SIEMENS

Ingenuity for life

OMAC PackML V3.0 for S7-1200 / S7-1500

LPMLV30 for SIMATIC

https://support.industry.siemens.com/cs/ww/en/view/49970441

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: http://www.siemens.com/industrialsecurity.

Table of contents

1 Library Overview

What you get

This document describes the LPMLV30 block library based on ISA TR88.00.02, June 2014. The block library provides you with the tested code with clearly defined interfaces. They can be used as a basis for your task to be implemented.

A key concern of the document is to describe

- all blocks of the block library
- the functionality implemented through these blocks.

Furthermore, this documentation shows possible fields of application and helps you integrate the library into your STEP 7 project using step-by-step instructions.

Scope of application

- STEP 7 Basic V14
- STEP 7 Professional V14
- S7-1200 CPU as of firmware 4.2
- S7-1500 CPU as of firmware 2.0

1.1 Different user scenarios

Possible application for the LPMLV30 library

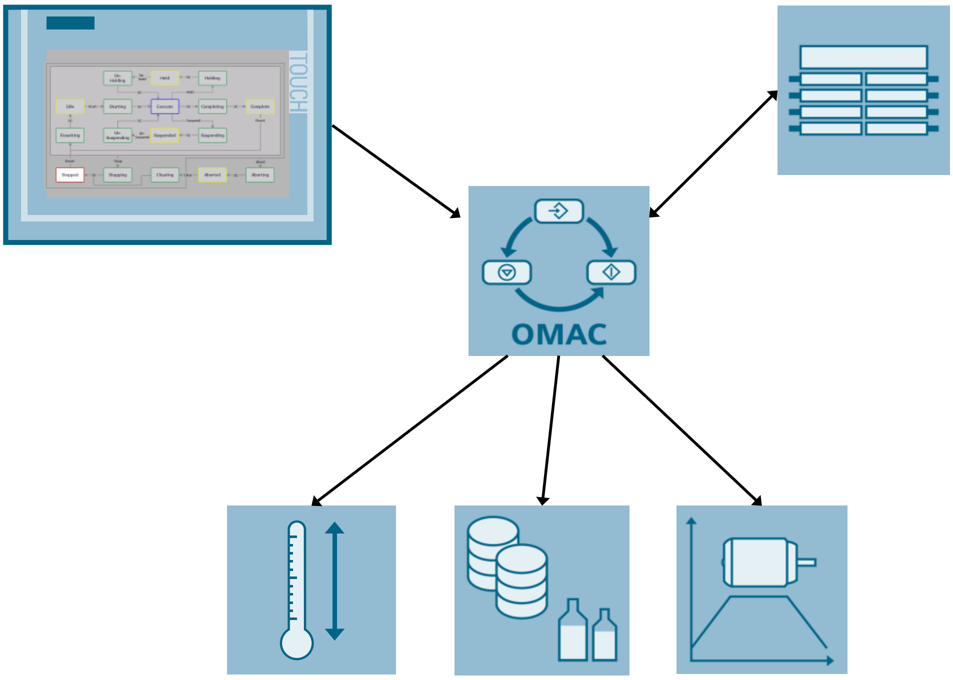
The following section shows a scenario for a possible application of the LPMLV30 library:

Scenario

A production machine consists of different machine modes (e.g. manual or production mode) and states (e.g. stopped or aborting). The machine state can be controlled with commands (e.g. start or stop).

The library LPMLV30 provides a standardized mode and state manager to control a machine. User defined code can be filled in an easy to use template structure. In addition a machine HMI or an upper level system (MES) can be connected by using the standardized machine interface.

Figure 1-1: User scenario



1.1.1 OMAC Packaging Working Group

Structure, contents and aim of the OMAC Packaging Working Group

OMAC (The Organization for Machine Automation and Control) is the organization for automation and manufacturing professionals that is dedicated to supporting the machine automation and operation needs of manufacturing.

The OMAC Packaging Working Group was formed as an initiative from big international end users. In the working group, end users, machine manufacturers (OEMs) and controller manufacturers discuss standards for the automation of production machines in order to reduce the range of variants for different products, technologies and applications.

The objective is to achieve significant improvements regarding the following points:

- Delivery time
- Commissioning time
- Machine dimensions
- Machine performance
- Integration capability
- Format change time
- Flexibility
- Machine modularity
- Machine downtime

Within the OMAC Packaging Working Group, the PackML working group is involved with the definition of guidelines and standards to achieve a standard automation software structure.

NOTE Knowledge about the contents of the basic OMAC documents is an advantage when it comes to understanding the solutions described in this documentation.

See also

OMAC website (http://www.omac.org)

1.1.2 OMAC PackML Guidelines

The major part of the guidelines describes the OMAC mode management (unit mode manager and state machine), see *ISA Technical Report TR88.00.02 Machine and Unit States*.

In addition the PackML pack tags are listed which are used as standardized variable structures (pack tags) for the cross-machine coupling between machine controllers and to higher-level HMI, MES or Enterprise systems, see also *ISA Technical Report TR88.00.02 Machine and Unit States*.

There exist the following variable structures for the pack tags:

- Command tags, to control and parameterize the machine
- Status tags, to provide information about the machine state
- Administration tags, to provide information about the machine efficiency (OEE data) and machine diagnostics

1.1.3 Unit Mode and State Manager

General information

The LPMLV30 library contains a function block for the unit mode and state management according to PackML V3.0.

- Unit modes *Manual*, *Maintenance*, *Production* and *user-defined modes*
- Uniform states within a unit mode
  Defined states, such as *Stopped*, *Starting*, *Execute*, *Aborting*, etc. can be used to handle the machine states within an operating mode. Users can individually remove states that are not used in compliance with the OMAC guidelines.

The machine functionality to be executed in the particular modes and states must be programmed by the user for the specific application.

Modes and states according to PackML V3.0

The *Production*, *Maintenance*, *Manual* and the *user-defined modes* with their associated states defined by PackML V3.0 are listed in this section. The state machines of the *Manual*, *Maintenance* and the *user-defined modes* are typically a subset of the state machine of the *Production* mode. Which states are used in the individual modes is not standardized and users can define them as required. The state model for the *Production* mode should be considered as the maximum quantity structure, which can be reduced, but should not be increased. This means that the state machine of the *Production* mode is always used and for smaller quantity structures, individual states are directly run-through or skipped.

Figure 1-2: Example of a state machine for the Production mode

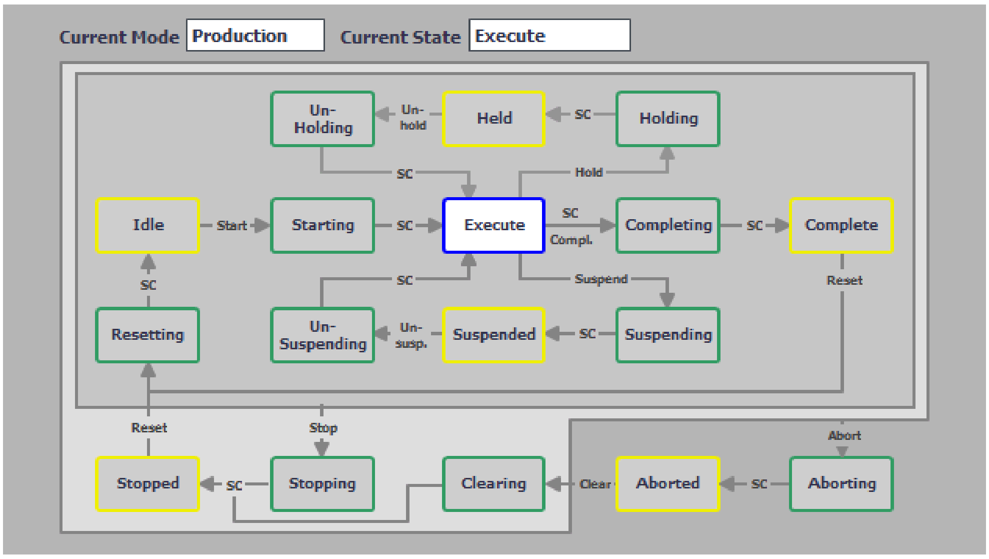
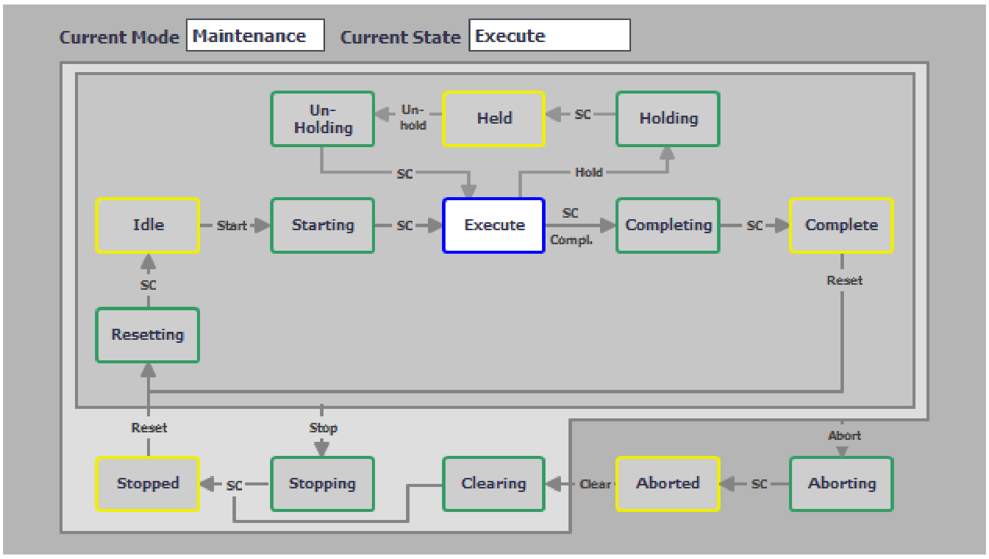


Figure 1-3: Example of a state machine for the Maintenance mode



Current Mode: Maintenance

Current State: Execute

Unit modes

Table 1-1: Description of the possible unit modes

Number	Unit Mode	Description
0	Invalid	Not a valid unit mode.
1	Production	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	This mode may allow suitably authorized personnel the ability to run an individual machine independent of other machines in a production line. This mode would typically be used for fault finding, 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	This provides direct control of individual machine modules. This feature is available depending upon the mechanical constraints of the mechanisms being exercised. This feature may be used for the commissioning of individual drives, verifying the operation of synchronized drives, testing the drive as a result of modifying parameters etc.
04..31	UserMode01…UserMode28	The requirements for user-defined unit modes differ depending on the machine and application. A typical user-defined unit mode is, for example, a cleaning mode.

States

Table 1-2: Description of the possible states

Number	State	Description
0	Undefined	Not a valid state.
1	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.
2	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.
3	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".
4	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*.
5	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.
6	Execute	**State Type: Acting** Once the machine is processing materials it 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.
7	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.
8	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.
9	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.
10	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

Number	State	Description
		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.
11	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.
12	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.
13	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.
14	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.
15	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.
16	Completing	**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.
17	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.

Control commands

Table 1-3: Possible control commands

Number	Control command
0	Undefined
1	Reset
2	Start
3	Stop
4	Hold
5	Unhold
6	Suspend
7	Unsuspend
8	Abort
9	Clear
10	Complete [1]

Unit mode transitions

Permitted change of the unit mode

Changing the unit mode is only permitted in w*ait* states (state type: Wait, e.g. *Stopped, Idle, Suspended, Aborted, Held and Complete*). The unit mode change is only possible if the wait state also exists in the requested unit mode.

State transitions

Example of reading the table

A change is made from the *Idle* state to the *Starting* state with the *Start* command. The further to the right that a command is located in the table, the higher its priority for the state change.

Additional info: if a unit mode configuration does not include *Completing*/*Complete* states then the transition from *Execute* to *Resetting* is possible with the *Reset* command.

[1] The *complete* command is not described in *ISA Technical Report TR88.00.02 Machine and Unit States*, but exists in this solution to be compatible to the “PackML V3 Demo in MS Excel” example on the OMAC website and also to the previous version of this library. According to the standard the SC signal should be used instead of the *complete* command.

State change with priority assignment

	State Commands								State Complete
Current State	Start	Reset[1]	Hold	Unhold	Suspend	Unsuspend	Clear[1]	Stop[2]	Abort[2]	
IDLE	STARTING							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

[1] It is common practice for Clearing and Resetting COMMANDS to be initiated using the same physical operator interface device.

[2] It is common practice in Packaging (but not Process) applications to permit use of STOP and ABORT commands while in the IDLE, COMPLETE, STOPPED, and RESETTING states.

1.2 Hardware and software requirements

Requirements for this library

To be able to use the functionality of the library described in this document, the following hardware and software requirements must be met:

Hardware

Table 1-4: Hardware components

No.	Component	Article number	Alternative
1.	CPU 1513-1 PN	6ES7513-1AL01-0AB0	Other S7-1500 CPU with FW V2.0
2.	Or CPU 1215C	6ES7215-1AG40-0XB0	Other S7-1200 CPU with FW V4.2

Software

Table 1-5: Software components

No.	Component	Article number	Quantity
1.	STEP 7 Professional V14	6ES7822-1..04-..	1
2.	Or STEP 7 Basic V14	6ES7822-0A.04-..	1

1.3 Library resources

What will you find in this section?

The following section gives you an overview of the size of the blocks of the LPMLV30 library in the main, load and retain memory.

Overall size

The overall size of the mandatory blocks (*UnitModeStateManager*) of the LPMLV30 library in the main memory is 11.8 Kbytes and 113.5 Kbytes in the load memory.

Size of the individual blocks [2]

Table 1-6: Size of the blocks

Block	Symbol	Size in main memory [Kbytes]	Size in load memory [Kbytes]	Size in retain memory [Kbytes]
FB 30100	LPMLV30_UnitModeStateManager	11	104	-
FB 30101	LPMLV30_UnitModeStateTimes	1.5	17.1	-
FC 30100	LPMLV30_ConfigureDisabledUnitModes	0.2	6	-
FC 30101	LPMLV30_ConfigureDisabledStates	0.2	5.7	-
FC 30102	LPMLV30_GetUnitModeStateNamesAsString	0.3	7	-
DB 30100	instLPMLV30_UnitModeStateManager	0.8	9.5	-
DB 30101	instLPMLV30_UnitModeStateTimes	2	4.8	0.9

[2] Instance data blocks (prefix *instLPMLV30_*) are not delivered with the library. They will be generated automatically with the call of a function block.

2 Blocks of the Library

What will you find in this section?

This chapter lists and explains all blocks of the LPMLV30 library. Before that, however, you are informed of the blocks that are essentially involved in the implementation of the functionality.

2.1 List of the blocks

The following table lists all blocks of the LPMLV30 library.

Table 2-1: List of the blocks

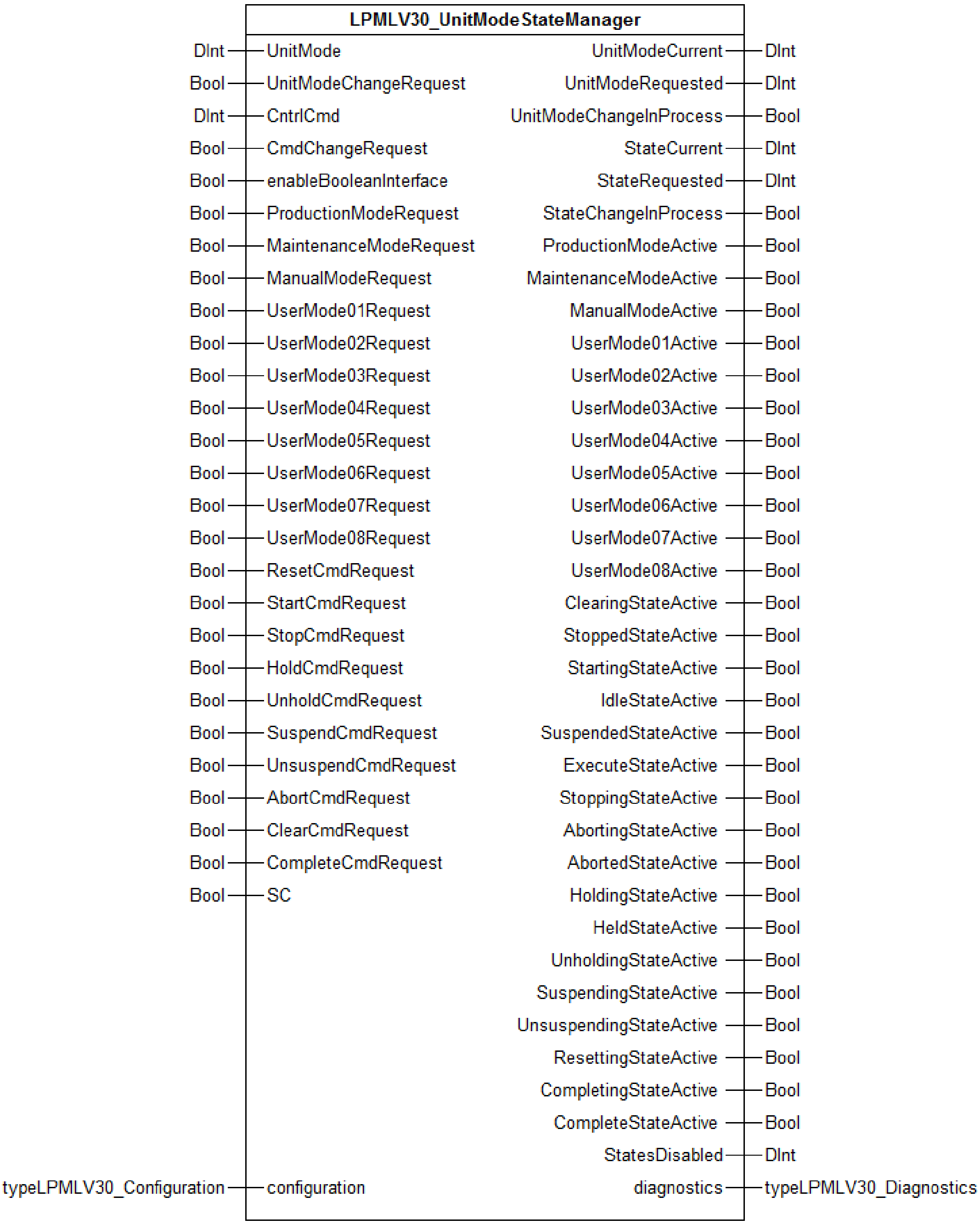
Block	Symbol	Classification
FB 30100	LPMLV30_UnitModeStateManager	In-house development
FB 30101	LPMLV30_UnitModeStateTimes	In-house development
FC 30100	LPMLV30_ConfigureDisabledUnitModes	In-house development
FC 30101	LPMLV30_ConfigureDisabledStates	In-house development
FC 30102	LPMLV30_GetUnitModeStateNamesAsString	In-house development

2.2 Explanation of the blocks

The following table explains all blocks of the LPMLV30 library.

2.2.1 FB LPMLV30_UnitModeStateManager (FB 30100)

Figure



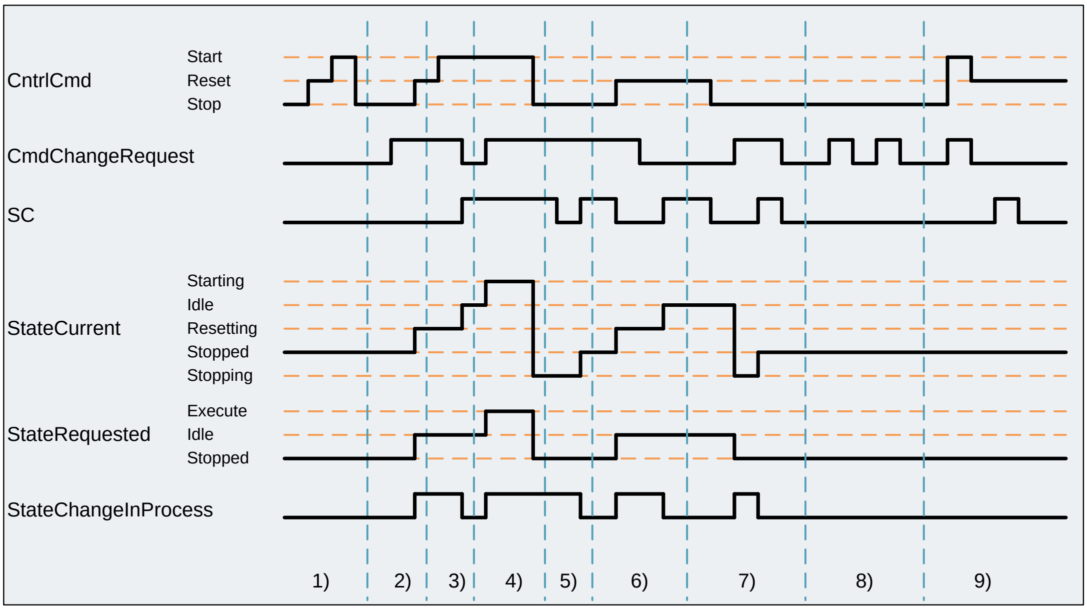
Principle of operation

The function block *LPMLV30_UnitModeStateManager* is the main part of the block library LPMLV30 and manages the transitions between the unit modes and states according to the OMAC standard.

NOTE If you want to use the boolean interface particularly in ladder, set input *enableBooleanInterface := TRUE*.

NOTE The number of unit modes can be changed by changing the constant *LPMLV30_MODES_UPPER_LIM*. In this case the number of the inputs has to be changed respectively.

Function characteristics



1. If *CmdChangeRequest* is not set, every C*ntrlCmd* is ignored.
2. If C*mdChangeRequest* is set to TRUE and a valid C*ntrlCmd* (in this case Reset) is set, the *StateChangeInProcess* bit is set and the *StateRequested* (in this case Idle) value is set (only wait states possible). A valid C*ntrlCmd* is always necessary if the current state is a wait state.

   If the current state is an acting state (here Resetting), a rising edge at input *SC* is necessary to leave the state.
   If C*mdChangeRequest* is FALSE, the C*ntrlCmd* (here Start) has already been set in the acting state (here Resetting) and the wait state (Idle) is reached through a rising edge at the input *SC*, the next acting state will not be reached automatically. For the change in the next wait state (here Execute) is a rising edge at C*mdChangeRequest* needed.
3. The *StateChangeInProcess* bit remains set until *StateCurrent* gets the same value as *StateRequested*. The *StateRequested* changes from Execute to Stopped, because in Starting state a valid Stop command was set.
4. The *SC* input is not level sensitive. If the *SC* input is already set when reaching the acting state, the *Unit Mode and State Manager* stays in the acting state as long as a rising edge at *SC* input is detected.
5. The SC command is not related to the C*mdChangeRequest* input. Even if C*mdChangeRequest* is FALSE, a state change can happen from acting to a wait state.
6. A new CntrlCmd can also be set if C*mdChangeRequest* is FALSE. If C*mdChangeRequest* changes to TRUE and the control command is valid in this wait state, the state will be changed.
7. If no valid control command for the current state is written to input, edges at *CmdChangeRequest* are ignored. To change from a wait to an acting state, *CmdChangeRequest* have to be TRUE and the according *CntrlCmd* have to be written to input *CntrlCmd*.
8. If no acting state is active the rising edge at the *SC* input is ignored.
9. If an invalid *CntrlCmd* is written to input *CntrlCmd* and *CmdChangeRequest* is set to TRUE, the control command is ignored and an entry is written to the diagnostics buffer of the FB.

Input parameters

Table 2-2: *LPMLV30_UnitModeStateManager* input parameters

Parameter	Data type	Description
UnitMode	DInt	Requested unit mode if enableBooleanInterface = FALSE (default: LPMLV30_MODE_INVALID) For valid unit modes see Table 1-1: Description of the possible unit modes.
UnitModeChange Request	Bool	TRUE: Request unit mode if enableBooleanInterface = FALSE (default: FALSE)
CntrlCmd	DInt	Request control command if enableBooleanInterface = FALSE (default: LPMLV30_CMD_UNDEFINED) For valid control commands see Table 1-3: Possible control commands
CmdChangeRequest	Bool	TRUE: Enable change into requested state if enableBooleanInterface = FALSE (default: FALSE)
enableBoolean Interface	Bool	TRUE: Enable boolean interface (default: FALSE)
ProductionMode Request	Bool	TRUE: Request change to unit mode *Production* if enableBooleanInterface = TRUE (default: FALSE)
MaintenanceMode Request	Bool	TRUE: Request change to unit mode *Maintenance* if enableBooleanInterface = TRUE (default: FALSE)
ManualModeRequest	Bool	TRUE: Request change to unit mode *Manual* if enableBooleanInterface = TRUE (default: FALSE)
UserMode01Request	Bool	TRUE: Request change to user-defined unit mode 01 if enableBooleanInterface = TRUE (default: FALSE)
UserMode02Request	Bool	TRUE: Request change to user-defined unit mode 02 if enableBooleanInterface = TRUE (default: FALSE)
UserMode03Request	Bool	TRUE: Request change to user-defined unit mode 03 if enableBooleanInterface = TRUE (default: FALSE)
UserMode04Request	Bool	TRUE: Request change to user-defined unit mode 04 if enableBooleanInterface = TRUE (default: FALSE)
UserMode05Request	Bool	TRUE: Request change to user-defined unit mode 05 if enableBooleanInterface = TRUE (default: FALSE)
UserMode06Request	Bool	TRUE: Request change to user-defined unit mode 06 if enableBooleanInterface = TRUE (default: FALSE)
UserMode07Request	Bool	TRUE: Request change to user-defined unit mode 07 if enableBooleanInterface = TRUE (default: FALSE)

Parameter	Data type	Description
UserMode08Request	Bool	TRUE: Request change to user-defined unit mode 08 if enableBooleanInterface = TRUE (default: FALSE)
ResetCmdRequest	Bool	TRUE: Request control command *Reset* if enableBooleanInterface = TRUE (default: FALSE)
StartCmdRequest	Bool	TRUE: Request control command *Start* if enableBooleanInterface = TRUE (default: FALSE)
StopCmdRequest	Bool	TRUE: Request control command *Stop* if enableBooleanInterface = TRUE (default: FALSE)
HoldCmdRequest	Bool	TRUE: Request control command *Hold* if enableBooleanInterface = TRUE (default: FALSE)
UnholdCmdRequest	Bool	TRUE: Request control command *Unhold* if enableBooleanInterface = TRUE (default: FALSE)
SuspendCmdRequest	Bool	TRUE: Request control command *Suspend* if enableBooleanInterface = TRUE (default: FALSE)
UnsuspendCmd Request	Bool	TRUE: Request control command *Unsuspend* if enableBooleanInterface = TRUE (default: FALSE)
AbortCmdRequest	Bool	TRUE: Request control command *Abort* if enableBooleanInterface = TRUE (default: FALSE)
ClearCmdRequest	Bool	TRUE: Request control command *Clear* if enableBooleanInterface = TRUE (default: FALSE)
CompleteCmdRequest	Bool	TRUE: Request control command *Complete* if enableBooleanInterface = TRUE (default: FALSE)
SC	Bool	State change from FALSE to TRUE (rising edge) triggers state complete signal (default: FALSE)
configuration	typeLPMLV30_ Configuration	FB configuration

Output parameters

Table 2-3: *LPMLV30_UnitModeStateManager* output parameters

Parameter	Data type	Description
UnitModeCurrent	DInt	Current unit mode For valid unit modes see Table 1-1: Description of the possible unit modes
UnitModeRequested	DInt	Requested unit mode
UnitModeChange InProcess	Bool	Unit mode change in process
StateCurrent	DInt	Current state For valid states see Table 1-2: Description of the possible states
StateRequested	DInt	Requested state
StateChangeIn Process	Bool	State change in process
ProductionModeActive	Bool	TRUE: Unit mode Production is currently active
MaintenanceMode Active	Bool	TRUE: Unit mode Maintenance is currently active
ManualModeActive	Bool	TRUE: Unit mode Manual is currently active
UserMode01Active	Bool	TRUE: User-defined unit mode 01 is currently active
UserMode02Active	Bool	TRUE: User-defined unit mode 02 is currently active
UserMode03Active	Bool	TRUE: User-defined unit mode 03 is currently active
UserMode04Active	Bool	TRUE: User-defined unit mode 04 is currently active
UserMode05Active	Bool	TRUE: User-defined unit mode 05 is currently active
UserMode06Active	Bool	TRUE: User-defined unit mode 06 is currently active
UserMode07Active	Bool	TRUE: User-defined unit mode 07 is currently active
UserMode08Active	Bool	TRUE: User-defined unit mode 08 is currently active
ClearingStateActive	Bool	TRUE: State Clearing is currently active
StoppedStateActive	Bool	TRUE: State Stopped is currently active
StartingStateActive	Bool	TRUE: State Starting is currently active
IdleStateActive	Bool	TRUE: State Idle is currently active
SuspendedStateActive	Bool	TRUE: State Suspended is currently active
ExecuteStateActive	Bool	TRUE: State Execute is currently active
StoppingStateActive	Bool	TRUE: State Stopping is currently active
AbortingStateActive	Bool	TRUE: State Aborting is currently active
AbortedStateActive	Bool	TRUE: State Aborted is currently active
HoldingStateActive	Bool	TRUE: State Holding is currently active
HeldStateActive	Bool	TRUE: State Held is currently active
UnholdingStateActive	Bool	TRUE: State Unholding is currently active
SuspendingState Active	Bool	TRUE: State Suspending is currently active
UnsuspendingState Active	Bool	TRUE: State Unsuspending is currently active
ResettingStateActive	Bool	TRUE: State Resetting is currently active
CompletingStateActive	Bool	TRUE: State Completing is currently active
CompleteStateActive	Bool	TRUE: State Complete is currently active

Parameter	Data type	Description
StatesDisabled	DInt	Disabled states in current unit mode
diagnostics	typeLPMLV30_ Diagnostics	Diagnostics information of FB

typeLPMLV30_Configuration

Table 2-4: *typeLPMLV30_Configuration*

Parameter	Data type	Description
disabledUnitModes	Array[0..31] of Bool	TRUE: Disable unit mode xx [0..LPMLV30_MAX_MODES_UPPER_LIM]
disabledStatesInUnit Modes	Array[0..31] of DInt	TRUE: Disable states in unit mode (bit number = state number) [0..LPMLV30_MAX_MODES_UPPER_LIM]
PackML_Version	Int	0: ANSI/ISA-TR88.00.02-2015 PackMLV30; 1: PackML Companion Specification 1.00.01 (default: 0)

NOTE

If parameter *PackML_Version* is set to 1 (PackML Companion Specification 1.00.01), the *Hold* command is also taken into account in the following states: *Starting*, *Suspending*, *Suspended*, *Unsuspending* and *Unholding*.

Therefore, a state transition from these states to *Held* via *Holding* is then possible.

typeLPMLV30_Diagnostics

Table 2-5: *typeLPMLV30_Diagnostics*

Parameter	Data type	Description
bufferIndex	Int	Index of actual buffer entry
buffer	Array[0..15] of typeLPMLV30_ DiagnosticsEntr y	Diagnostics information buffer [0..LPMLV30_DIAG_BUFFER_UPPER_LIM]

typeLPMLV30_DiagnosticsEntry

Table 2-6: *typeLPMLV30_DiagnosticsEntry*

Parameter	Data type	Description
timestamp	DTL	Timestamp for this entry
UnitModeCurrent	Byte	Current unit mode
StateCurrent	Byte	Current state
UnitMode	Byte	Requested unit mode
CntrlCmd	Byte	Request control command
SC	Bool	State complete signal
message	Byte	Message for this entry

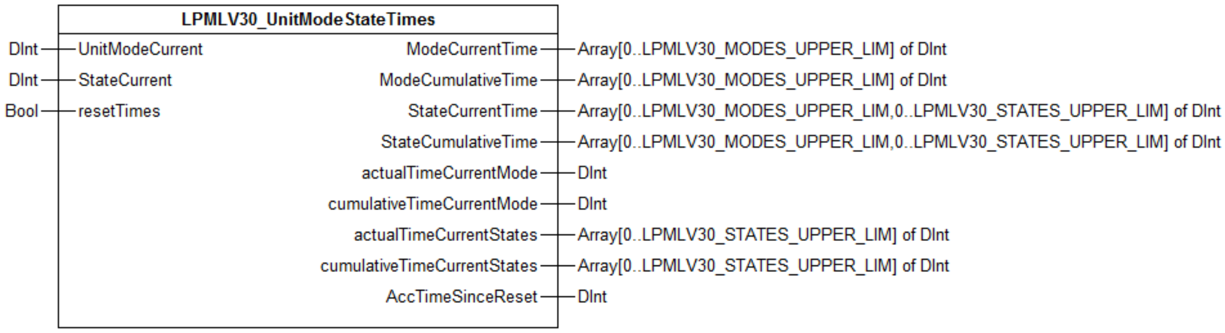
Status and error displays

Table 2-7

Status	Meaning	Remedy / notes
16#00	MSG_NO_MESSAGE	Initial value
16#01	MSG_MODE_CHANGED_SUCCESSFULLY	Unit mode changed successfully
16#02	MSG_STATE_CHANGED_SUCCESSFULLY	State changed successfully
16#03	MSG_MODE_ALREADY_ACTIVE	Requested unit mode already active
16#80	MSG_MODE_NOT_DEFINED	Unit mode not defined
16#81	MSG_CMD_NOT_DEFINED	Control command not defined
16#82	MSG_REQ_MODE_NOT_CONFIGURED	Requested unit mode not configured
16#83	MSG_MODE_TRANSITION_NOT_ALLOWED	Unit mode transition not allowed
16#84	MSG_CMD_NOT_ALLOWED	Control command in this state not allowed
16#85	MSG_SC_NOT_ALLOWED	SC in this state not allowed
16#86	MSG_STATE_CONFIG_FORCED	State configuration forced to OMAC standard (corrected configuration see FB output *StatesDisabled*)

2.2.2 FB LPMLV30_UnitModeStateTimes (FB 30101)

Figure


LPMLV30_UnitModeStateTimes
DInt
UnitModeCurrent
DInt
StateCurrent
Bool
resetTimes
ModeCurrentTime
Array[0..LPMLV30_MODES_UPPER_LIM] of DInt
ModeCumulativeTime
Array[0..LPMLV30_MODES_UPPER_LIM] of DInt
StateCurrentTime
Array[0..LPMLV30_MODES_UPPER_LIM,0..LPMLV30_STATES_UPPER_LIM] of DInt
StateCumulativeTime
Array[0..LPMLV30_MODES_UPPER_LIM,0..LPMLV30_STATES_UPPER_LIM] of DInt
actualTimeCurrentMode
DInt
cumulativeTimeCurrentMode
DInt
actualTimeCurrentStates
Array[0..LPMLV30_STATES_UPPER_LIM] of DInt
cumulativeTimeCurrentStates
Array[0..LPMLV30_STATES_UPPER_LIM] of DInt
AccTimeSinceReset
DInt

Principle of operation

The function block *LPMLV30_UnitModeStateTimes* is optional and counts the time in seconds for every state in every unit mode.

NOTE The number of unit modes can be changed by changing the constant *LPMLV30_MODES_UPPER_LIM*. In this case the number of the inputs has to be changed respectively.

Input parameters

Table 2-8: *LPMLV30_GetUnitModeStateNamesAsString* input parameters

Parameter	Data type	Description
UnitModeCurrent	DInt	Current unit mode
StateCurrent	DInt	Current state
language	Int	Requested language (LPMLV30_LANGUAGE_1, LPMLV30_LANGUAGE_2, ...)
namesConfiguration	typeLPMLV30_NamesConfiguration	All names of unit modes and states (in different languages)

Output parameters

Table 2-9: *LPMLV30_GetUnitModeStateNamesAsString* output parameters

Parameter	Data type	Description
unitModeCurrentName	String	Name of current unit mode in the requested language
stateCurrentName	String	Name of current state in the requested language

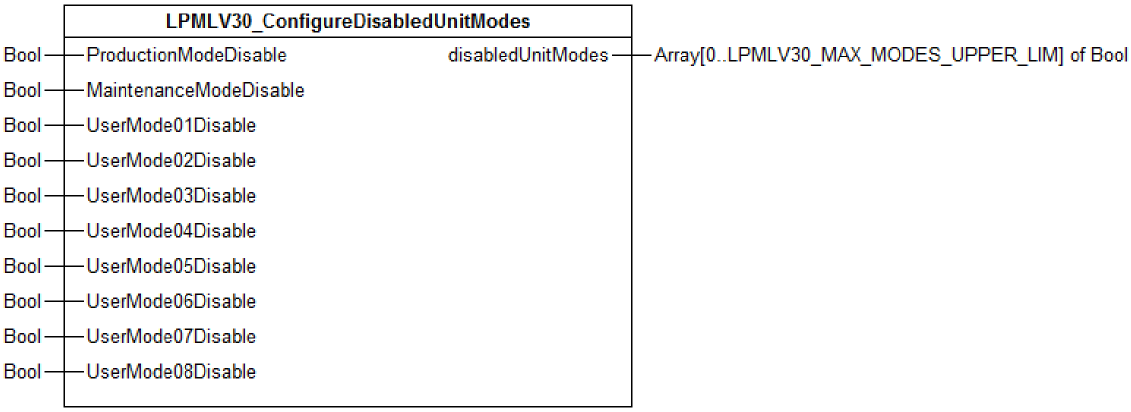
typeLPMLV30_NamesConfiguration

Table 2-10: *typeLPMLV30_NamesConfiguration*

Parameter	Data type	Description
unitModesNames	Array[0..1, 0..11] of String[18]	Names of unit modes [First index language 0..LPMLV30_LANGUAGES_UPPER_LIM, second index unit mode number 0..LPMLV30_MODES_UPPER_LIM]
statesNames	Array[0..1, 0..17] of String[16]	Names of states [First index language 0..LPMLV30_LANGUAGES_UPPER_LIM, second index state number 0..LPMLV30_STATES_UPPER_LIM]

2.2.3 FC LPMLV30_ConfigureDisabledUnitModes (FC 30100)

Figure



Principle of operation

This function allows the user to set the unit mode configuration for the FB *LPMLV30_UnitModeStateManager* easily. Of course it is also possible to set the unit mode configuration directly in the FB *LPMLV30_UnitModeStateManager* configuration.

With the function the user has to set the associated inputs for the different unit modes to “TRUE”, e.g. *“MaintenanceModeDisable := TRUE*” for disabling the unit mode *Maintenance*.

To write the unit mode configuration from the function output to the according *“Unit Mode and State Manager”,* the output *disabledUnitModes* has to be connected to the configuration of the corresponding FB *LPMLV30_UnitModeStateManager*.

NOTE

The unit modes *Invalid* and *Manual* are mandatory and cannot be disabled. If they are nevertheless disabled the “*Unit Mode and State Manager”* will enable these unit modes automatically.

NOTE

The number of unit modes can be changed by changing the constant *LPMLV30_MODES_UPPER_LIM*. In this case the number of the inputs has to be changed respectively.

Input parameters

Table 2-11: *LPMLV30_ConfigureDisabledUnitModes* input parameters

Parameter	Data type	Description
ProductionMode Disable	Bool	TRUE: Disable unit mode Production
MaintenanceMode Disable	Bool	TRUE: Disable unit mode Maintenance
UserMode01Disable	Bool	TRUE: Disable user-defined unit mode 01
UserMode02Disable	Bool	TRUE: Disable user-defined unit mode 02
UserMode03Disable	Bool	TRUE: Disable user-defined unit mode 03
UserMode04Disable	Bool	TRUE: Disable user-defined unit mode 04
UserMode05Disable	Bool	TRUE: Disable user-defined unit mode 05
UserMode06Disable	Bool	TRUE: Disable user-defined unit mode 06
UserMode07Disable	Bool	TRUE: Disable user-defined unit mode 07
UserMode08Disable	Bool	TRUE: Disable user-defined unit mode 08

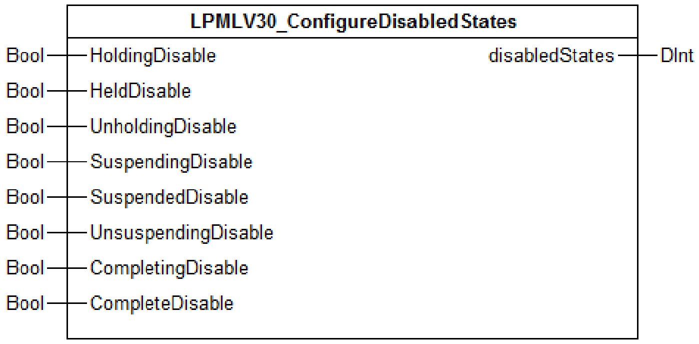
Output parameters

Table 2-12: *LPMLV30_ConfigureDisabledUnitModes* output parameters

Parameter	Data type	Description
disabledUnitModes	Array[0..LPMLV30_MAX_MODES_UPPER_LIM] of Bool	Disabled unit modes for direct connection with the input *configuration.disabledUnitModes* of the desired instance of the *LPMLV30_UnitModeStateManager* FB

2.2.4 FC LPMLV30_ConfigureDisabledStates (FC 30101)

Figure



Principle of operation

This function allows the user to set the state configuration for every unit mode in *LPMLV30_UnitModeStateManager* easily. Of course it is also possible to set the state configurations directly in the FB *LPMLV30_UnitModeStateManager* configuration.

With the function the user has to set the associated inputs for the different states to “TRUE”, e.g. “*HeldDisable := TRUE*” for disabling the state *Held*.

The function generates a double integer value which represents the state configuration for one unit mode. This value is bit coded and means that every bit represents a switch where states can be dis- or enabled for a unit mode, e.g. disabling the state *Held* the bit number 11 has to be set to “TRUE”. As can be seen in the example the state numbers according to the OMAC standard also define the bit numbers in the double integer value. (see Table 1-2: Description of the possible states)

To write the state configuration from the function output to the according *“Unit Mode and State Manager”,* the output *disabledStates* has to be connected to the configuration of the corresponding FB *LPMLV30_UnitModeStateManager*.

NOTE

According to the OMAC standard some states are mandatory and cannot be disabled. If they are nevertheless disabled the “*Unit Mode and State Manager”* will enable these states automatically and provides the corrected configuration as a double integer value at output *StatesDisabled*.

Input parameters

Table 2-13: *LPMLV30_ConfigureDisabledStates* input parameters

Parameter	Data type	Description
HoldingDisable	Bool	TRUE: Disable state Holding
HeldDisable	Bool	TRUE: Disable state Held
UnholdingDisable	Bool	TRUE: Disable state Unholding
SuspendingDisable	Bool	TRUE: Disable state Suspending
SuspendedDisable	Bool	TRUE: Disable state Suspended
UnsuspendingDisable	Bool	TRUE: Disable state Unsuspending
CompletingDisable	Bool	TRUE: Disable state Completing
CompleteDisable	Bool	TRUE: Disable state Complete

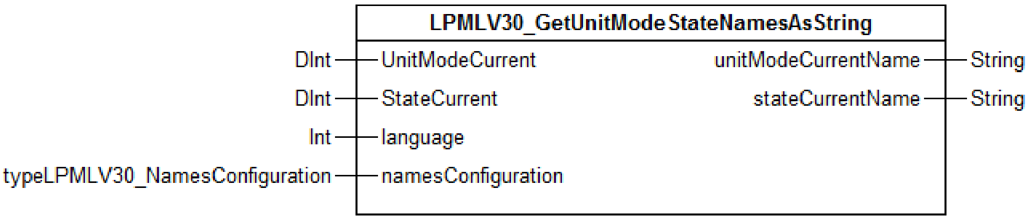
Output parameters

Table 2-14: *LPMLV30_ConfigureDisabledStates* output parameters

Parameter	Data type	Description
disabledStates	DInt	Disabled states for direct connection with the input *configuration.disabledStatesInUnitModes* of the desired instance of the FB *LPMLV30_UnitModeStateManager*

2.2.5 FC LPMLV30_GetUnitModeStateNamesAsString (FC 30102)

Figure



Principle of operation

The function *LPMLV30_GetUnitModeStateNamesAsString* is optional and provides the unit mode and state names as strings. The default names can be edited in the PLC data type *typeLPMLV30_NamesConfiguration*.

NOTE The number of languages can be changed by changing the constant *LPMLV30_LANGUAGES_UPPER_LIM*.

Input parameters

Table 2-15: *LPMLV30_GetUnitModeStateNamesAsString* input parameters

Parameter	Data type	Description
UnitModeCurrent	DInt	Current unit mode
StateCurrent	DInt	Current state
language	Int	Requested language (LPMLV30_LANGUAGE_1, LPMLV30_LANGUAGE_2, ...)
namesConfiguration	typeLPMLV30_NamesConfiguration	All names of unit modes and states (in different languages)

Output parameters

Table 2-16: *LPMLV30_GetUnitModeStateNamesAsString* output parameters

Parameter	Data type	Description
unitModeCurrentName	String	Name of current unit mode in the requested language
stateCurrentName	String	Name of current state in the requested language

typeLPMLV30_NamesConfiguration

Table 2-17: *typeLPMLV30_NamesConfiguration*

Parameter	Data type	Description
unitModesNames	Array[0..1, 0..11] of String[18]	Names of unit modes [First index language 0..LPMLV30_LANGUAGES_UPPER_LIM, second index unit mode number 0..LPMLV30_MODES_UPPER_LIM]
statesNames	Array[0..1, 0..17] of String[16]	Names of states [First index language 0..LPMLV30_LANGUAGES_UPPER_LIM, second index state number 0..LPMLV30_STATES_UPPER_LIM]

2.2.6 LPMLV30_Constants

The PLC tag table *LPMLV30_Constants* contains user constants for unit modes, states, control commands and array boundaries.

Unit Modes

Table 2-18: Constants for unit mode

Name	Data type	Value	Comment
LPMLV30_MODE_INVALID	DInt	0	OMAC PackMLV30 unit mode Invalid
LPMLV30_MODE_PRODUCTION	DInt	1	OMAC PackMLV30 unit mode Production
LPMLV30_MODE_MAINTENANCE	DInt	2	OMAC PackMLV30 unit mode Maintenance
LPMLV30_MODE_MANUAL	DInt	3	OMAC PackMLV30 unit mode Manual
LPMLV30_MODE_USER_01	DInt	4	OMAC PackMLV30 user-defined unit mode 01
LPMLV30_MODE_USER_02	DInt	5	OMAC PackMLV30 user-defined unit mode 02
LPMLV30_MODE_USER_03	DInt	6	OMAC PackMLV30 user-defined unit mode 03
LPMLV30_MODE_USER_04	DInt	7	OMAC PackMLV30 user-defined unit mode 04
LPMLV30_MODE_USER_05	DInt	8	OMAC PackMLV30 user-defined unit mode 05
LPMLV30_MODE_USER_06	DInt	9	OMAC PackMLV30 user-defined unit mode 06
LPMLV30_MODE_USER_07	DInt	10	OMAC PackMLV30 user-defined unit mode 07
LPMLV30_MODE_USER_08	DInt	11	OMAC PackMLV30 user-defined unit mode 08
LPMLV30_MODE_USER_09	DInt	12	OMAC PackMLV30 user-defined unit mode 09
LPMLV30_MODE_USER_10	DInt	13	OMAC PackMLV30 user-defined unit mode 10
LPMLV30_MODE_USER_11	DInt	14	OMAC PackMLV30 user-defined unit mode 11
LPMLV30_MODE_USER_12	DInt	15	OMAC PackMLV30 user-defined unit mode 12
LPMLV30_MODE_USER_13	DInt	16	OMAC PackMLV30 user-defined unit mode 13
LPMLV30_MODE_USER_14	DInt	17	OMAC PackMLV30 user-defined unit mode 14
LPMLV30_MODE_USER_15	DInt	18	OMAC PackMLV30 user-defined unit mode 15
LPMLV30_MODE_USER_16	DInt	19	OMAC PackMLV30 user-defined unit mode 16
LPMLV30_MODE_USER_17	DInt	20	OMAC PackMLV30 user-defined unit mode 17
LPMLV30_MODE_USER_18	DInt	21	OMAC PackMLV30 user-defined unit mode 18
LPMLV30_MODE_USER_19	DInt	22	OMAC PackMLV30 user-defined unit mode 19
LPMLV30_MODE_USER_20	DInt	23	OMAC PackMLV30 user-defined unit mode 20
LPMLV30_MODE_USER_21	DInt	24	OMAC PackMLV30 user-defined unit mode 21
LPMLV30_MODE_USER_22	DInt	25	OMAC PackMLV30 user-defined unit mode 22
LPMLV30_MODE_USER_23	DInt	26	OMAC PackMLV30 user-defined unit mode 23
LPMLV30_MODE_USER_24	DInt	27	OMAC PackMLV30 user-defined unit mode 24
LPMLV30_MODE_USER_25	DInt	28	OMAC PackMLV30 user-defined unit mode 25
LPMLV30_MODE_USER_26	DInt	29	OMAC PackMLV30 user-defined unit mode 26
LPMLV30_MODE_USER_27	DInt	30	OMAC PackMLV30 user-defined unit mode 27
LPMLV30_MODE_USER_28	DInt	31	OMAC PackMLV30 user-defined unit mode 28

States

Table 2-19: Constants for states

Name	Data type	Value	Comment
LPMLV30_STATE_UNDEFINED	DInt	0	OMAC PackMLV30 state Undefined
LPMLV30_STATE_CLEARING	DInt	1	OMAC PackMLV30 state Clearing
LPMLV30_STATE_STOPPED	DInt	2	OMAC PackMLV30 state Stopped
LPMLV30_STATE_STARTING	DInt	3	OMAC PackMLV30 state Starting
LPMLV30_STATE_IDLE	DInt	4	OMAC PackMLV30 state Idle
LPMLV30_STATE_SUSPENDED	DInt	5	OMAC PackMLV30 state Suspended
LPMLV30_STATE_EXECUTE	DInt	6	OMAC PackMLV30 state Execute
LPMLV30_STATE_STOPPING	DInt	7	OMAC PackMLV30 state Stopping
LPMLV30_STATE_ABORTING	DInt	8	OMAC PackMLV30 state Aborting
LPMLV30_STATE_ABORTED	DInt	9	OMAC PackMLV30 state Aborted
LPMLV30_STATE_HOLDING	DInt	10	OMAC PackMLV30 state Holding
LPMLV30_STATE_HELD	DInt	11	OMAC PackMLV30 state Held
LPMLV30_STATE_UNHOLDING	DInt	12	OMAC PackMLV30 state Unholding
LPMLV30_STATE_SUSPENDING	DInt	13	OMAC PackMLV30 state Suspending
LPMLV30_STATE_UNSUSPENDING	DInt	14	OMAC PackMLV30 state Unsuspending
LPMLV30_STATE_RESETTING	DInt	15	OMAC PackMLV30 state Resetting
LPMLV30_STATE_COMPLETING	DInt	16	OMAC PackMLV30 state Completing
LPMLV30_STATE_COMPLETE	DInt	17	OMAC PackMLV30 state Complete

Control commands

Table 2-20: Constants for control commands

Name	Data type	Value	Comment
LPMLV30_CMD_UNDEFINED	DInt	0	OMAC PackMLV30 control command Undefined
LPMLV30_CMD_RESET	DInt	1	OMAC PackMLV30 control command Reset
LPMLV30_CMD_START	DInt	2	OMAC PackMLV30 control command Start
LPMLV30_CMD_STOP	DInt	3	OMAC PackMLV30 control command Stop
LPMLV30_CMD_HOLD	DInt	4	OMAC PackMLV30 control command Hold
LPMLV30_CMD_UNHOLD	DInt	5	OMAC PackMLV30 control command Unhold
LPMLV30_CMD_SUSPEND	DInt	6	OMAC PackMLV30 control command Suspend
LPMLV30_CMD_UNSUSPEND	DInt	7	OMAC PackMLV30 control command Unsuspend
LPMLV30_CMD_ABORT	DInt	8	OMAC PackMLV30 control command Abort
LPMLV30_CMD_CLEAR	DInt	9	OMAC PackMLV30 control command Clear
LPMLV30_CMD_COMPLETE[3]	DInt	10	OMAC PackMLV30 control command Complete

[3] The *complete* command is not described in *ISA Technical Report TR88.00.02 Machine and Unit States*, but exists in this solution to be compatible to the “PackML V3 Demo in MS Excel” example on the OMAC website and also to the previous version of this library. According to the standard the *SC* signal should be used instead of the *complete* command.

Languages

Table 2-21: Constants for languages

Name	Data type	Value	Comment
LPMLV30_LANGUAGE_1	Int	0	Texts in 1st language (default English)
LPMLV30_LANGUAGE_2	Int	1	Texts in 2nd language (default German)

Array boundaries

Table 2-22: Constants for array boundaries

Name	Data type	Value	Comment
LPMLV30_DIAG_BUFFER_UPPER_LIM	Int	15	Diagnostics buffer array upper boundary (0-based)
LPMLV30_LANGUAGES_UPPER_LIM	Int	1	(Number of languages - 1) -> Array[0..LPMLV30_LANGUAGES_UPPER_LIM]
LPMLV30_MODES_UPPER_LIM	Int	11	(Number of unit modes - 1) -> Array[0..LPMLV30_MODES_UPPER_LIM]
LPMLV30_STATES_UPPER_LIM	Int	17	(Number of states - 1) -> Array[0..LPMLV30_STATES_UPPER_LIM]
LPMLV30_MAX_MODES_UPPER_LIM	Int	31	(Maximum number of unit modes - 1) -> Array[0..LPMLV30_MAX_MODES_UPPER_LIM]

NOTE If a constant for array boundaries is changed, also the corresponding local constant in the LPMLV30 blocks must accordingly be changed. In addition, the corresponding array boundary in the LPMLV30 PLC data types must also be changed. To find all places, it is recommended to use the "Search in project" functionality (e.g. search for LPMLV30_LANGUAGES_UPPER_LIM).

3 Working with the Library

What will you find in this section?

This chapter consists of instructions for integrating the LPMLV30 library into your STEP 7 project and instructions for using the library blocks.

3.1 Integrating the library into STEP 7

The table below lists the steps for integrating the LPMLV30 library into your STEP 7 project. Subsequently, you can use the blocks of the LPMLV30 library.

Note The following section assumes that a STEP 7 project exists.

Table 3-1: Integrating the library into STEP 7

No.	Action	Note
1.	Extract the library LPMLV30_V3_x_x.zip to a local folder.	
2.	In TIA Portal select “Options” -> “Global libraries” -> “Open library…”.	
3.	Browse to the file LPMLV30.al14. It can be found in the subfolder LPMLV30 of the extracted zip file.	
4.	Open the global library in read-only mode.	
5.	The LPMLV30 library is now available in the task card “Global libraries”	

3.2 Integrating the library blocks into STEP 7

The table below lists the steps for integrating the blocks of the LPMLV30 library into your STEP 7 program.

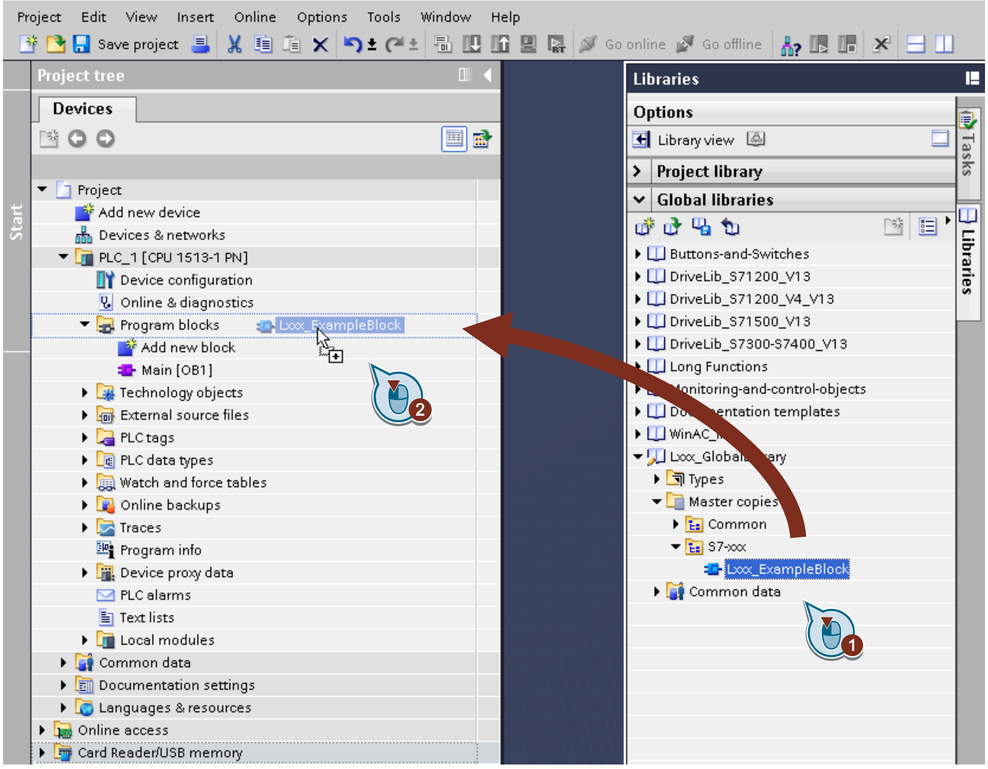


Table 3-2: Integrating the library blocks into STEP 7

No.	Action	Note
1.	Copy the *LPMLV30_UnitModeStateManager* FB in subfolder *LPMLV30_Blocks* with Drag & Drop into the “Program blocks” in the PLC. Alternatively copy the whole folder *LPMLV30_Blocks* via Drag & Drop into the “Program blocks” in the PLC. In this case also additional and optional blocks of LPMLV30 library are available in the user program (e.g. *LPMLV30_UnitModeStateTimes*, *LPMLV30_ConfigureDisabledStates*).	
2.	Copy the folder *LPMLV30_Tags* with Drag & Drop into the folder “PLC tags” in the PLC.	
3.	Copy the folder *LPMLV30_Types* with Drag & Drop into the folder “PLC data types” in the PLC. *typeLPMLV30_NamesConfiguration* is optional and needed only if FC *LPMLV30_GetUnitModeStateNamesAsString* is used.	
4.	Now the blocks can be configured and called in the user program.	

4 Notes and Support

What will you find in this section?

This chapter provides further support in handling the described LPMLV30 library.

5 Related literature

Table 5-1

	Topic	Title / Link
\1\	Siemens Industry Online Support	http://support.automation.siemens.com
\2\	Download page of this entry	https://support.industry.siemens.com/cs/ww/en/view/49970441
\3\	OMAC	http://www.omac.org
\4\	Packaging	http://www.siemens.com/packaging
\5\	SIMATIC CPG Template	https://support.industry.siemens.com/cs/ww/en/view/109475572

6 Application support

Siemens AG

Digital Industries
Factory Automation
Production Machines
DI FA PMA APC
Frauenauracher Str. 80
91056 Erlangen, Germany

mailto: tech.team.motioncontrol@siemens.com

7 History

Table 7-1

Version	Date	Modifications
V3.0	05/2015	First version
V3.0	09/2021	Scope of application is now STEP 7 Basic/Professional V14 Parameter *PackML_Version* added in *typeLPMLV30_Configuration*