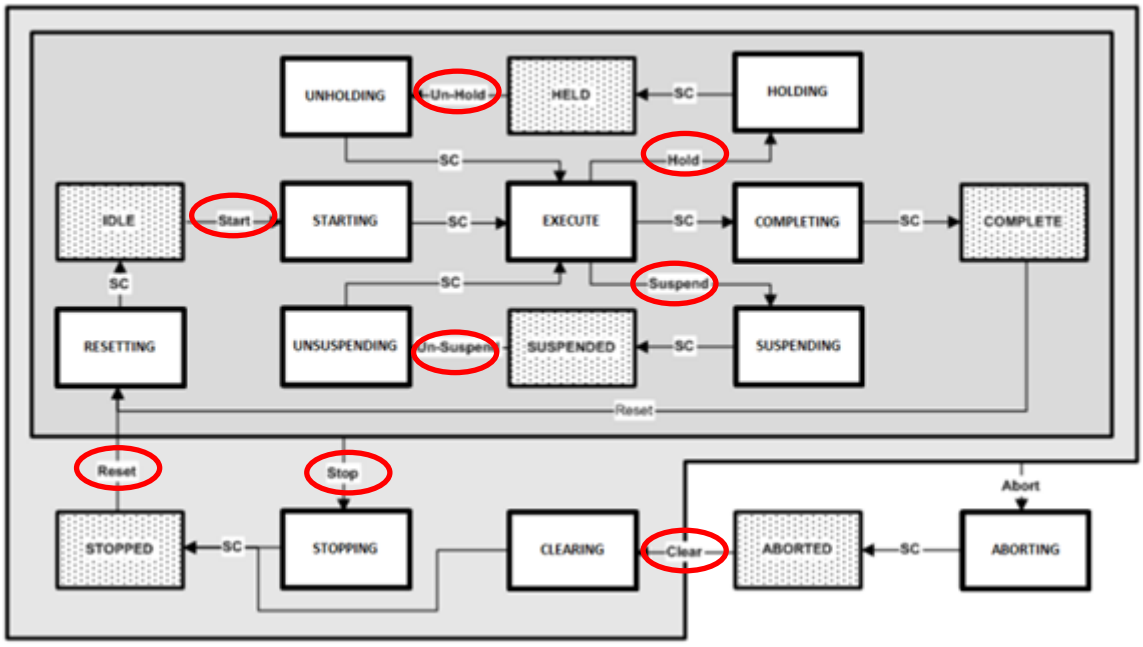
This FB doesn't have Error detect function, so it doesn't output Error and ErrorID.
(These functions are for future expansion.)

3.5. PMLCtrlCmd_<transition name> Function Blocks

3.5.1. Provided Functions

The functions will check which transaction command is the number of Command.CntrlCmd stipulated by PackTag. With these functions, the user no longer needs to see the specification to find out which transaction number actually represents which transaction.

- The following diagram shows the transactions and their numbers specified by Command.CntrlCmd.



0	Undefined
1	Reset
2	Start
3	Stop
4	Hold
5	Unhold
6	Suspend
7	Unsuspend
8	Abort
9	Clear

- The example of how to see which specified transaction command is the Stop command.

3.5.2. Function List

Name	Description
PMLCtrlCmd_Reset	TRUE is returned when the entered transition number is 1.
PMLCtrlCmd_Start	TRUE is returned when the entered transition number is 2.
PMLCtrlCmd_Stop	TRUE is returned when the entered transition number is 3.
PMLCtrlCmd_Hold	TRUE is returned when the entered transition number is 4.
PMLCtrlCmd_Unhold	TRUE is returned when the entered transition number is 5.
PMLCtrlCmd_Suspend	TRUE is returned when the entered transition number is 6.
PMLCtrlCmd_Unsuspend	TRUE is returned when the entered transition number is 7.
PMLCtrlCmd_Abort	TRUE is returned when the entered transition number is 8.
PMLCtrlCmd_Clear	TRUE is returned when the entered transition number is 9.

3.5.3. Input Variable/Output Variable/ In/Out Variable

Name	In/Out	Data Type	Default	Description
EN	Input	BOOL	TRUE	Function Execution Control Flag. When FALSE, internal logic is not executed even if it is called.
CtrlCmd	Input	DINT	0	Transition Command Number. Specifies the value obtained from the Command.CntrlCmd tag of

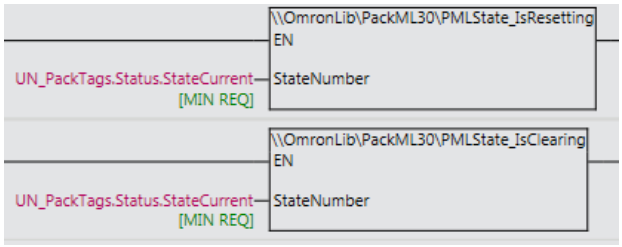
				PackTag. The range of input value is from 1 to 9.
<Function name>	Return value	BOOL	-	Return value. TRUE is returned only when the entered transition number represents the function name. (FALSE is returned when a value out of the range is entered.)

3.6. PMLState_Is <state name> Function Blocks

3.6.1. Provided Functions

The functions will check which state number stipulated by PackML represents which state. With these functions, the user no longer needs to see the specification to find out which state number, which is output by the PackML mode/state control function block, represents which state.

■ Appearance of the function (call representation example in ladder language)



3.6.2. Function List

Name	Description
PMLState_IsClearing	TRUE is returned when the entered state number is 1.
PMLState_IsStopped	TRUE is returned when the entered state number is 2.
PMLState_IsStarting	TRUE is returned when the entered state number is 3.
PMLState_IsIdle	TRUE is returned when the entered state number is 4.
PMLState_IsSuspended	TRUE is returned when the entered state number is 5.
PMLState_IsExecute	TRUE is returned when the entered state number is 6.
PMLState_IsStopping	TRUE is returned when the entered state number is 7.

PMLState_IsAborting	TRUE is returned when the entered state number is 8.
PMLState_IsAborted	TRUE is returned when the entered state number is 9.
PMLState_IsHolding	TRUE is returned when the entered state number is 10.
PMLState_IsHeld	TRUE is returned when the entered state number is 11.
PMLState_IsUnholding	TRUE is returned when the entered state number is 12.
PMLState_IsSuspending	TRUE is returned when the entered state number is 13.
PMLState_IsUnsuspending	TRUE is returned when the entered state number is 14.
PMLState_IsResetting	TRUE is returned when the entered state number is 15.
PMLState_IsCompleting	TRUE is returned when the entered state number is 16.
PMLState_IsComplete	TRUE is returned when the entered state number is 17.

3.6.3. Input Variable/Output Variable/ In/Out Variable

All of the functions have the same input and output variables.

Name	In/Out	Data Type	Default	Description
EN	Input	BOOL	TRUE	Function Control Flag. When FALSE, internal logic does not operate even if it is called.
StateNumber	Input	DINT(1..17)	0	State number. Specifies the state number to check.
<the same with the function name>	Return value	BOOL	-	Return value. TRUE is returned only when the entered state number represents the function name. (FALSE is returned when a state number out of the range is entered.)

4. PackML Module Command POU

4.1. Structure Definition

4.1.1. sPACKML_MODULE_COMMAND

Based on the mode/state machine stipulated by PackML, this structure retains the external interface that each module (EM and CM) should have.

Name		Data Type	Description
sPACKML_MODULE_COMMAND		STRUCT	
	In_ModuleActive	BOOL	The external command to enable/disable the module.
	Out_InternalTransitionRequest	OmronLib¥PackML¥ sPACKML_TRANSITION_COMMAND	The internal command that requests the state transition to the host module.

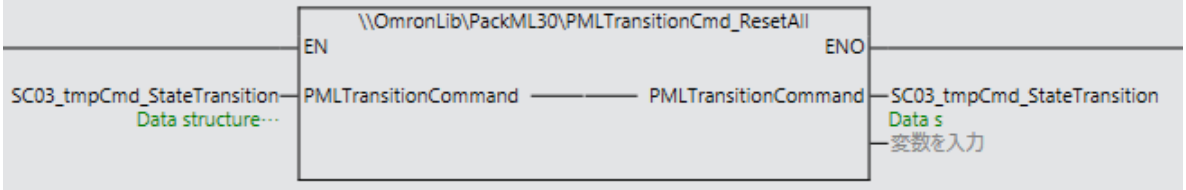
4.2. PMLTransitionCmd_ResetAll Function

4.2.1. Provided Function

For the state transition command sPACKML_TRANSITION_COMMAND structure-type variables, this function resets every BOOL member that indicates the state transition to FALSE.

This function is used for initializing the state transition request to the host module.

■ Appearance of the function (call representation example in ladder language)



4.2.2. Input Variable/Output Variable/ In/Out Variable

Name	In/Out	Data Type	Default	Description
EN	Input	BOOL	TRUE	Function Execution Control Flag. When FALSE, internal logic does not operate even if it is called.
ENO	Output	BOOL	-	Function Execution Control Flag Output
PMLTransitionCommand	In/Out	OmronLib¥PackML¥ sPACKML_TRANSITION_COMMAND	-	PackML state transition command.

4.2.3. Internal Processing

Internal processing of this function is as follows:

```
PMLTransitionCommand.Cmd_Abort := FALSE;
PMLTransitionCommand.Cmd_Clear := FALSE;
PMLTransitionCommand.Cmd_Hold := FALSE;
PMLTransitionCommand.Cmd_Reset := FALSE;
PMLTransitionCommand.Cmd_Start := FALSE;
PMLTransitionCommand.Cmd_Stop := FALSE;
PMLTransitionCommand.Cmd_Suspend:= FALSE;
PMLTransitionCommand.Cmd_UnHold := FALSE;
PMLTransitionCommand.Cmd_UnSuspend := FALSE;
```

```
PMLTransitionCommand.Sts_Aborting_SC := FALSE;
PMLTransitionCommand.Sts_Clearing_SC := FALSE;
PMLTransitionCommand.Sts_Completing_SC := FALSE;
PMLTransitionCommand.Sts_Execute_SC:= FALSE;
PMLTransitionCommand.Sts_Holding_SC:= FALSE;
PMLTransitionCommand.Sts_Resetting_SC:= FALSE;
PMLTransitionCommand.Sts_Starting_SC:= FALSE;
PMLTransitionCommand.Sts_Stopping_SC:= FALSE;
PMLTransitionCommand.Sts_Suspending_SC:= FALSE;
PMLTransitionCommand.Sts_UnHolding_SC:= FALSE;
PMLTransitionCommand.Sts_UnSuspending_SC:= FALSE;
```

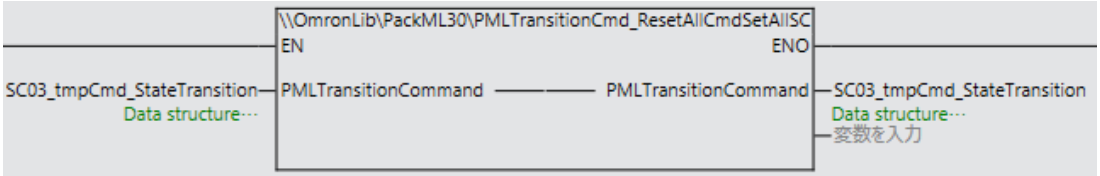
4.3. PMLTransitionCmd_ResetAllCmdSetAllSC Function

4.3.1. Provided Function

For the state transition command sPACKML_TRANSITION_COMMAND structure-type variables, this function resets every state transition command (Cmd_<state name) of the BOOL members that represent the state transitions, and resets every *Wait* state completion notification (Sts_<state name>_SC).

This function is used for initializing the state transition request to the host module.

■ Appearance of the function (call representation example in ladder language)



4.3.2. Input Variable/Output Variable/ In/Out Variable

Name	In/Out	Data Type	Default	Description
EN	Input	BOOL	TRUE	Function Execution Control Flag. When FALSE, internal logic does not operate even if it is called.
ENO	Output	BOOL	-	Function Execution Control Flag Output.
PMLTransitionCommand	In/Out	OmronLib¥PackML¥ sPACKML_TRANSITION_COMMAND	-	PackML state transition command.

4.3.1. Internal Processing

Internal processing of this function is as follows:

```
PMLTransitionCommand.Cmd_Abort := FALSE;
PMLTransitionCommand.Cmd_Clear := FALSE;
PMLTransitionCommand.Cmd_Hold := FALSE;
PMLTransitionCommand.Cmd_Reset := FALSE;
PMLTransitionCommand.Cmd_Start := FALSE;
PMLTransitionCommand.Cmd_Stop := FALSE;
PMLTransitionCommand.Cmd_Suspend:= FALSE;
PMLTransitionCommand.Cmd_UnHold := FALSE;
```

```
PMLTransitionCommand.Cmd_UnSuspend := FALSE;

PMLTransitionCommand.Sts_Aborting_SC := TRUE;
PMLTransitionCommand.Sts_Clearing_SC := TRUE;
PMLTransitionCommand.Sts_Completing_SC := TRUE;
PMLTransitionCommand.Sts_Execute_SC:= TRUE;
PMLTransitionCommand.Sts_Holding_SC:= TRUE;
PMLTransitionCommand.Sts_Resetting_SC:= TRUE;
PMLTransitionCommand.Sts_Starting_SC:= TRUE;
PMLTransitionCommand.Sts_Stopping_SC:= TRUE;
PMLTransitionCommand.Sts_Suspending_SC:= TRUE;
PMLTransitionCommand.Sts_UnHolding_SC:= TRUE;
PMLTransitionCommand.Sts_UnSuspending_SC:= TRUE;
```

4.4. PMLTransitionCmd_Summarize Function

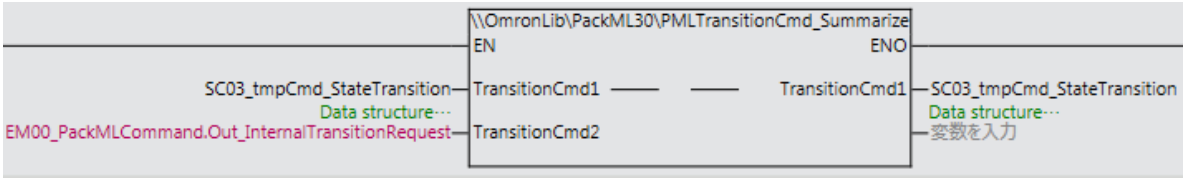
4.4.1. Provided Function

State transition requests are merged for the host module by processing the state transition requests (sPACKML_TRANSITION_COMMAND structure-type variables) arose from the lower modules as described below:

- Execute OR evaluation on State transition commands (Cmd_<state name>)
- Execute AND evaluation on *Wait* state completion notifications (Sts_<state name>_SC)

This function is used for merging each of state transition requests of CM below EM into the state transitions of EM, and for merging each of state transition requests of EM into the state transition requests of UN.

■ Appearance of the function (call representation example in ladder language)



4.4.2. Input Variable/Output Variable/ In/Out Variable

Name	In/Out	Data Type	Default	Description
EN	Input	BOOL	TRUE	Function Execution Control Flag. When FALSE, internal logic does not operate even if it is called.
ENO	Output	BOOL	-	Function Execution Control Flag Output.

Transition1	In/Out	OmronLib¥PackML¥ sPACKML_TRANSITINO_COMMAND	-	Specifies transition command which Transition2 are to be merged into.
Transition2	Input	OmronLib¥PackML¥ sPACKML_TRANSITINO_COMMAND	FALSE (All member)	Specifies a state transition request to be merged into Transition1.

4.4.3. Internal Processing

```
//OR evaluation for the transition rquest of cmd_
TransitionCmd1.Cmd_Abort := TransitionCmd1.Cmd_Abort OR
TransitionCmd2.Cmd_Abort;
TransitionCmd1.Cmd_Clear := TransitionCmd1.Cmd_Clear OR
TransitionCmd2.Cmd_Clear;
TransitionCmd1.Cmd_Hold := TransitionCmd1.Cmd_Hold OR
TransitionCmd2.Cmd_Hold;
TransitionCmd1.Cmd_Reset := TransitionCmd1.Cmd_Reset OR
TransitionCmd2.Cmd_Reset;
TransitionCmd1.Cmd_Start := TransitionCmd1.Cmd_Start OR
TransitionCmd2.Cmd_Start;
TransitionCmd1.Cmd_Stop := TransitionCmd1.Cmd_Stop OR
TransitionCmd2.Cmd_Stop;
TransitionCmd1.Cmd_Suspend := TransitionCmd1.Cmd_Suspend OR
TransitionCmd2.Cmd_Suspend;
TransitionCmd1.Cmd_UnHold := TransitionCmd1.Cmd_UnHold OR
TransitionCmd2.Cmd_UnHold;
TransitionCmd1.Cmd_UnSuspend := TransitionCmd1.Cmd_UnSuspend OR
TransitionCmd2.Cmd_UnSuspend;

//AND evaluation for the transition request of SC (state completed event)
TransitionCmd1.Sts_Aborting_SC := TransitionCmd1.Sts_Aborting_SC AND
TransitionCmd2.Sts_Aborting_SC;
TransitionCmd1.Sts_Clearing_SC := TransitionCmd1.Sts_Clearing_SC AND
TransitionCmd2.Sts_Clearing_SC;
TransitionCmd1.Sts_Completing_SC := TransitionCmd1.Sts_Completing_SC AND
TransitionCmd2.Sts_Completing_SC;
TransitionCmd1.Sts_Execute_SC := TransitionCmd1.Sts_Execute_SC AND
TransitionCmd2.Sts_Execute_SC;
TransitionCmd1.Sts_Holding_SC := TransitionCmd1.Sts_Holding_SC AND
TransitionCmd2.Sts_Holding_SC;
```

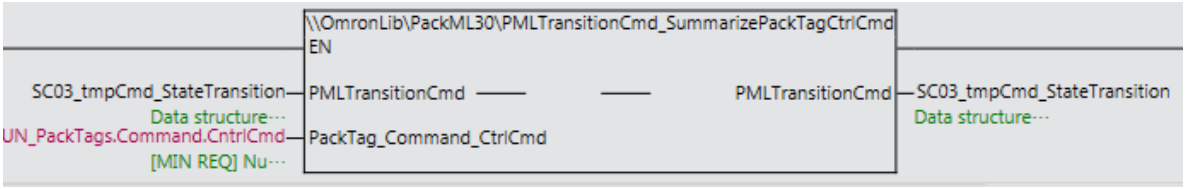
```
TransitionCmd1.Sts_Resetting_SC := TransitionCmd1.Sts_Resetting_SC AND
TransitionCmd2.Sts_Resetting_SC;
TransitionCmd1.Sts_Starting_SC := TransitionCmd1.Sts_Starting_SC AND
TransitionCmd2.Sts_Starting_SC;
TransitionCmd1.Sts_Stopping_SC := TransitionCmd1.Sts_Stopping_SC AND
TransitionCmd2.Sts_Stopping_SC;
TransitionCmd1.Sts_Suspending_SC := TransitionCmd1.Sts_Suspending_SC AND
TransitionCmd2.Sts_Suspending_SC;
TransitionCmd1.Sts_UnHolding_SC := TransitionCmd1.Sts_UnHolding_SC AND
TransitionCmd2.Sts_UnHolding_SC;
TransitionCmd1.Sts_UnSuspending_SC  :=  TransitionCmd1.Sts_UnSuspending_SC
AND TransitionCmd2.Sts_UnSuspending_SC;
```

4.5. PMLTransitionCmd_SummarizePackTagCtrlCmd Function

4.5.1. Provided Function

The function merges the state transition requests from outside of the machine through the Command.CntrlCmd tag of PackTag, and the state transition requests gained in the machine by merging the state transition requests from EM and CM below UN.

■ Appearance of the function block (call representation example in ladder language)



4.5.2. Input Variable/Output Variable/ In/Out Variable

Name	In/Out	Data Type	Default	Description
EN	Input	BOOL	TRUE	Function Execution Control Flag. When FALSE, internal logic does not operate even if it is called.
ENO	Output	BOOL	-	Function Execution Control Flag Output.
PMLTransitionCommand	In/Out	OmronLib¥PackML¥ sPACKML_TRANSITINO_COMMAND	-	Transition commands from outside of the machine are to be merged.

PackTag_Command_CtrlCmd	Input	DINT(1..9)	0	The transition requests gained by PackTag Command from outside of the machine. These gained state transitions shall be merged into PMLTransitionCommand.

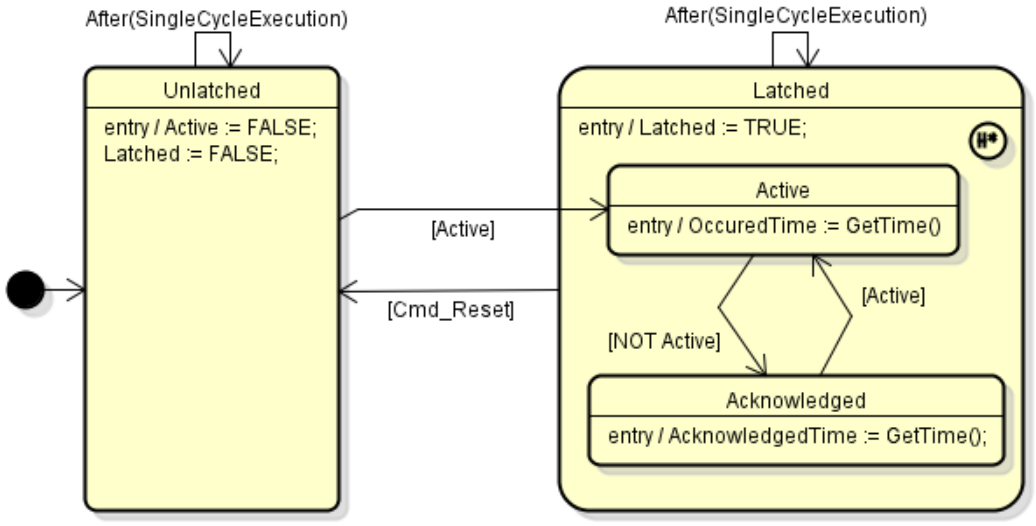
5. Alarm Control POU

5.1. Overview

5.1.1. Provided Function

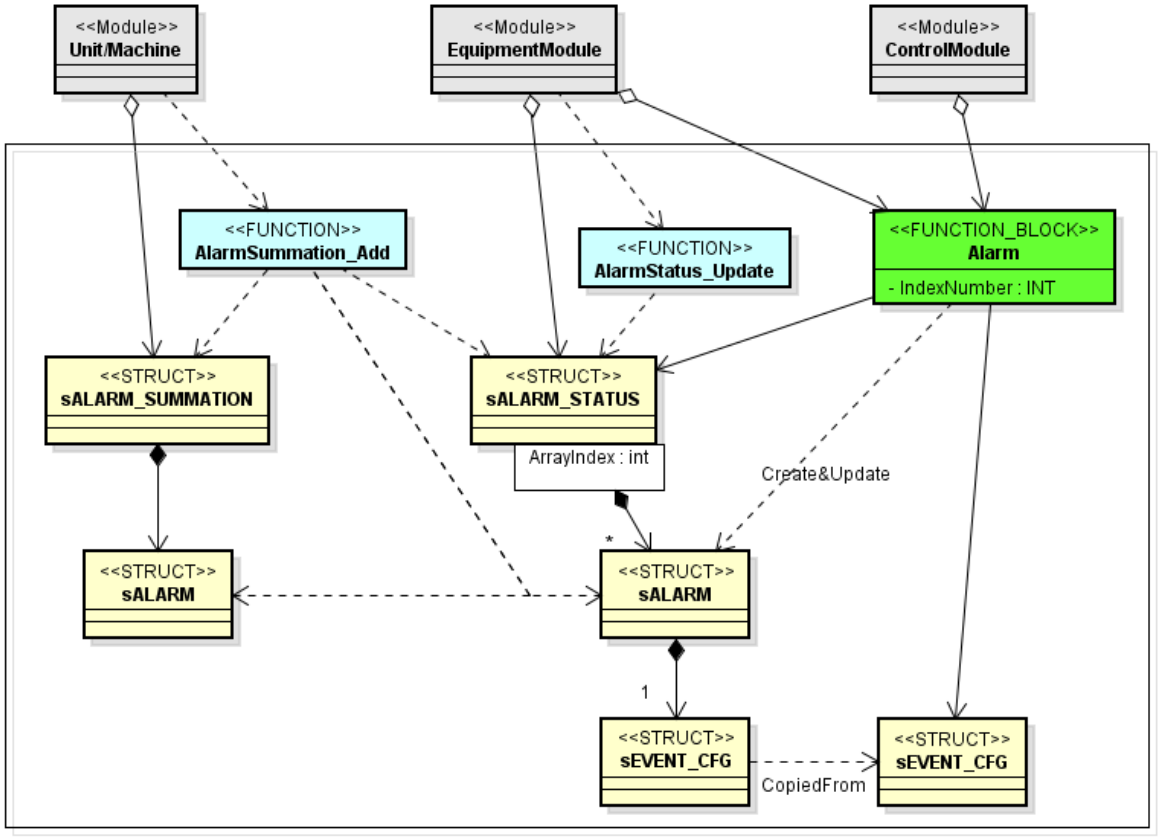
This function sends *Alarm* information of the module to the host module.

"*Alarm*" is a concept that supports an event. *Alarm* has two flags. The one is *Active* flag that indicates that a relevant event is currently occurring or not. The other one is *Latched* flag. Once the alarm becomes Active, the Latched flag is being flagged until reset explicitly. The Alarm conceptual state machine is as follows:



※Most of the competitors name this kind of function *Event control*. However, this provided function actually should be called *Alarm*. The latest TR88.00.02 Rev.2 (2015) also calls it Alarm. Thus, we employ the name of Alarm to POU.

5.1.2. Configuration



※ The configuration of POU and Structure employed here is based on the P&G implementation guide, Yaskawa, and Mitsubishi that are widely known as de facto standard.

5.2. Structure Definition

5.2.1. sEVENT_CFG

This is the structure that retains detail information of events that Alarm supports.

Name	Data Type	Description
sEVENT_CFG		The structure that defines the events to be used for sALARM.
ID	DINT	An identifier of event type
Value	DINT	Additional information of event type
Message	STRING[80]	Message to be indicated for event
Description	STRING[256]	Detail description of event type
Category	USINT(0..9)	Event category number

5.2.2. sALARM

This is the structure that represents a single Alarm.

Name		Data Type	Description
sALARM		STRUCT	The structure that represents a single Alarm.
	EventType	OmronLib¥PackML¥sEVENT_CFG	Event type supported by this Alarm
	OccuredTime	DATE_AND_TIME	Event occurrence time
	Active	BOOL	The flag that indicates whether this Alarm is active or not (not acknowledged yet).
	Latched	BOOL	The flag indicates that this Alarm used to be active after the last reset. (An related event fired.)
	AcknowledgedTime	DATE_AND_TIME	Date and Time when this Alarm became inactive.
	ReporterName	STRING[256]	Information, which shows a source of Alarm, for debugging.

5.2.3. sALARM_STATUS

This is the structure that merges the states of *Alarm* collected per EM (equipment module).

Name		Data Type	Description
sALARM_STATUS		STRUCT	The structure that shows the states of Alarm collected per equipment module (EM).
	Sts_FirstOutAlarm	OmronLib¥PackML¥sALARM	The snapshot of the first active Alarm.
	Sts_FirstOutAlarmByCategory	OmronLib¥PackML¥sALARM	The snapshot of the first active Alarm in each category.
	Sts_Alarms	ARRAY[0..29] OF OmronLib¥PackML¥sALARM	The array of Alarm collected by the equipment module.
	Sts_NumOfAlarms	UINT(0..30)	The actual size of the above Alarm array.
	Sts_CategoryActiveFlag	ARRAY[0..9] OF BOOL	The array of the flag that shows whether each category includes active Alarm or not. The array index represents the category number.
	Sts_CategoryLatchedFlag	ARRAY[0..9] OF BOOL	The array of the flag that shows whether each category includes Latched Alarm (Alarm has an evidence that it used to be active) or not. The array index represents

			the category number.
	NeedToBeUpdated	BOOL	The flag that shows the necessity of updating data by AlarmStatus_Update FB because the state of Sts_Alarm is updated by Alarm FB.
	NeedToBeSummarized	BOOL	The flag that shows the necessity of updating AlarmSummation.

5.2.4. sALARM_SUMMATION

The structure that merges *Alarms* collected from all EM below UN (unit/machine).

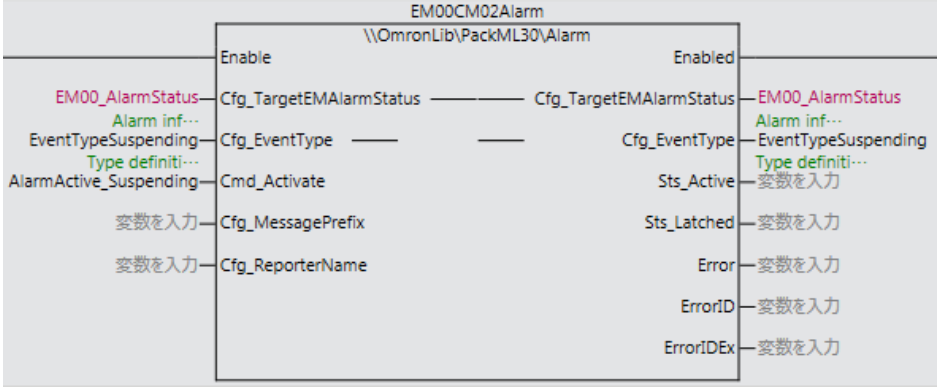
Name		Data Type	Description
sALARM_SUMMATION			Alarm status information collected by UN (machines).
	Sts_FirstOutAlarm	OmronLib¥PackML¥ sALARM	The first active Alarm.
	Sts_Alarms	ARRAY[0..100] OF OmronLib¥PackML¥ sALARM	The array of collected Alarms.
	Sts_NumOfAlarms	UINT(0..29)	The actual size of the above Alarm array.
	ActiveOneExists	BOOL	The array of the flag that shows whether each category includes active Alarm or not. The array index represents the category number.
	LatchedOneExists	BOOL	The array of the flag that shows whether each category includes Latched Alarm (Alarm has an evidence that it used to be active) or not. The array index represents the category number.
	Sts_CategoryActiveFlag	ARRAY[0..9] OF BOOL	The first active Alrm.
	Sts_CategoryLatchedFlag	ARRAY[0..9] OF BOOL	The array of Alarms collected by the equipment module.

5.3. Alarm Function Block

5.3.1. Provided Function

This function defines *"Alarm"* to support events. This function reports the state of the defined *Alarm* to *sALARM_STATUS* structure-type variables under the host module control.

■ Appearance of the function block (call representation example in ladder language)



5.3.2. Input Variable/Output Variable/ In/Out Variable

Name	In/Out	Data Type	Default	Description
Enable	Input	BOOL	FALSE	FB-enabled flag. Enables this FB. At FALSE, nothing executes.
Cfg_TargetEMAlarmStatus	In/Out	OmronLib¥PackML¥ sALARM_STATUS	-	Report destination alarm status. Specifies *sAlarmStatus* variables to which this Alarm status is reported. *Once it is set, it shall not be changed.
Cfg_EventType	In/Out	OmronLib¥PackML¥ sEVENT_CFG	-	Event type. Specifies the event type to be supported as *Alarm*. *Once it is set, it shall not be changed.
Cmd_Activate	Input	BOOL	FALSE	Alarm activation flag input. Sets TRUE when *Alarm* is activated after the Event occurs. To reset, sets FALSE.
Cfg_MessagePrefix	Input	STRING[10]	''	Alarm message prefix. When reporting *Alarm*, specifies a prefix that should be attached to the message specified by

				Cfg_EventType.
Cfg_ReporterName	Input	STRING[100]	‘’	Report source name. Specifies the necessary name in order to identify the *Alarm* report source. (for debugging)
Enabled	Output	BOOL	-	FB-enabled flag output. It becomes TRUE when *Enable* becomes TRUE and this FB is operating normally.
Sts_Active	Output	BOOL	-	Alarm Activation Flag Output. It becomes TRUE when this *Alarm* is activated.
Sts_Latched	Output	BOOL	-	Alarm Latch Flag Output. When this Alarm is activated, it becomes TRUE. Even after being reset, it retains TRUE. When it is reset by *AlarmStatus_Update* function, it goes back to FALSE.
Error	Output	BOOL	-	Output Error It is always 0 (normal) because internal error never occurs in this FB.
ErrorID	Output	BOOL	-	Output ErrorID It is always 0 (normal) because internal error never occurs in this FB.
ErrorIDEx	Output	BOOL	-	Output ErrorIDEx It is always 0 (normal) because internal error never occurs in this FB.

5.3.1. Function details

This function block takes the following internal actions.

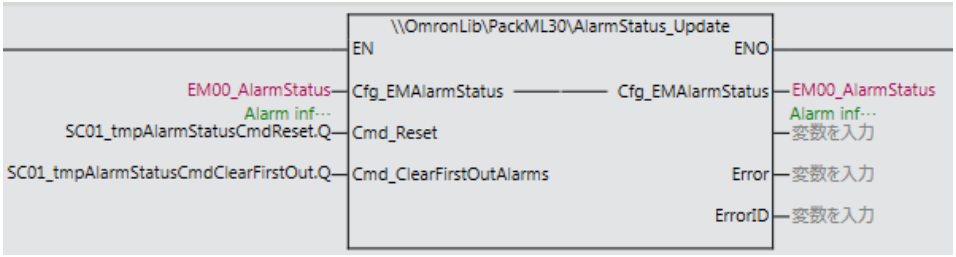
- At first execution, this FB checks *Sts_NumOfAlarms* of the internal variable, *Cfg_TargetEMAlarmStatus*; retains the index number of the unused element of the *Sts_Alarms* array in the internal variable; and increments *Sts_NumOfAlarms*.
- Afterwards, the FB writes the contents based on other input variables for the *sALARM* structure-type variable of the index number of internal input *Cfg_TargetEMAlarmStatus* member *Sts_Alarms* (*sALARM* structure array), and then sets *Privete_NeedToBeUpdated* for TRUE.
- The FB outputs the member corresponding to *sALARM* structure of its index number.

5.4. AlarmStatus_Update Function

5.4.1. Provided Function

This function checks whether each alarm status changed against *Cfg_EMAlarmStatus* that indicates the status of Alarms collected to EM as In/Out variables, and then updates each member of *sALARM_STATUS*.
Also, the FB resets *Cfg_EMAlarmStatus* based on the instruction given as In/Out variables.

■ Appearance of the function (call representation example in ladder language)



5.4.2. Input Variable/Output Variable/ In/Out Variable

Name	In/Out	Data Type	Default	Description
EN	Input	BOOL	TRUE	Execution Control Flag. At TRUE, the internal code in this function is executed. At FALSE, nothing executes.
Cfg_EMAlarmStatus	In/Out	OmronLib¥PackML¥ sALARM_STATUS	-	Update Alarm States. Specifies the Alarm status variable to be updated by this function.

Cmd_Reset	Input	BOOL	FALSE	Reset Command. At TRUE, all information except *FirstOutAlarm* of the target alarm status is reset.
Cmd_ClearFirstOutAlarms	Input	BOOL	FALSE	First Alarm Clear Command. At TRUE, *FirstOutAlarm* of the target Alarm status is cleared.
ENO	Output	BOOL	-	Execution Control Flag Output. EN is reflected as it is.

※The data of “sALARM_STATUS” that “Cmd_Reset” reset are as follows.

- `Sts_Alarms（EventType は除く）`
- `Sts_CategoryActiveFlag`
- `Sts_CategoryLatchedFlag`

5.4.3. Function details

When EN = TRUE, this function takes the following actions.

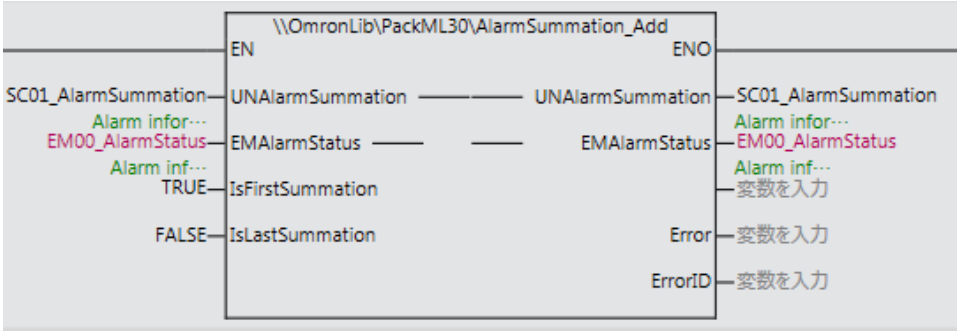
- When *Cmd_Reset* is TRUE, the function resets *"Active"* and *"Latched"* of each element of *Cfg_EMAlarmStatus.Sts_Alarms* to FALSE
- When *Cfg_EMAlarmStatus.Privete_NeedToBeUpdated* is TRUE, the function scans the *Cfg_EMAlarmStatus.Sts_Alarms* array, and updates each member underneath of it.

5.5. AlarmSummation_Add Function

5.5.1. Provided Function

This function adds the specific EM *Alarm* status given by In/Out variable *Sfg_EMEventStatus* for the In/Out variable that retains the *Alarm* statuses merged to *Cfg_UNEventSummation* UN (unit/machine).

■ Appearance of the function (call representation example in ladder language)



5.5.2. Input Variable/Output Variable/ In/Out Variable

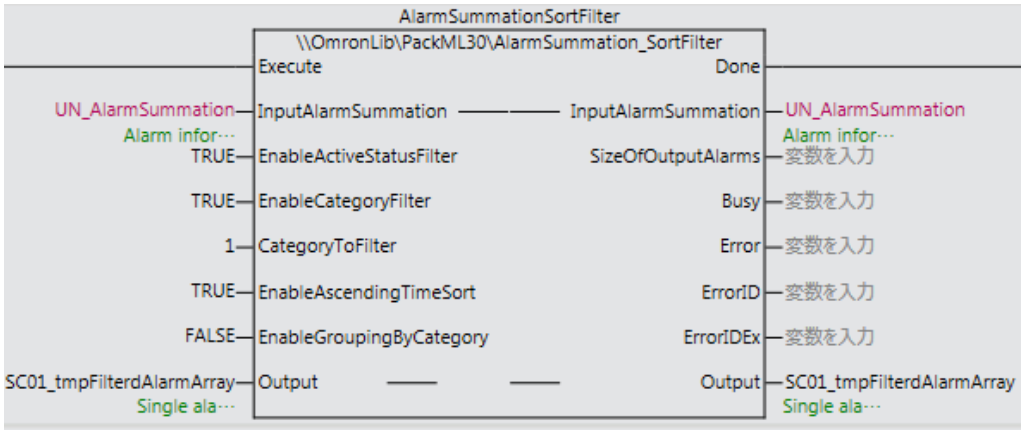
Name	In/Out	Data Type	Default	Description
EN	Input	BOOL	TRUE	Execution Control Flag. At TRUE, the internal code in this function is executed. At FALSE, nothing executes.
UNAlarmSummation	In/Out	OmronLib¥PackML¥ sALARM_SUMMMATION	-	Update Machine-level Alarm Status. Specifies the *sALARM_SUMMATION* variable to be updated by this function.
EMAlarmSutatus	In/Out	OmronLib¥PackML¥ sALARM_STATUS	-	Added EM Alarm Status. The EM-level alarm status that should be added to the machine-level alarm status.
IsFirstSummation	Input	BOOL	FALSE	Sets TRUE when the first EM-level status is added to the machine-level alarm status. At TRUE, *UNAlarmSummation* is cleared and then *EMAlarmStatus* is added on top. At FALSE, it is added to the tail of the existing valid array size.
IsLastSummation	Input	BOOL	TRUE	Sets TRUE when the last EM-level status is added to the machine-level alarm status. At TRUE, the necessary members for *UNAlarmSummation* are updated after *EMAlarmStatus* is added to *UNAlarmSummation*.
ENO	Output	BOOL	-	Execution Control Flag. EN is reflected as it is.

5.6. AlarmSummation_SortFilter Function Block

5.6.1. Provided Function

This function reflects the results of filtering and sorting that are conducted with the conditions specified by the In/Out variable *Cfg_UNEventSummation* that retains the Alarm statuses merged into UN (unit/machine), to *sALARM* array variable Output.

■ Appearance of the function (call representation example in ladder language)



5.6.2. Input Variable/Output Variable/ In/Out Variable

Name	In/Out	Data Type	Default	Description
EN	Input	BOOL	TRUE	Execution Control Flag. At TRUE, the internal code in this function is executed. At FALSE, nothing executes.
InputAlarmSummation	In/Out	OmronLib¥PackML¥ sALARM_SUMMATION	- .	Source Alarm Information. The machine-level alarm status structure variable that includes the alarm array source of filtering and sorting.
EnableActiveStatusFilter	Input	BOOL	FALSE	Active Alarm Filter Enabled Flag. At TRUE, only the *Alarms* whose ACTIVE = TRUE is output.
EnableCategoryFilter	Input	BOOL	FALSE	Category Filter Enabled Flag. At TRUE, only the categories that are specified by the following *CategoryFilter* are output.

CategoryToFilter	Input	USINT(0..9)	0	Category Number for Filter. The category number to be output by the category filter is specified.
EnableAscendingTimeSort	Input	BOOL	FALSE	Time Ascending Flag. At TRUE, sorted alarm occurrence time in ascending order is updated to Output.
EnableGroupingByCategory	Input	BOOL	FALSE	Categorized Groping Flag. At TRUE, the sorted group numbers in ascending order are updated to Output.
Output	In/Out	ARRAY[0..399] OF OmronLib¥PackML¥sALARM	-	The Alarm array to which the sorted/filtered results are output.
Done	Output	BOOL	-	This flag turns on after having executed the function.
Busy	Output	BOOL	-	This flag turns on during executing this function.
Error	Output	BOOL	-	It is always 0 (normal) because internal error never occurs in this FB.
ErrorID	Output	WORD	-	It is always 0 (normal) because internal error never occurs in this FB.
ErrorIDEx	Output	DWORD	-	It is always 0 (normal) because internal error never occurs in this FB.

5.6.3. Filter Function

When Execute turns on the FB execute as follows;

- If EnableActiveStatusFilter is true, Alarms that the “Active” is True are resistered "Output".

- If EnableCateoryFilter is True,Alarms that the category corresponse with the one in the “CategoryToFilter” are resistered “Output”
- If “EnableActiveStatusFilter” and “EnableCateoryFilter” are true,Alarms “Active” is true and the category corresponse with the one in the “CategoryToFilter” are resistered “Output”

5.6.4. Sort Function

When Execute turns on the FB execute as follows;

- If EnableAscendingTimeSort is True,Alarms are sorted occurrence time in ascending order is updated to Output.

- If EnableGroupingByCategory is True,Alarms are sorted group numbers in ascending order are updated to Output.

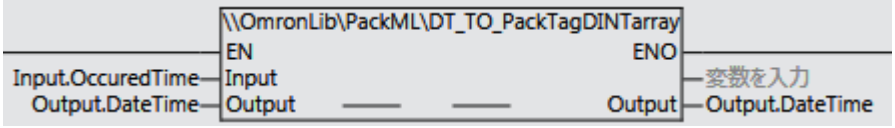
- If EnableAscendingTimeSort and EnableGroupingByCategory are true, first alarms are sorted group numbers in ascending order and then are sorted occurrence time in ascending order.

5.7. DT_TO_PackTagDINTarray Function

5.7.1. Provided Functions

This function converts the input of DATE_AND_TIME into the date-time array specified by PackTags.

■ Appearance of the function (call representation example in ladder language)



5.7.2. Input Variable/Output Variable/Input and Output Variable

Name	I/O	Data Type	Default	Description
EN	Input	BOOL	TRUE	Execution Control Flag. At TRUE, the internal code in this function is executed. At FALSE, nothing executes.
Input	Input	DATE_AND_ TIME	0	DATE_AND_TIME value to be converted.
Output	In/Out	ARRAY[0..6] OF DINT	-	Destination of conversion result in PackTag format - Array element 0 = Year - Array element 1 = Month - Array element 2 = Day - Array element 3 = Hour (24hr format) - Array element 4 = Min - Array element 5 = Sec - Array element 6 = USec (1/1,000,000 sec)

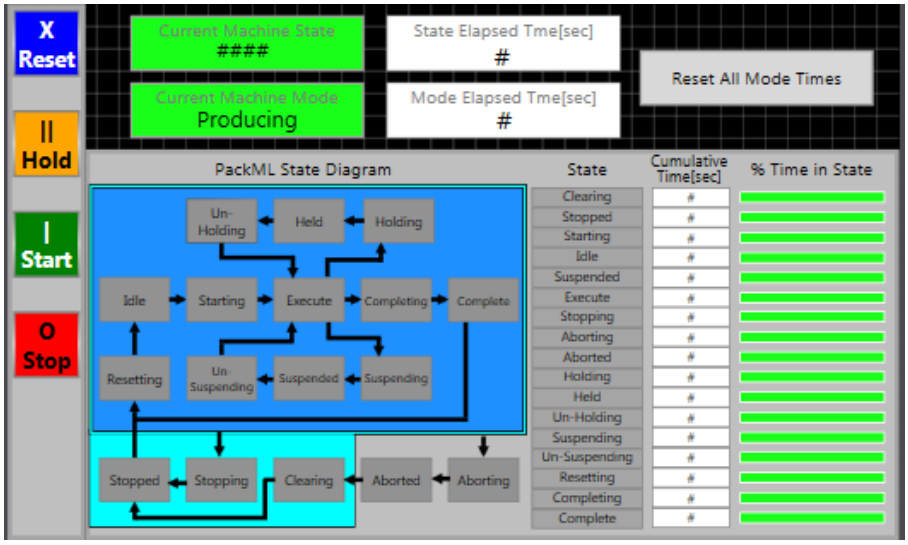
6. Sample Project

6.1. Sample HMI

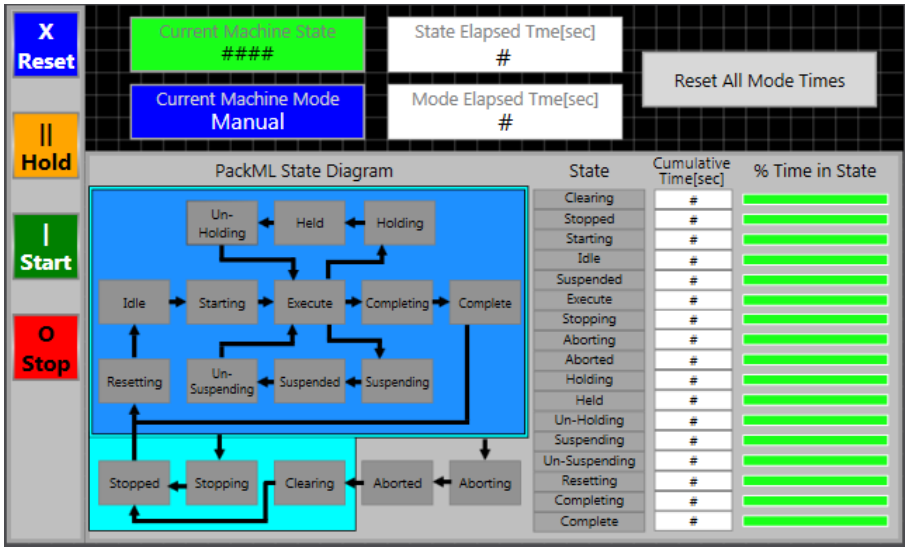
We supply the sample HMI of OMAC Machine Template PackML HMI Screen
for NA series Support Type: NA5 – 9W
There are three screens in this project for three modes.

- Producing mode
- Manual mode
- Maintenace mode

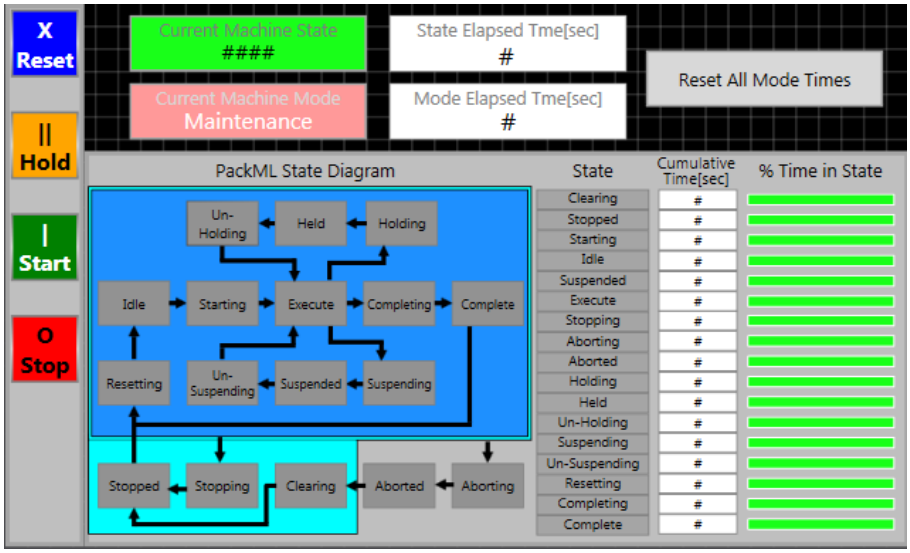
<Producing mode>



<Manual mode>

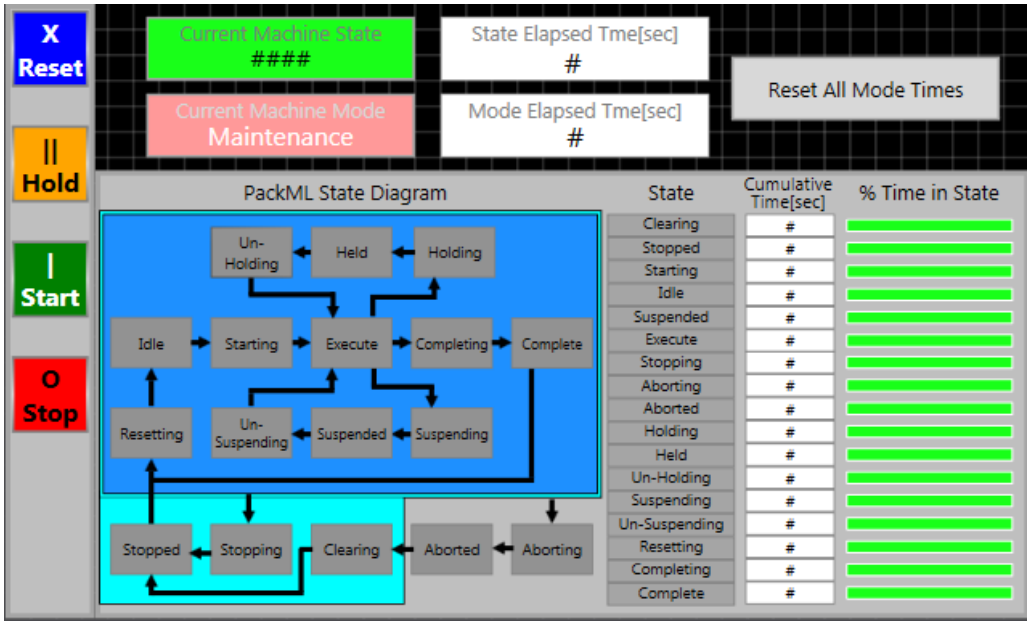


<Maintenance mode>



<Operations of screens>

Operations of all screens are same as follows.



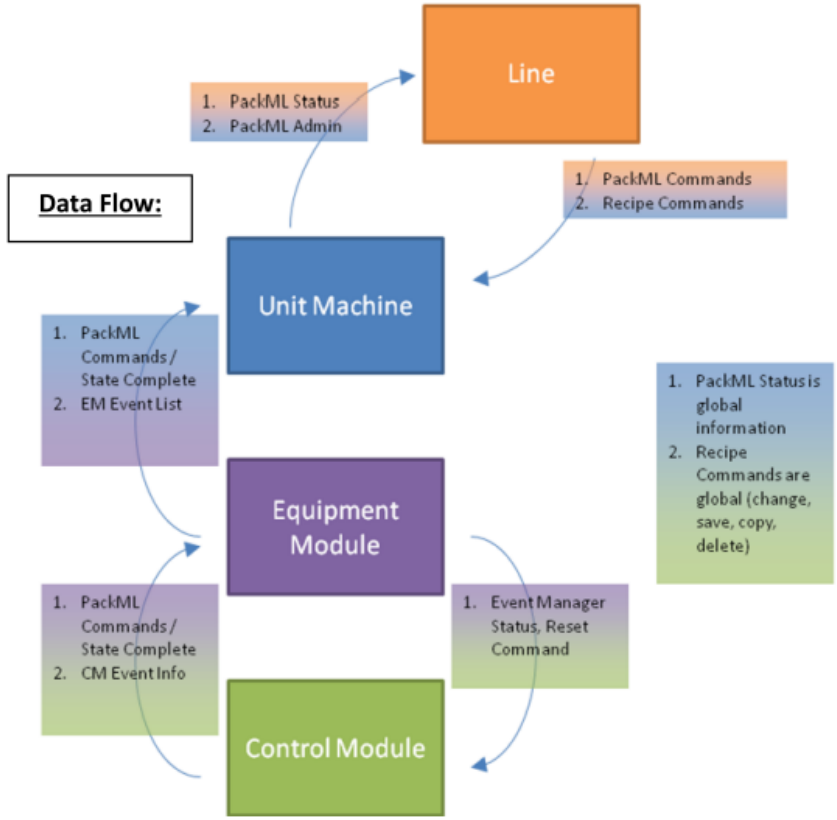
Parts name	Contents	
Button: Reset	When the button is pushed .. Current State: Activity Stopped: State is Transit to Resetting Other States: Nothing occurs.	
Button: Hold	When the button is pushed .. Current State: Activity Execute: State is Transit to Holding Other States: Nothing occurs.	
Button: Start	When the button is pushed .. Current State: Activity Idle: State is Transit to Starting Other States: Nothing occurs.	
Button: Stop	When the button is pushed .. Current State: Activity The States in the blue area: State is Transit to Stopping Other States: Nothing occurs.	
Monitor: Current Machine State	Indicate current State name	
Monitor: Current Machine Mode	Indicate current Mode name	
Monitor: State Elapsed Time	Indicate the elapsed time of current state	
Monitor: Mode Elapsed Time	Indicate the elapsed time of current mode	
Button Reset All mode time	When the button is pushed, the controller reset the time of all states in all mode.	
Monitor PackMLStateDiagram	Indicate Curent State is turned on	
Monitor:State	Indicate Curent State is turned on	

Monitor: Cumultive time	Indicate time of Cumulative time of each States of current Mode	
Monitor: % Time in State		

6.2. Sample Program

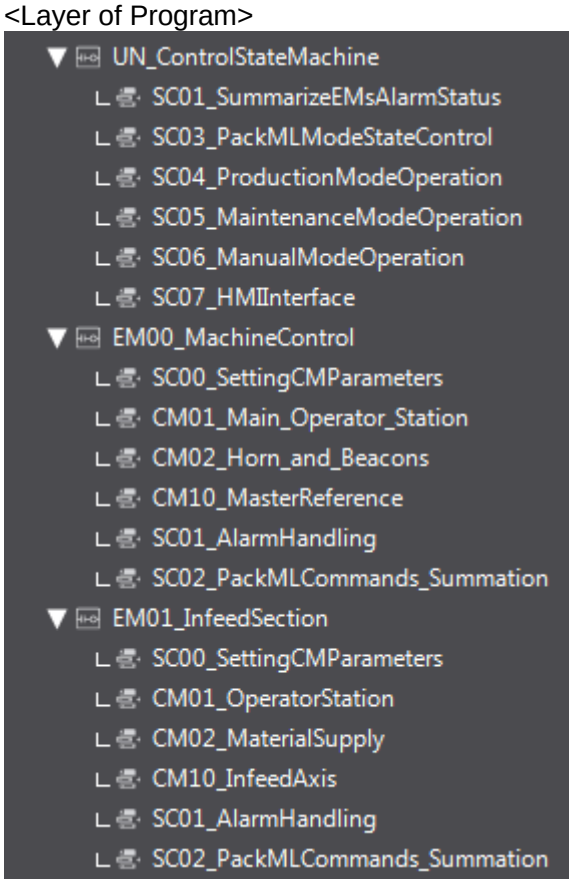
Refer to project “PackML_Template.smc2”
In this Project we define the module as follows.

The program has three layers for machine module.



UN	Unit Machine: Administers the state in the whole machine. Execute PackML ModeState machine Summarize Alarm status from all EMs Summarize Transition commands from all EMs Mesure the elapsed time of all Modes and all States Update PackTag for Communicating to upper system. Communicate HMI
EM	Equipment machine:mechanical module Summarizes Transition command from it’s own CMs. Makes the SC for each transition
CM	Control module: Reports information for each Alarm.

<Layer of Program>



OMRON AUTOMATION AMERICAS HEADQUARTERS • Chicago, IL USA • 847.843.7900 • 800.556.6766 • www.omron247.com

OMRON CANADA, INC. • HEAD OFFICE
Toronto, ON, Canada • 416.286.6465 • 866.986.6766 • www.omron247.com

OMRON ELECTRONICS DE MEXICO • HEAD OFFICE
México DF • 52.55.59.01.43.00 • 01-800-226-6766 • mela@omron.com

OMRON ELECTRONICS DE MEXICO • SALES OFFICE
Apodaca, N.L. • 52.81.11.56.99.20 • 01-800-226-6766 • mela@omron.com

OMRON ELETRÔNICA DO BRASIL LTDA • HEAD OFFICE
São Paulo, SP, Brasil • 55.11.2101.6300 • www.omron.com.br

OMRON ARGENTINA • SALES OFFICE
Cono Sur • 54.11.4783.5300

OMRON CHILE • SALES OFFICE
Santiago • 56.9.9917.3920

OTHER OMRON LATIN AMERICA SALES
54.11.4783.5300

OMRON EUROPE B.V. • Wegalaan 67-69, NL-2132 JD, Hoofddorp, The Netherlands. • +31 (0) 23 568 13 00 • www.industrial.omron.eu

Authorized Distributor:

Controllers & I/O
• Machine Automation Controllers (MAC) • Motion Controllers
• Programmable Logic Controllers (PLC) • Temperature Controllers • Remote I/O

Robotics
• Industrial Robots • Mobile Robots

Operator Interfaces
• Human Machine Interface (HMI)

Motion & Drives
• Machine Automation Controllers (MAC) • Motion Controllers • Servo Systems
• Frequency Inverters

Vision, Measurement & Identification
• Vision Sensors & Systems • Measurement Sensors • Auto Identification Systems

Sensing
• Photoelectric Sensors • Fiber-Optic Sensors • Proximity Sensors
• Rotary Encoders • Ultrasonic Sensors

Safety
• Safety Light Curtains • Safety Laser Scanners • Programmable Safety Systems
• Safety Mats and Edges • Safety Door Switches • Emergency Stop Devices
• Safety Switches & Operator Controls • Safety Monitoring/Force-guided Relays

Control Components
• Power Supplies • Timers • Counters • Programmable Relays
• Digital Panel Meters • Monitoring Products

Switches & Relays
• Limit Switches • Pushbutton Switches • Electromechanical Relays
• Solid State Relays

Software
• Programming & Configuration • Runtime

W22I-E-01 07/16 Note: Specifications are subject to change. Printed in U.S.A.

Printed on recycled paper.