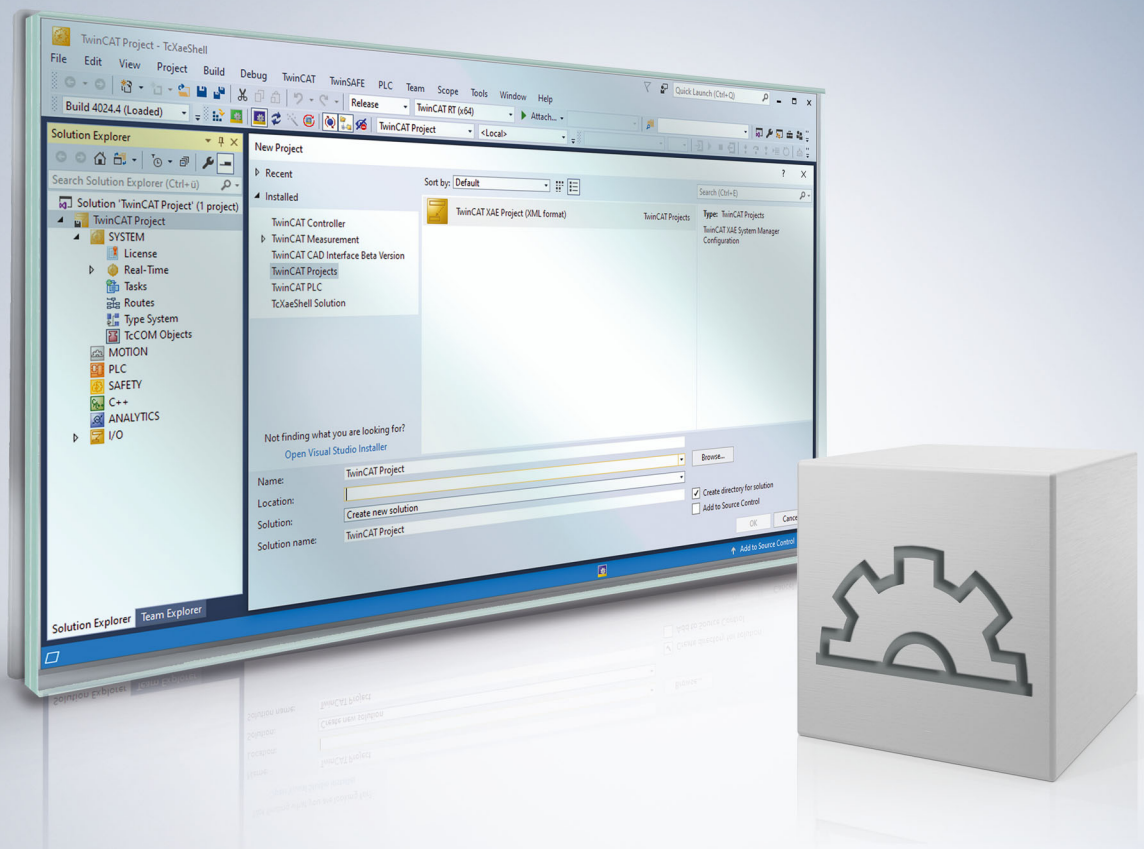
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc3_PackML



2023-09-11 | Version: 1.2.1

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Introduction

2.1 Overview

OMAC Packaging workgroup (OPW), a subset of the Open Modular Architecture Controls Users Group, has defined a set of functions to make an easy way for the end users for the control and automation of packaging machinery and systems.

General Objectives of the lib

- Machine related as first priority, process related as second
- Providing an easy-to-use interface to the Packaging Functionality
- Related to existing packaging standards
- Re-usable parts, usable in a wide range of applications
- Application program should be implementable on any platforms
- Accepted / User-tested Functionality / Concepts providing the basis for FBs
- Combining these FB's to an application program needs an environment that is suitable for Packaging related applications. Requirements and restrictions for such an environment are partly dealt with in this standard.

Packaging Machine Behavior Organization

PS_PackML_StateMachine_Auto [▶ 11]

PS_PackML_StateMachine_Maintenance [▶ 13]

PS_PackML_StateMachine_Manual [▶ 16]

PS_PackML_StateMachine_SemiAuto [▶ 19]

PS_UnitModeManager [▶ 22]

Development environment	Target system type	PLC libraries to be linked
TwinCAT 3.1 Build 4018 onwards	PC (i386)	Tc3_PackML

3 Packaging Machine State

Packaging Machine State Function Blocks provide a common interface to the existing PackML Machine State Model implementations available. It is expected that the application specific logic including the transitions between states is programmed in external function blocks, but the central logic of the state machine and the status representation should be handled by the Packaging Machine State Function block. Therefore, this FB comes with a recommendation how to combine with other logic.

The state transitions to a machine application are always application specific. Therefore, to facilitate standardization, it should be best practice to form state function blocks that are connected to the PackML State Machine V3. The state function blocks collect application specific signals anf form the transition logic to the adjacent states as displayed in the PackML state model. All state FB feed back into PackML State Machine V3, offering a standard state machine and state reporting. The state FBs will contain the machine execution code next to the application specific transition logic.

State Function Blocks are listed below and will be programmed by application programmer accordingly to maintain integrity and function of the PackML State Machine:

Pack ML State Machine V3 Function Block Names:

- PS_Starting
- PS_Completing
- PS_Resetting
- PS_Holding
- PS_UNHolding
- PS_Suspending
- PS_Clearing
- PS_Stopping
- PS_Aborting
- PS_Execute
- PS_Complete
- PS_Idle
- PS_Held
- PS_Suspended
- PS_Stopped
- PS_Aborted

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

3.1 DataTypes

3.1.1 E_PMLState

E_PMLState

```
TYPE E_PMLState : (
    (* states according to PackTags v3.0 *)
    ePMLState_UNDEFINED     := 0,
    ePMLState_CLEARING      := 1,
    ePMLState_STOPPED       := 2,
    ePMLState_STARTING      := 3,
    ePMLState_IDLE          := 4,
    ePMLState_SUSPENDED     := 5,
    ePMLState_EXECUTE       := 6,
    ePMLState_STOPPING      := 7,
```

```
    ePMLState_ABORTING      := 8,
    ePMLState_ABORTED       := 9,
    ePMLState_HOLDING       := 10,
    ePMLState_HELD          := 11,
    ePMLState_UNHOLDING     := 12,
    ePMLState_SUSPENDING    := 13,
    ePMLState_UNSUSPENDING  := 14,
    ePMLState_RESETTING     := 15,
    ePMLState_COMPLETING    := 16,
    ePMLState_COMPLETE      := 17
);
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

3.1.2 E_PMLUnitMode

E_PMLUnitMode

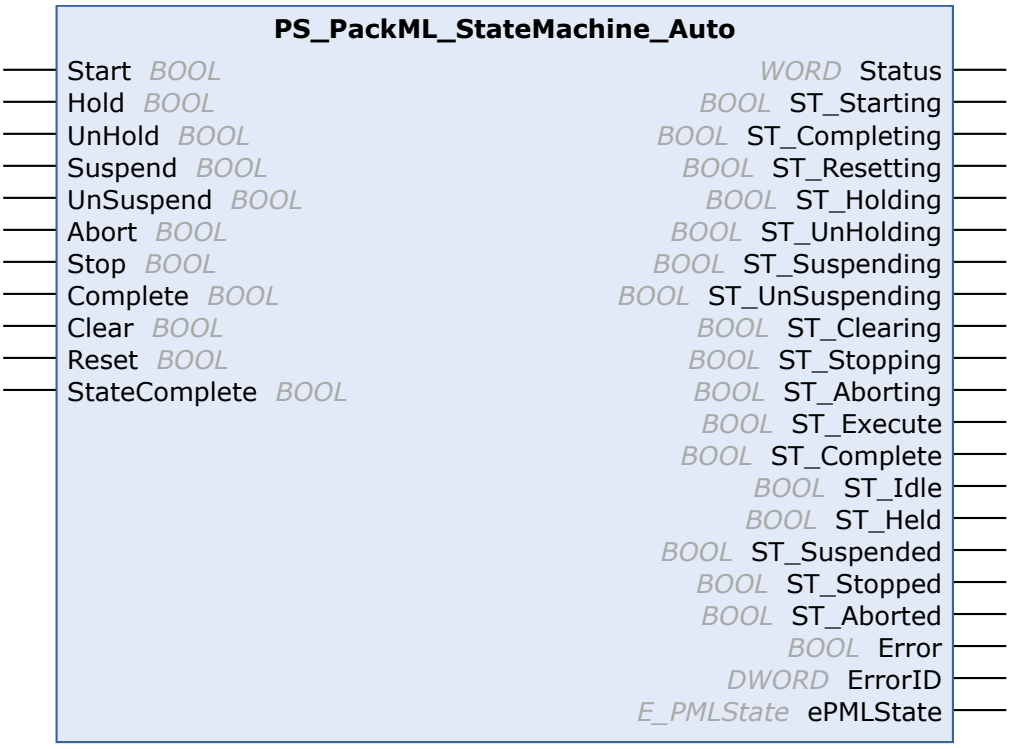
```
TYPE E_PMLUnitMode : (
    ePMLUnitMode_UNDEFINED      := 0,
    ePMLUnitMode_AUTOMATIC      := 1,
    ePMLUnitMode_MAINTENANCE    := 2,
    ePMLUnitMode_MANUAL         := 3,
    ePMLUnitMode_SEMIAUTOMATIC  := 4,
    ePMLUnitMode_DRYRUN         := 5,
    ePMLUnitMode_USERMODE1      := 6,
    ePMLUnitMode_USERMODE2      := 7,
    ePMLUnitMode_IDLE           := 8,
    ePMLUnitMode_ESTOP          := 9
);
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

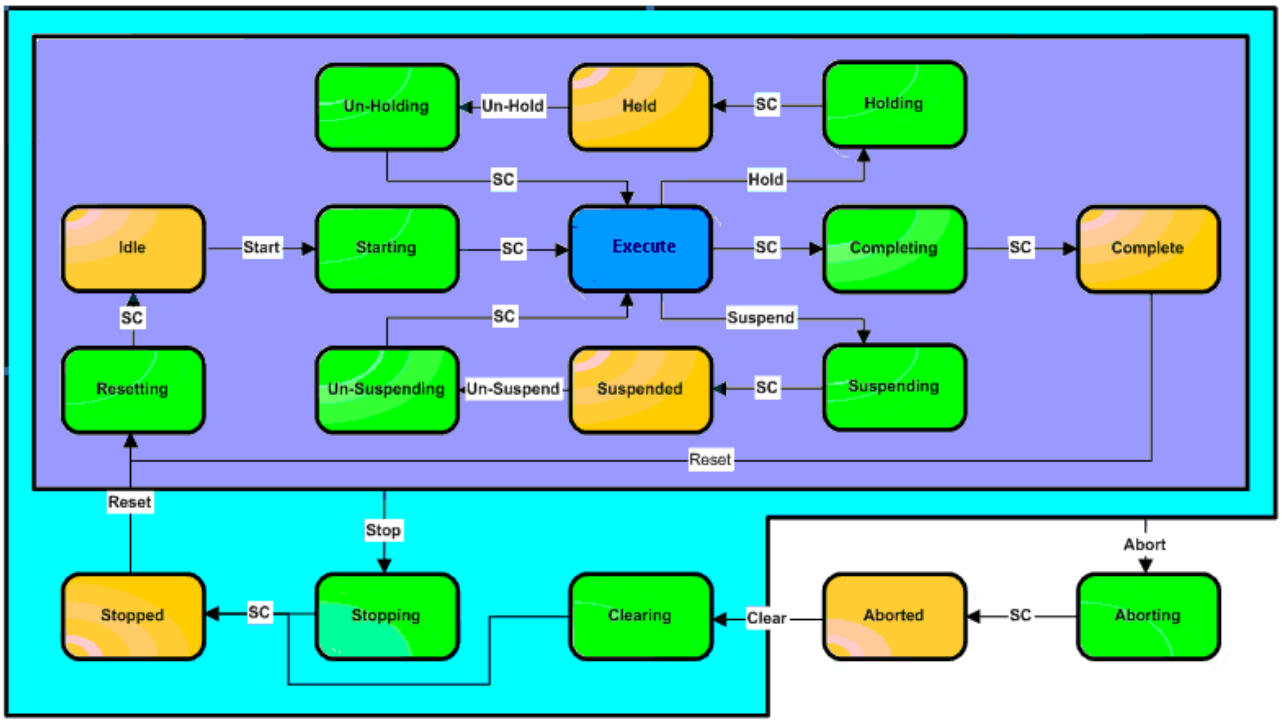
3.2 Function Blocks

3.2.1 PS_PackML_StateMachine_Auto



The function blocks for Packaging Machine State have a common interface to the PackML Machine State Model V3 in the updated form. It is expected that application-specific logic, such as state transitions, is programmed in external function blocks and that the Pack_ML_State_Machine function block handles the central logic of the state machine and the state representation. For that reason there is a recommendation for this function block with regard to how it can be combined with other logic.

The logic for transitions, especially between manual, semi-automatic and automatic modes (see figure), depends on the application.



Inputs

```
VAR_INPUT
    Start           : BOOL;
    Hold            : BOOL;
    unHold          : BOOL;
    Suspend         : BOOL;
    unSuspend       : BOOL;
    Abort           : BOOL;
    Stop            : BOOL;
    Complete        : BOOL;
    Clear           : BOOL;
    Reset           : BOOL;
    StateComplete   : BOOL;
END_VAR
```

Table/description: Execute state machine from rising edge...

Name	Type	Description
Start	BOOL	…to Starting.
Hold	BOOL	…to Holding or Held.
unHold	BOOL	…to Unholding.
Suspend	BOOL	...to Suspending or Suspend.
unSuspend	BOOL	…to Unsuspending.
Abort	BOOL	…to Aborting.
Stop	BOOL	…to Stopping.
Complete	BOOL	…to Resetting.
Clear	BOOL	…to Clearing.
Reset	BOOL	…to Resetting.
StateComplete	BOOL	Transition

Outputs

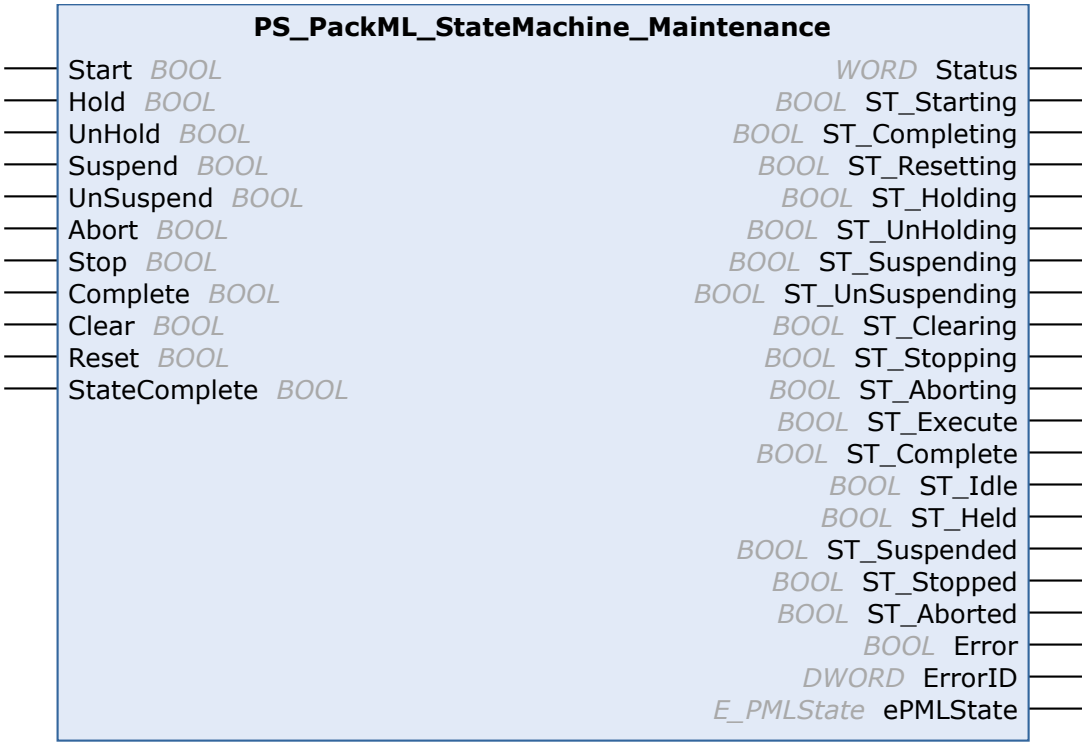
```
VAR_OUTPUT
    Status          : WORD;
    ST_Starting     : BOOL;
    ST_Completing   : BOOL;
    ST_Resetting    : BOOL;
    ST_Holding      : BOOL;
    ST_UnHolding    : BOOL;
    ST_Suspending   : BOOL;
    ST_UnSuspending : BOOL;
    ST_Clearing     : BOOL;
    ST_Stopping     : BOOL;
    ST_Aborting     : BOOL;
    ST_Execute      : BOOL;
    ST_Complete     : BOOL;
    ST_Idle         : BOOL;
    ST_Held         : BOOL;
    ST_Suspended    : BOOL;
    ST_Stopped      : BOOL;
    ST_Aborted      : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
    ePMLState       : E_PMLState;
END_VAR
```

Name	Type	Description
State	WORD	Status word representing the state of the state machine.
ST_Starting	BOOL	True if state machine is in Starting state.
ST_Completing	BOOL	True if state machine is in Completing state.
ST_Resetting	BOOL	True if state machine is in Resetting state.
ST_Holding	BOOL	True if state machine is in Holding state.
ST_UnHolding	BOOL	True if state machine is in Unholding state.
ST_Suspending	BOOL	True if state machine is in Suspending state.
ST_UnSuspending	BOOL	True if state machine is in Unsuspending state.
ST_Clearing	BOOL	True if state machine is in Clearing state.
ST_Stopping	BOOL	True if state machine is in Stopping state.
ST_Aborting	BOOL	True if state machine is in Aborting state.
ST_Execute	BOOL	True if state machine is in Execute state.
ST_Complete	BOOL	True if state machine in Complete state.
ST_Idle	BOOL	True if state machine is in Idle state.
ST_Held	BOOL	True if state machine in Held state.
ST_Suspended	BOOL	True if state machine is in Suspended state.
ST_Stopped	BOOL	True if state machine is in Stopped state.
ST_Aborted	BOOL	True if state machine is in Aborted state.
Error	BOOL	Becomes TRUE, as soon as an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
ePMLState	E_PMLState	Current PML state of the automatic state machine.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

3.2.2 PS_PackML_StateMachine_Maintenance



The function blocks for Packaging Machine State have a common interface to the PackML Machine State Model V3 in the updated form. It is expected that application-specific logic, such as state transitions, is programmed in external function blocks and that the Pack_ML_State_Machine function block handles the central logic of the state machine and the state representation. For that reason there is a recommendation for this function block with regard to how it can be combined with other logic.

The logic for transitions, especially between manual, semi-automatic and automatic modes (see figure), depends on the application.

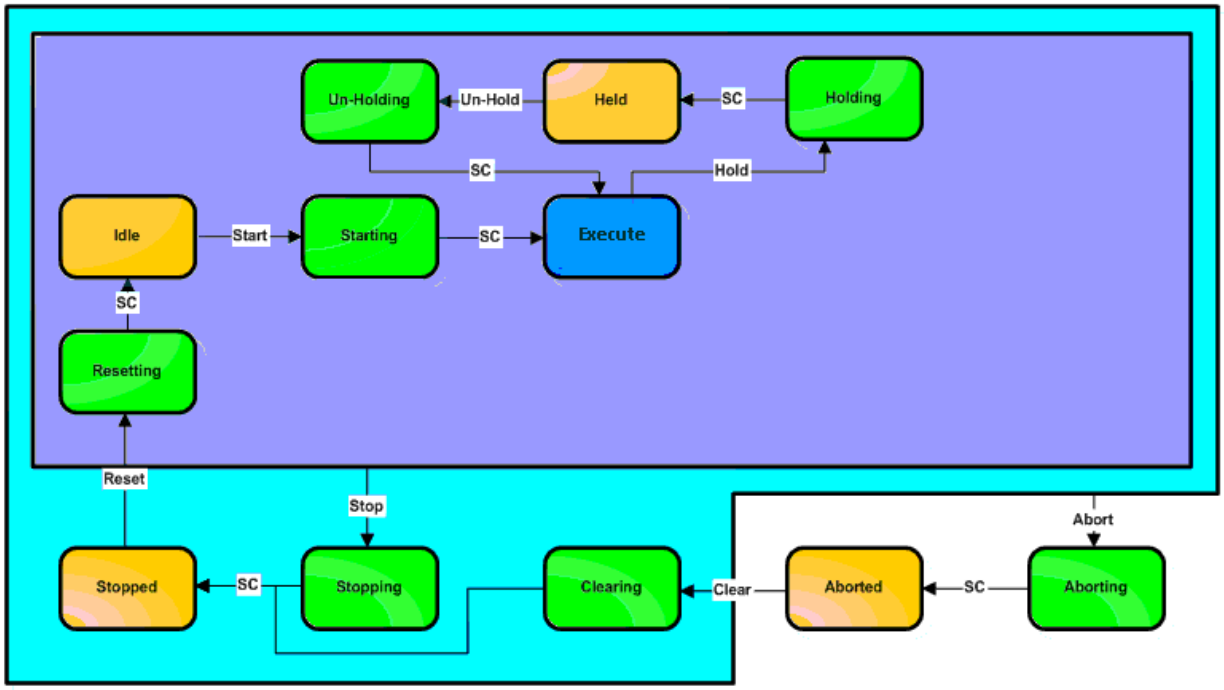


Figure : PS_Pack_ML_State_Model_Maintenance Mode

Inputs

```
VAR_INPUT
    Start           : BOOL;
    Hold            : BOOL;
    unHold          : BOOL;
    Suspend         : BOOL;
    unSuspend       : BOOL;
    Abort           : BOOL;
    Stop            : BOOL;
    Complete        : BOOL;
    Clear           : BOOL;
    Reset           : BOOL;
    StateComplete   : BOOL;
END_VAR
```

Table/description: Execute state machine from rising edge...

Name	Type	Description
Start	BOOL	…to Starting.
Hold	BOOL	…to Holding or Held.
unHold	BOOL	…to Unholding.
Suspend	BOOL	...to Suspending or Suspend.
unSuspend	BOOL	…to Unsuspending.
Abort	BOOL	…to Aborting.
Stop	BOOL	…to Stopping.
Complete	BOOL	…to Resetting.
Clear	BOOL	…to Clearing.
Reset	BOOL	…to Resetting.
StateComplete	BOOL	Transition

Outputs

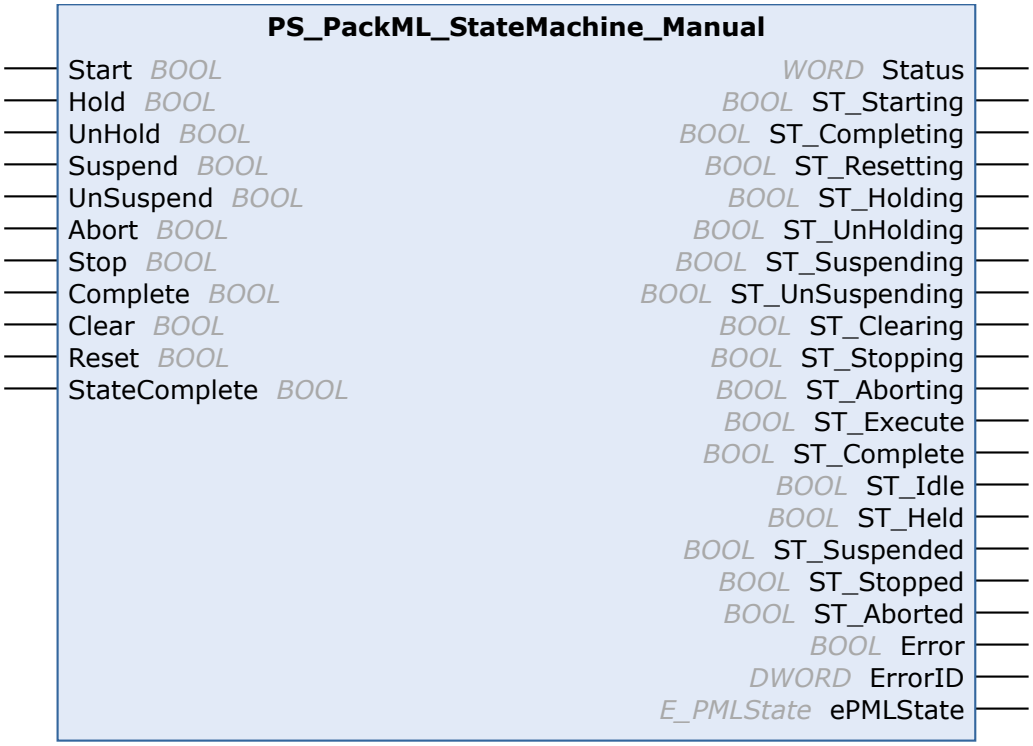
```
VAR_OUTPUT
    Status          : WORD;
    ST_Starting     : BOOL;
    ST_Completing   : BOOL;
    ST_Resetting    : BOOL;
    ST_Holding      : BOOL;
    ST_UnHolding    : BOOL;
    ST_Suspending   : BOOL;
    ST_UnSuspending : BOOL;
    ST_Clearing     : BOOL;
    ST_Stopping     : BOOL;
    ST_Aborting     : BOOL;
    ST_Execute      : BOOL;
    ST_Complete     : BOOL;
    ST_Idle         : BOOL;
    ST_Held         : BOOL;
    ST_Suspended    : BOOL;
    ST_Stopped      : BOOL;
    ST_Aborted      : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
    ePMLState       : E_PMLState;
END_VAR
```

Name	Type	Description
State	WORD	Status word representing the state of the state machine.
ST_Starting	BOOL	True if state machine is in Starting state.
ST_Completing	BOOL	True if state machine is in Completing state.
ST_Resetting	BOOL	True if state machine is in Resetting state.
ST_Holding	BOOL	True if state machine is in Holding state.
ST_UnHolding	BOOL	True if state machine is in Unholding state.
ST_Suspending	BOOL	True if state machine is in Suspending state.
ST_UnSuspending	BOOL	True if state machine is in Unsuspending state.
ST_Clearing	BOOL	True if state machine is in Clearing state.
ST_Stopping	BOOL	True if state machine is in Stopping state.
ST_Aborting	BOOL	True if state machine is in Aborting state.
ST_Execute	BOOL	True if state machine is in Execute state.
ST_Complete	BOOL	True if state machine in Complete state.
ST_Idle	BOOL	True if state machine is in Idle state.
ST_Held	BOOL	True if state machine in Held state.
ST_Suspended	BOOL	True if state machine is in Suspended state.
ST_Stopped	BOOL	True if state machine is in Stopped state.
ST_Aborted	BOOL	True if state machine is in Aborted state.
Error	BOOL	Becomes TRUE, as soon as an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
ePMLState	E_PMLState	Current PML state of the automatic state machine.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

3.2.3 PS_PackML_StateMachine_Manual



The Packaging Machine State function blocks have a common interface to the PackML Machine State Model V3 in the updated form. It is expected that application-specific logic, such as state transitions, is programmed in external function blocks and that the Pack_ML_State_Machine function block handles the central logic of the state machine and the state representation. For that reason there is a recommendation for this function block with regard to how it can be combined with other logic.

The logic for transitions, especially between manual, semi-automatic and automatic modes (see figure), depends on the application.

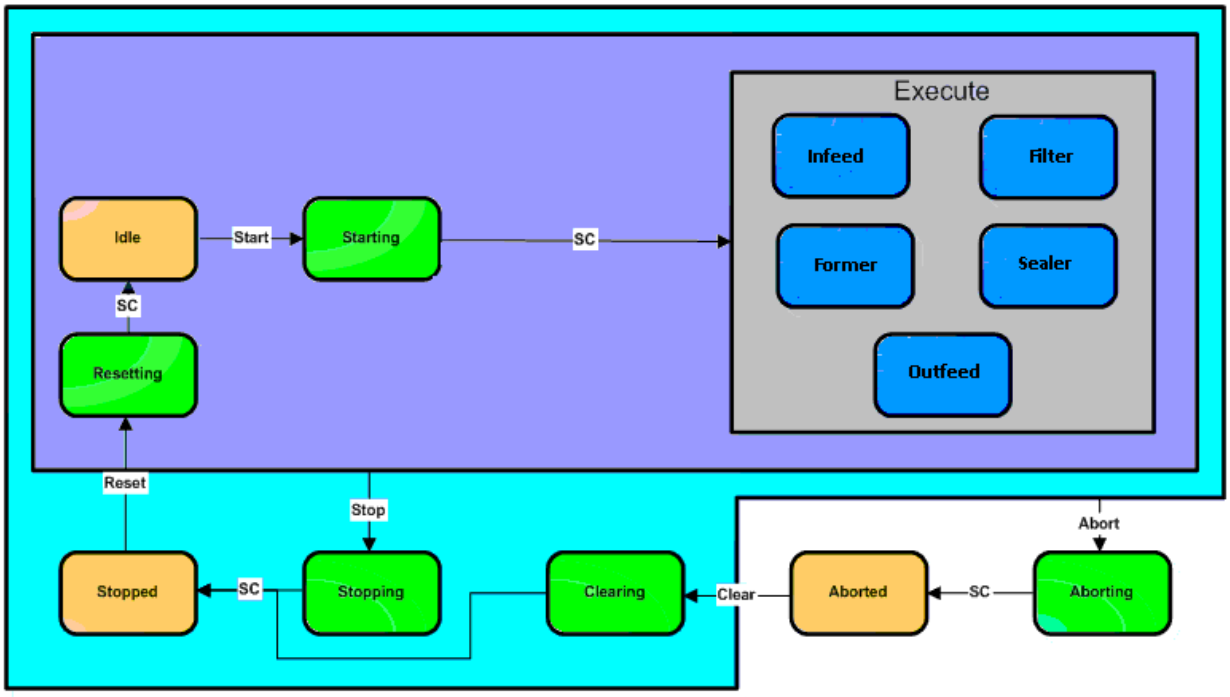


Figure : PS_Pack_ML_State_Model_Manual Mode

Inputs

```
VAR_INPUT
    Start           : BOOL;
    Hold            : BOOL;
    unHold          : BOOL;
    Suspend         : BOOL;
    unSuspend       : BOOL;
    Abort           : BOOL;
    Stop            : BOOL;
    Complete        : BOOL;
    Clear           : BOOL;
    Reset           : BOOL;
    StateComplete   : BOOL;
END_VAR
```

Table/description: Execute state machine from rising edge...

Name	Type	Description
Start	BOOL	…to Starting.
Hold	BOOL	…to Holding or Held.
unHold	BOOL	…to Unholding.
Suspend	BOOL	...to Suspending or Suspend.
unSuspend	BOOL	…to Unsuspending.
Abort	BOOL	…to Aborting.
Stop	BOOL	…to Stopping.
Complete	BOOL	…to Resetting.
Clear	BOOL	…to Clearing.
Reset	BOOL	…to Resetting.
StateComplete	BOOL	Transition

Outputs

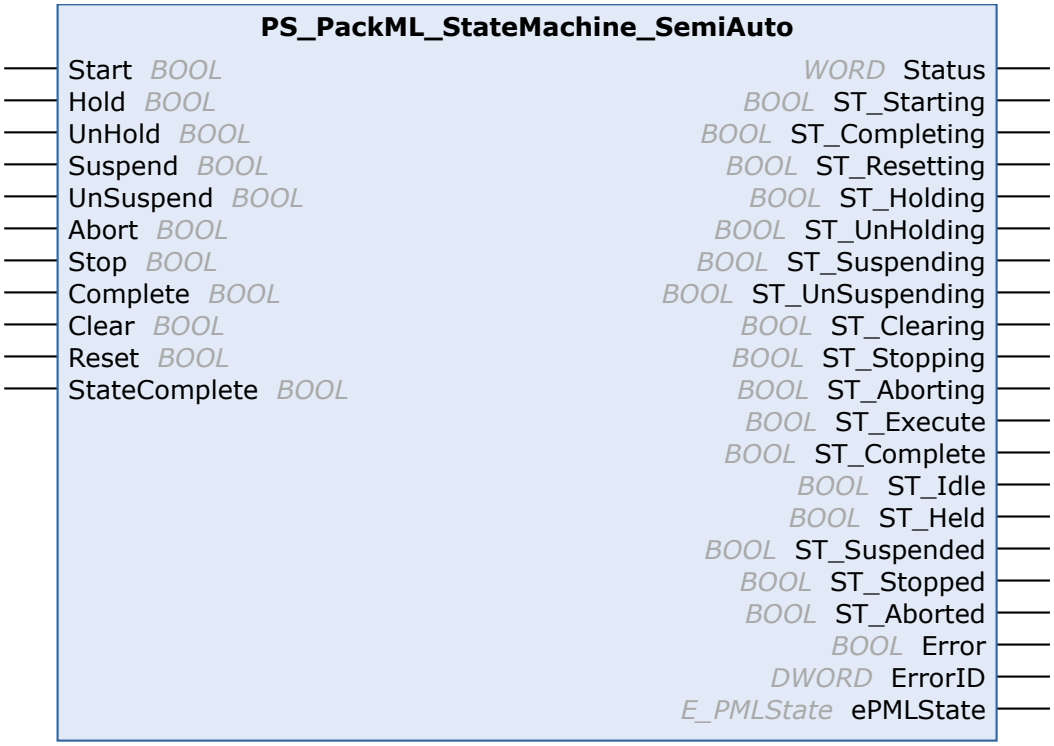
```
VAR_OUTPUT
    Status          : WORD;
    ST_Starting     : BOOL;
    ST_Completing   : BOOL;
    ST_Resetting    : BOOL;
    ST_Holding      : BOOL;
    ST_UnHolding    : BOOL;
    ST_Suspending   : BOOL;
    ST_UnSuspending : BOOL;
    ST_Clearing     : BOOL;
    ST_Stopping     : BOOL;
    ST_Aborting     : BOOL;
    ST_Execute      : BOOL;
    ST_Complete     : BOOL;
    ST_Idle         : BOOL;
    ST_Held         : BOOL;
    ST_Suspended    : BOOL;
    ST_Stopped      : BOOL;
    ST_Aborted      : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
    ePMLState       : E_PMLState;
END_VAR
```

Name	Type	Description
State	WORD	Status word representing the state of the state machine.
ST_Starting	BOOL	True if state machine is in Starting state.
ST_Completing	BOOL	True if state machine is in Completing state.
ST_Resetting	BOOL	True if state machine is in Resetting state.
ST_Holding	BOOL	True if state machine is in Holding state.
ST_UnHolding	BOOL	True if state machine is in Unholding state.
ST_Suspending	BOOL	True if state machine is in Suspending state.
ST_UnSuspending	BOOL	True if state machine is in Unsuspending state.
ST_Clearing	BOOL	True if state machine is in Clearing state.
ST_Stopping	BOOL	True if state machine is in Stopping state.
ST_Aborting	BOOL	True if state machine is in Aborting state.
ST_Execute	BOOL	True if state machine is in Execute state.
ST_Complete	BOOL	True if state machine in Complete state.
ST_Idle	BOOL	True if state machine is in Idle state.
ST_Held	BOOL	True if state machine in Held state.
ST_Suspended	BOOL	True if state machine is in Suspended state.
ST_Stopped	BOOL	True if state machine is in Stopped state.
ST_Aborted	BOOL	True if state machine is in Aborted state.
Error	BOOL	Becomes TRUE, as soon as an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
ePMLState	E_PMLState	Current PML state of the automatic state machine.

Requirements

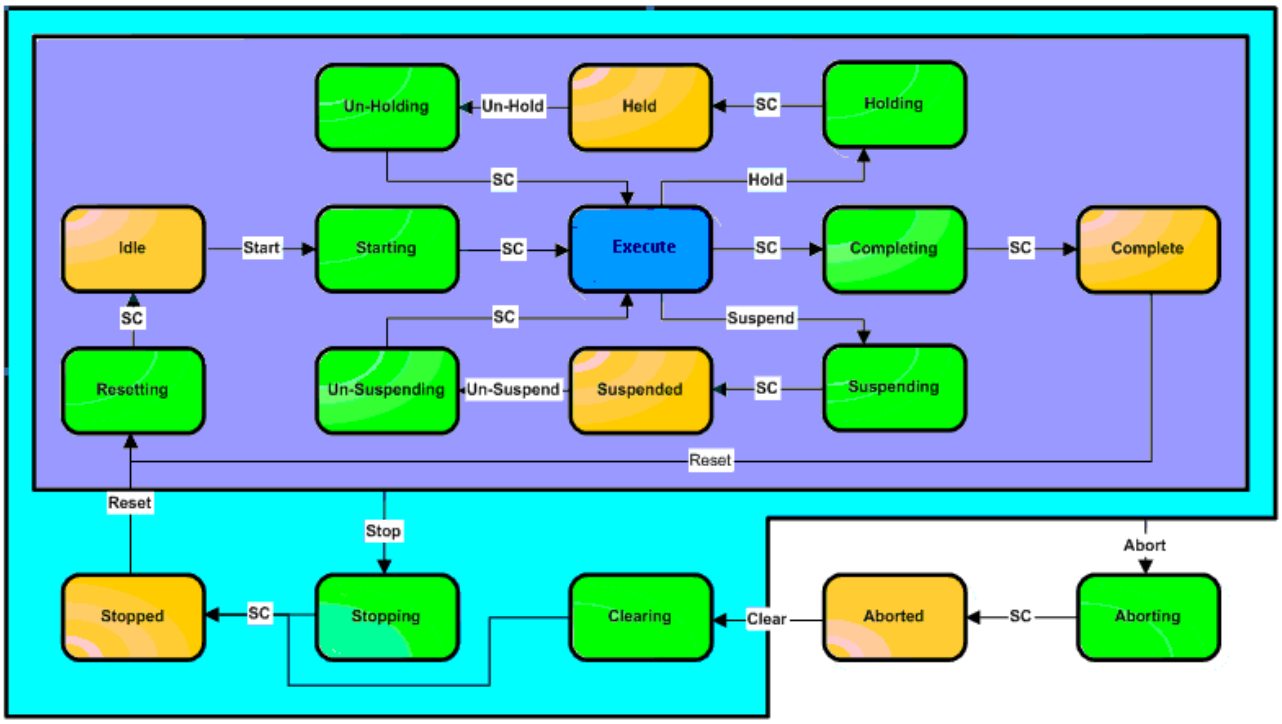
Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

3.2.4 PS_PackML_StateMachine_SemiAuto



The Packaging Machine State function blocks have a common interface to the PackML Machine State Model V3 in the updated form. It is expected that application-specific logic, such as state transitions, is programmed in external function blocks and that the Pack_ML_State_Machine function block handles the central logic of the state machine and the state representation. For that reason there is a recommendation for this function block with regard to how it can be combined with other logic.

The logic for transitions, especially between manual, semi-automatic and automatic modes (see figure), depends on the application.



Inputs

```
VAR_INPUT
    Start           : BOOL;
    Hold            : BOOL;
    unHold          : BOOL;
    Suspend         : BOOL;
    unSuspend       : BOOL;
    Abort           : BOOL;
    Stop            : BOOL;
    Complete        : BOOL;
    Clear           : BOOL;
    Reset           : BOOL;
    StateComplete   : BOOL;
END_VAR
```

Table/description: Execute state machine from rising edge...

Name	Type	Description
Start	BOOL	…to Starting.
Hold	BOOL	…to Holding or Held.
unHold	BOOL	…to Unholding.
Suspend	BOOL	...to Suspending or Suspend.
unSuspend	BOOL	…to Unsuspending.
Abort	BOOL	…to Aborting.
Stop	BOOL	…to Stopping.
Complete	BOOL	…to Resetting.
Clear	BOOL	…to Clearing.
Reset	BOOL	…to Resetting.
StateComplete	BOOL	Transition

Outputs

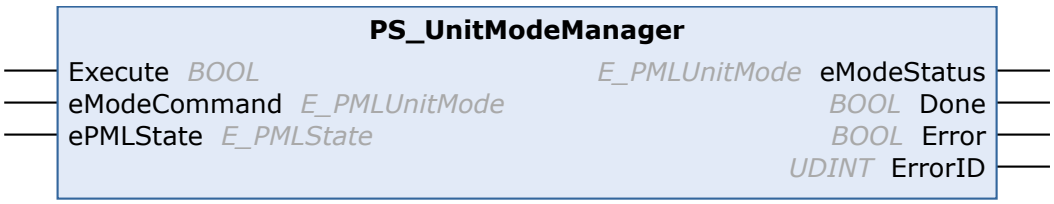
```
VAR_OUTPUT
    Status          : WORD;
    ST_Starting     : BOOL;
    ST_Completing   : BOOL;
    ST_Resetting    : BOOL;
    ST_Holding      : BOOL;
    ST_UnHolding    : BOOL;
    ST_Suspending   : BOOL;
    ST_UnSuspending : BOOL;
    ST_Clearing     : BOOL;
    ST_Stopping     : BOOL;
    ST_Aborting     : BOOL;
    ST_Execute      : BOOL;
    ST_Complete     : BOOL;
    ST_Idle         : BOOL;
    ST_Held         : BOOL;
    ST_Suspended    : BOOL;
    ST_Stopped      : BOOL;
    ST_Aborted      : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
    ePMLState       : E_PMLState;
END_VAR
```

Name	Type	Description
State	WORD	Status word representing the state of the state machine.
ST_Starting	BOOL	True if state machine is in Starting state.
ST_Completing	BOOL	True if state machine is in Completing state.
ST_Resetting	BOOL	True if state machine is in Resetting state.
ST_Holding	BOOL	True if state machine is in Holding state.
ST_UnHolding	BOOL	True if state machine is in Unholding state.
ST_Suspending	BOOL	True if state machine is in Suspending state.
ST_UnSuspending	BOOL	True if state machine is in Unsuspending state.
ST_Clearing	BOOL	True if state machine is in Clearing state.
ST_Stopping	BOOL	True if state machine is in Stopping state.
ST_Aborting	BOOL	True if state machine is in Aborting state.
ST_Execute	BOOL	True if state machine is in Execute state.
ST_Complete	BOOL	True if state machine in Complete state.
ST_Idle	BOOL	True if state machine is in Idle state.
ST_Held	BOOL	True if state machine in Held state.
ST_Suspended	BOOL	True if state machine is in Suspended state.
ST_Stopped	BOOL	True if state machine is in Stopped state.
ST_Aborted	BOOL	True if state machine is in Aborted state.
Error	BOOL	Becomes TRUE, as soon as an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
ePMLState	E_PMLState	Current PML state of the automatic state machine.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

3.2.5 PS_UnitModeManager



Packaging machines have different unit modes and not only the "automatic" mode. Each unit mode is defined by its own state model. A "Mode Manager" must be defined for transitions between the modes. The "Mode Manager" determines how and in which state a machine can change unit modes; i.e. built-in barriers prevent the machine from changing to unsuitable states.

See figure below as an example.

⚠ WARNING

Adhere to proper mode changes

The logic for transitions between modes depends on the application, especially for transitions between manual, semi-automatic and automatic modes. In addition, hardware barriers or safety equipment may be necessary for such mode changes. The responsibility for proper mode changes lies with whoever implements them.

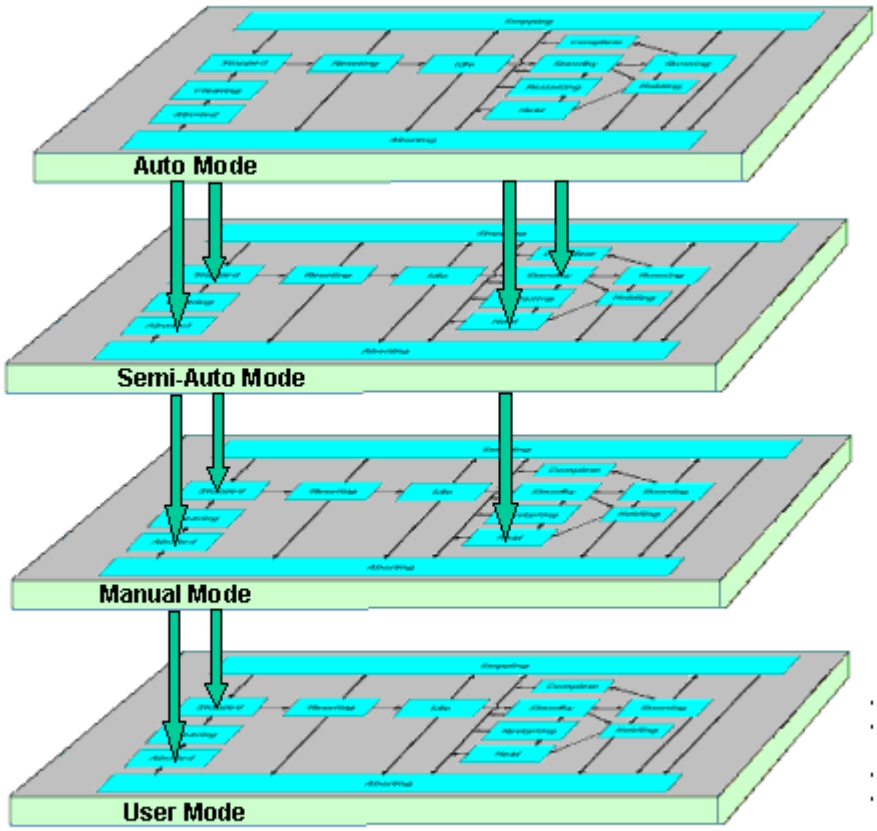


Figure : User Mode Operations State Model

Inputs

```
VAR_INPUT
    Execute         : BOOL;
    eModeCommand    : E_PMLUnitMode;
    ePMLState       : E_PMLState;
END_VAR
```

Name	Type	Description
Execute	BOOL	Mode change on rising edge.
eModeCommand	E_PMLUnitMode	Requested unit mode (E_PMLUnitMode [▶ 10])
ePMLState	E_PMLState	Current PML state of the current mode (E_PMLState [▶ 9])

Inputs/Outputs

```
VAR_IN_OUT
    Machine_ID      : MACHINE_REF;    (* Identifies the axis which position shall be latched at the
trigger event *)
END_VAR
```

Name	Type	Description
Machine_ID	MACHINE_REF	Identification of the machine executed by the state model

Outputs

```
VAR_OUTPUT
    eModeStatus     : E_PMLUnitMode;
    Done            : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
END_VAR
```

Name	Type	Description
eModeStatus	E_PMLUnitMode	Current unit mode (E_PMLUnitMode [▶ 10])
Done	BOOL	True if mode change was successful.
Error	BOOL	Signals an error in the function block, e.g. "Mode change not permitted".
ErrorID	UDINT	If the error output is set, this parameter returns the error number. 0 = no error; 1 = mode change not allowed. ePMLState not idle, stopped, aborted, held, suspended or completed or the corresponding state does not exist in the requested mode.

Implementation

Mode change is limited to certain modes, see implementation below.

Not all modes are implemented yet.

```
rTrig(CLK:= Execute);
IF rTrig.Q THEN
    Done := FALSE;
    Error := FALSE;
    ErrorID := 0;
    CASE eModeStatus OF
    ePMLUnitMode_AUTOMATIC:
        IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLSt
ate_IDLE) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSIF ((ePMLState = ePMLState_SUSPENDED) OR (ePMLState = ePMLState_HELD) OR (ePMLState = ePM
LState_COMPLETE))
        AND (eModeCommand = ePMLUnitMode_SEMIAUTOMATIC) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSIF (ePMLState = ePMLState_HELD) AND (eModeCommand = ePMLUnitMode_MAINTENANCE) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSE
            Error := TRUE;
```

```
            ErrorID := 1;
        END_IF
    ePMLUnitMode_MAINTENANCE:
        IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLSt
ate_IDLE) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSIF (ePMLState = ePMLState_HELD) AND ((eModeCommand = ePMLUnitMode_AUTOMATIC) OR (eModeCom
mand = ePMLUnitMode_SEMIAUTOMATIC))         THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSE
            Error := TRUE;
            ErrorID := 1;
        END_IF
    ePMLUnitMode_MANUAL:
        IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLSt
ate_IDLE) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSE
            Error := TRUE;
            ErrorID := 1;
        END_IF
    ePMLUnitMode_SEMIAUTOMATIC:
        IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLSt
ate_IDLE) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSIF ((ePMLState = ePMLState_SUSPENDED) OR (ePMLState = ePMLState_HELD) OR (ePMLState = ePM
LState_COMPLETE))
        AND (eModeCommand = ePMLUnitMode_AUTOMATIC) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSIF (ePMLState = ePMLState_HELD) AND (eModeCommand = ePMLUnitMode_MAINTENANCE) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSE
            Error := TRUE;
            ErrorID := 1;
        END_IF
    ePMLUnitMode_IDLE:
        IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLSt
ate_IDLE) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSE
            Error := TRUE;
            ErrorID := 1;
        END_IF
    ePMLUnitMode_ESTOP:
        IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLSt
ate_IDLE) THEN
            eModeStatus := eModeCommand;
            Done := TRUE;
        ELSE
            Error := TRUE;
            ErrorID := 1;
        END_IF
    ELSE
        eModeStatus := eModeCommand;
        Done := TRUE;
   END_CASE
END_IF
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4 Packaging Machine Functional Tag Description

4.1 Introduction

PackTags provide a uniform set of naming conventions for data elements used within the procedural elements of the base state model. As seen earlier in the document the Base State Model provides a uniform set of machine states, so that all automated machinery can be looked at in a common way. PackTags are named data elements used for open architecture, interoperable data exchange in automated machinery. This document includes the fundamental names of the data elements as well as the data type, values, ranges and where necessary, data structures. PackTags are useful for machine-to-machine (intermachine) communications; for example between a Filler and a Capper. PackTags can also be used for data exchange between machines and higher-level information systems like Manufacturing Operations Management and Enterprise Information Systems.

This document defines all the PackTags necessary to navigate through a state model, as well as those that are required to define and manipulate the unit control mode. This document also defines a list of PackTags that will provide necessary information that might be available from a machine. The use of all PackTags is needed to be consistent with the principles for integrated connectivity with systems using this same implementation method.

Required tags are those necessary for the function of the automated machine or the connectivity to supervisory or remote systems.

4.2 Tag Types

PackTags are broken out into three groups; Command, Status and Administration. Command and Status tags contain data required for interfacing between machines and line control for coordination, or for recipe / parameter download. Command tags are "written" to and consumed by the machine program, as the “Information Receiver”, while status tags are produced by and read from the machine program. Administration Tags contain data collected by higher level systems for machine performance analysis, or operator information.

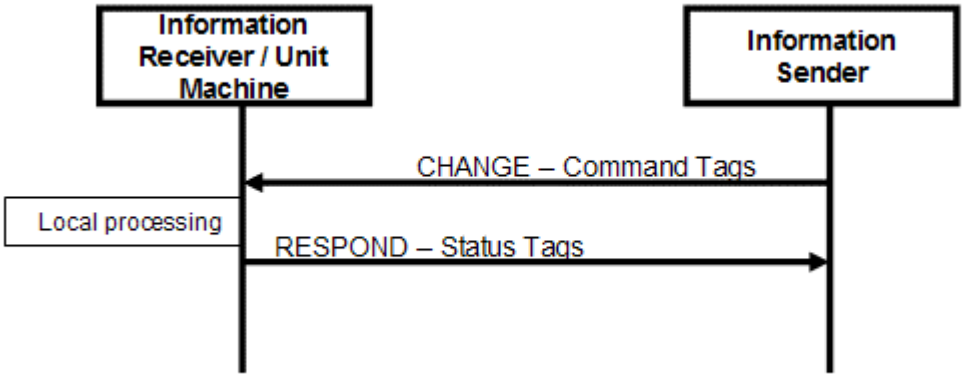
Each grouping of data should be in a contiguous grouping of registers to optimise communications.

Generally informational data is passed using OPC on an Ethernet-based communication network.

Command Tags are prefixed by “PMLc”.

Status Tags are prefixed by “PMLs”.

Administration Tags are prefixed by “PMLa”.



4.3 Tag Details

The following section is a summary listing of the tags. Tables below list the command, status and admin PackTags.

Command structure PMLc

				Tag Name	Data Type
PMLc				PMLc	ST_PMLc
	UnitMode			PMLc.UnitMode	DINT
	UnitModeChangeRequest			PMLc.UnitModeChangeRequest	BOOL
	ProcMode			PMLc.ProcMode	DINT
	ProcModeChangeRequest			PMLc.ProcModeChangeRequest	BOOL
	CurMachSpeed			PMLc.CurMachSpeed	DINT
	MatReady			PMLc.MatReady	ST_Material
		RawMaterial		PMLc.MatReady.RawMaterial	BOOL
		CO2		PMLc.MatReady.CO2	BOOL
		Container		PMLc.MatReady.Container	BOOL
		Lubrication		PMLc.MatReady.Lubrication	BOOL
		Water		PMLc.MatReady.Water	BOOL
		ContainerClosures		PMLc.MatReady.ContainerClosure	BOOL
		Unused0		PMLc.MatReady.Unused0	BOOL
		Unused1		PMLc.MatReady.Unused1	BOOL
		Unused2		PMLc.MatReady.Unused2	BOOL
		Unused3		PMLc.MatReady.Unused3	BOOL
		Unused4		PMLc.MatReady.Unused4	BOOL
		Unused5		PMLc.MatReady.Unused5	BOOL
		Unused6		PMLc.MatReady.Unused6	BOOL
		Unused7		PMLc.MatReady.Unused7	BOOL
		Unused8		PMLc.MatReady.Unused8	BOOL
		Unused9		PMLc.MatReady.Unused9	BOOL
	MatLow			PMLc.MatLow	ST_Material
		RawMaterial		PMLc.MatLow.RawMaterial	BOOL
		CO2		PMLc.MatLow.CO2	BOOL
		Container		PMLc.MatLow.Container	BOOL
		Lubrication		PMLc.MatLow.Lubrication	BOOL
		Water		PMLc.MatLow.Water	BOOL
		ContainerClosures		PMLc.MatLow.ContainerClosures	BOOL
		Unused0		PMLc.MatLow.Unused0	BOOL
		Unused1		PMLc.MatLow.Unused1	BOOL
		Unused2		PMLc.MatLow.Unused2	BOOL
		Unused3		PMLc.MatLow.Unused3	BOOL
		Unused4		PMLc.MatLow.Unused4	BOOL
		Unused5		PMLc.MatLow.Unused5	BOOL
		Unused6		PMLc.MatLow.Unused6	BOOL
		Unused7		PMLc.MatLow.Unused7	BOOL
		Unused8		PMLc.MatLow.Unused8	BOOL
		Unused9		PMLc.MatLow.Unused9	BOOL
	State			PMLc.State	DINT
	StateChangeRequest			PMLc.StateChangeRequest	BOOL
	CntrlCmd			PMLc.CntrlCmd	DINT
	Node[#]			PMLc.Node[#]	ST_Node
		Number		PMLc.Node[#].Number	DINT
		ControlCmdNumber		PMLc.Node[#].ControlCmdNumber	DINT
		CmdValue		PMLc.Node[#]CmdValue	DINT
		Parameter[#]		PMLc.Node[#].Parameter[#]	ST_Descriptor
			Id	PMLc.Node[#].Parameter[#].Id	DINT
			Name	PMLc.Node[#].Parameter[#].Name	STRING
			Unit	PMLc.Node[#].Parameter[#].Unit	STRING
			Value	PMLc.Node[#].Parameter[#].Value	REAL
	ProcessVariables[#]			PMLc.ProcessVariables[#]	ST_Descriptor
		Id		PMLc.ProcessVariables[#].Id	DINT
		Name		PMLc.ProcessVariables[#].Name	STRING
		Unit		PMLc.ProcessVariables[#].Unit	STRING
		Value		PMLc.ProcessVariables[#].Values	REAL

Product[#]				PMLc.Product[#]	ST_Product
	ProductId			PMLc.Product[#].ProductId	DINT
	ProcessVariables[#]			PMLc.Product[#].ProcessVariables[#]	ST_Descriptor
		Id		PMLc.Product[#].ProcessVariables[#].Id	DINT
		Name		PMLc.Product[#].ProcessVariables[#].Name	STRING
		Unit		PMLc.Product[#].ProcessVariables[#].Unit	STRING
		Value		PMLc.Product[#].ProcessVariables[#].Value	REAL
	Ingredients[#]			PMLc.Product[#].Ingredients[#]	ST_Ingredient
		IngredientId		PMLc.Product[#].Ingredients[#].IngredientId	DINT
		Parameter[#]		PMLc.Product[#].Ingredients[#].Parameter[#]	ST_Descriptor
			Id	PMLc.Product[#].Ingredients[#].Parameter[#].Id	DINT
			Name	PMLc.Product[#].Ingredients[#].Parameter[#].Name	STRING
			Unit	PMLc.Product[#].Ingredients[#].Parameter[#].Unit	STRING
			Value	PMLc.Product[#].Ingredients[#].Parameter[#].Value	REAL
Limits[#]				PMLc.Limits[#]	ST_Descriptor
	Id			PMLc.Limits[#].Id	DINT
	Name			PMLc.Limits[#].Name	STRING
	Unit			PMLc.Limits[#].Unit	STRING
	Value			PMLc.Limits[#].Value	REAL
TargetDownstreamNodeID				PMLc.TargetDownstreamNodeId	DINT
TargetUpstreamNodeID				PMLc.TargetUpstreamNodeId	DINT
ChangeNodeServicedUpstream				PMLc.ChangeNodeServicedUpstream	DINT
ChangeNodeServicedDownstream				PMLc.ChangeNodeServicedDownstream	DINT

Status structure PMLs

			Tag Name	Data Type
PMLs			PMLs	ST_PMLs
	CommandRejected		PMLs.CommandRejected	BOOL
	UnitModeCurrent		PMLs.UnitModeCurrent	DINT
	UnitModeRequested		PMLs.UnitModerequested	DINT
	UnitModeChangeInProcess		PMLs.UnitModeChangeInProcess	BOOL
	ProcModeCurrent		PMLs.ProcModeCurrent	DINT
	ProcModeRequested		PMLs.ProcModeRequested	DINT
	ProcModeChangeInProcess		PMLs.ProcModeChangeInProcess	BOOL
	StateCurrent		PMLs.StateCurrent	DINT
	StateRequested		PMLs.StateRequested	DINT
	StateChangeInProcess		PMLs.StateChangeInProcess	BOOL
	StateChangeProgress		StatusStateChangeProgress	DINT
	StateLastCompleted		PMLs.StateLastCompleted	DINT
	SeqNumber		PMLs.SeqNumber	DINT
	CurMachSpd		PMLs.CurMachSpd	DINT
	MatReady		PMLs.MatReady	ST_Material
		RawMaterial	PMLs.MatReady.RawMaterial	BOOL
		CO2	PMLs.MatReady.CO2	BOOL
		Container	PMLs.MatReady.Container	BOOL
		Lubrication	PMLs.MatReady.Lubrication	BOOL
		Water	PMLs.MatReady.Water	BOOL
		ContainerClosures	PMLs.MatReady.ContainerClosures	BOOL
		Unused0	PMLs.MatReady.Unused0	BOOL
		Unused1	PMLs.MatReady.Unused1	BOOL
		Unused2	PMLs.MatReady.Unused2	BOOL
		Unused3	PMLs.MatReady.Unused3	BOOL
		Unused4	PMLs.MatReady.Unused4	BOOL
		Unused5	PMLs.MatReady.Unused5	BOOL
		Unused6	PMLs.MatReady.Unused6	BOOL
		Unused7	PMLs.MatReady.Unused7	BOOL
		Unused8	PMLs.MatReady.Unused8	BOOL
		Unused9	PMLs.MatReady.Unused9	BOOL
	MatLow		PMLs.MatLow	ST_Material
		RawMaterial	PMLs.MatLow.RawMaterial	BOOL
		CO2	PMLs.MatLow.CO2	BOOL
		Container	PMLs.MatLow.Container	BOOL
		Lubrication	PMLs.MatLow.Lubrication	BOOL
		Water	PMLs.MatLow.Water	BOOL
		ContainerClosures	PMLs.MatLow.ContainerClosures	BOOL
		Unused0	PMLs.MatLow.Unused0	BOOL
		Unused1	PMLs.MatLow.Unused1	BOOL
		Unused2	PMLs.MatLow.Unused2	BOOL
		Unused3	PMLs.MatLow.Unused3	BOOL
		Unused4	PMLs.MatLow.Unused4	BOOL
		Unused5	PMLs.MatLow.Unused5	BOOL
		Unused6	PMLs.MatLow.Unused6	BOOL
		Unused7	PMLs.MatLow.Unused7	BOOL
		Unused8	PMLs.MatLow.Unused8	BOOL
		Unused9	PMLs.MatLow.Unused9	BOOL
	MachDesignSpeed		PMLs.MachDesignSpeed	REAL
	MachCycle		PMLs.MachCycle	DINT
	ProdRatio		PMLs.ProdRatio	DINT
	Dirty		PMLs.Dirty	BOOL
	Clean		PMLs.Clean	BOOL
	TimeToDirty		PMLs.TimeToDirty	DINT

EquipmentAllocatedToUnitModelID				PMLs.EquipmentAllocatedToUnitModelID	DINT
MachineReusableForUnitModelID				PMLs.MachineReusableForUnitModelID	DINT
MachineReusableTimeLeft				PMLs.MachineReusableTimeLeft	DINT
MachineStoringProductID				PMLs.MachineStoringProductID	DINT
MachineTransferringProductID				PMLs.MachineTransferringProductID	DINT
Node[#]				PMLs.Node[#]	ST_Node
	Number			PMLs.Node[#].Number	DINT
	ControlCmdNumber			PMLs.Node[#].ControlCmdNumber	DINT
	CmdValue			PMLs.Node[#].CmdValue	DINT
	Parameter[#]			PMLs.Node[#].Parameter[#]	ST_Descriptor
		Id		PMLs.Node[#].Parameter[#].Id	DINT
		Name		PMLs.Node[#].Parameter[#].Name	STRING
		Unit		PMLs.Node[#].Parameter[#].Unit	STRING
		Value		PMLs.Node[#].Parameter[#].Value	REAL
ProcessVariables[#]				PMLs.ProcessVariables[#]	ST_Descriptor
	Id			PMLs.ProcessVariables[#].Id	DINT
	Name			PMLs.ProcessVariables[#].Name	STRING
	Unit			PMLs.ProcessVariables[#].Unit	STRING
	Value			PMLs.ProcessVariables[#].Value	REAL
Product[#]				PMLs.Product[#]	ST_Product
	ProductId			PMLs.Product[#].ProductId	DINT
	ProcessVariables[#]			PMLs.Product[#].ProcessVariables[#]	ST_Descriptor
		Id		PMLs.Product[#].ProcessVariables[#].Id	DINT
		Name		PMLs.Product[#].ProcessVariables[#].Name	STRING
		Unit		PMLs.Product[#].ProcessVariables[#].Unit	STRING
		Value		PMLs.Product[#].ProcessVariables[#].Value	REAL
	Ingredients[#]			PMLs.Product[#].Ingredients[#]	ST_Ingedient
		IngredientId		PMLs.Product[#].Ingredients[#].IngredientId	DINT
		Parameter[#]		PMLs.Product[#].Ingredients[#].Parameter[#]	ST_Descriptor
			Id	PMLs.Product[#].Ingredients[#].Parameter[#].Id	DINT
			Name	PMLs.Product[#].Ingredients[#].Parameter[#].Name	STRING
			Unit	PMLs.Product[#].Ingredients[#].Parameter[#].Unit	STRING
			Value	PMLs.Product[#].Ingredients[#].Parameter[#].Value	REAL
Limits[#]				PMLs.Limits[#]	ST_Descriptor
	Id			PMLs.Limits[#].Id	DINT
	Name			PMLs.Limits[#].Name	STRING
	Unit			PMLs.Limits[#].Unit	STRING
	Value			PMLs.Limits[#].Value	REAL

Admin structure PMLa

				Tag Name	Data Type
PMLa				Admin	ST_PMLa
	Alarm[#]			PMLa.Alarm[#]	ST_Alarm
		Id		PMLa.Alarm[#].Id	DINT
		Value		PMLa.Alarm[#].Value	DINT
		Message		PMLa.Alarm[#].Message	STRING
		TimeEvent		PMLa.Alarm[#].TimeEvent	ST_TimeStamp
			Year	PMLa.Alarm[#].TimeEvent.Year	DINT
			Month	PMLa.Alarm[#].TimeEvent.Month	DINT
			Day	PMLa.Alarm[#].TimeEvent.Day	DINT
			Hour	PMLa.Alarm[#].TimeEvent.Hour	DINT
			Minute	PMLa.Alarm[#].TimeEvent.Minute	DINT
			Second	PMLa.Alarm[#].TimeEvent.Second	DINT
			mSec	PMLa.Alarm[#].TimeEvent.mSec	DINT
		TimeAck		PMLa.Alarm[#].TimeAck	ST_TimeStamp
			Year	PMLa.Alarm[#].TimeAck.Year	DINT
			Month	PMLa.Alarm[#].TimeAck.Month	DINT
			Day	PMLa.Alarm[#].TimeAck.Day	DINT
			Hour	PMLa.Alarm[#].TimeAck.Hour	DINT
			Minute	PMLa.Alarm[#].TimeAck.Minute	DINT
			Second	PMLa.Alarm[#].TimeAck.Second	DINT
			mSec	PMLa.Alarm[#].TimeAck.mSec	DINT
	ModeCurrentTime[#]			PMLa.Alarm[#].ModeCurrentTime[#]	DINT
	ModeCummulativeTime[#]			PMLa.Alarm[#].ModeCummulativeTime[#]	DINT
	StateCurrentTime[#,#]			PMLa.Alarm[#].StateCurrentTime[#,#]	DINT
	StateCummulativeTime[#,#]			PMLa.Alarm[#].StateCummulativeTime[#,#]	DINT
	ProdProcessed			PMLa.Alarm[#].ProdProcessed	DINT
	DefectProd			PMLa.Alarm[#].DefectProd	DINT
	ReworkProd			PMLa.Alarm[#].ReworkProd	DINT
	ResetTimersCounters			PMLa.Alarm[#].ResetTimersCounters	DINT
	UpstreamMessage			PMLa.Alarm[#].UpstreamMessage	DINT
	DownstreamMessage			PMLa.Alarm[#].DownStreamMessage	DINT
	CurrentDownstreamNodeID[#]			PMLa.Alarm[#].CurrentDownstreamNodeId[#]	DINT
	CurrentUpstreamNodeID[#]			PMLa.Alarm[#].CurrentUpstreamNodeId[#]	DINT

4.4 DataTypes

4.4.1 Alarm

4.4.1.1 E_AlarmID

The data type E_AlarmId defines the error cause.

```
TYPE E_AlarmID : (
    eAID_Undefined                          := 0,

    (*** 1..32 Machine internal reason - safeties - OMAC defined ***)
    eAID_EStop_Pushed                       := 1,
    eAID_PerimeterProtectionFault,
    eAID_MainsTurnedOff,
    eAID_SafetyGateOrGuardDoorOpen,
    (* 5..32 reserved for future OMAC safety codes *)

    (*** 33..64 Machine internal reason - operator actions - OMAC defined *)
    eAID_CycleStopButtonPushed              := 33,
```

```
    eAID_StartButtonPushed,
    eAID_ResetButtonPushed,
    eAID_JogModeSelected,
    eAID_AutomaticModeSelected,
    eAID_ManualModeSelected,
    eAID_SemiAutomaticModeSelected,
    (* 40..64 reserved for future OMAC defined operator action codes *)

    (*** 65..256 Machine internal reason - internal machine faults - product related - OMAC defined
*)
    eAID_MaterialJam                        := 65,
    (* 66..256 reserved for future OMAC defined internal material related codes *)

    (*** 257..512 Machine internal reason - internal machine faults - machine related - OMAC defined
*)
    eAID_MachineJam                         := 257,
    eAID_ElectricalOverload,
    eAID_MechanicalOverload,
    eAID_DriveFault,
    eAID_DriveFailure,
    eAID_ServoAxisFault,
    eAID_ServoAxisFailure,
    eAID_CommunicationError,
    eAID_PlcErrorCode,
    eAID_Vacuum,
    eAID_AirPressure,
    eAID_Voltage,
    eAID_Temperature,
    eAID_HydraulicPressure,
    eAID_HydraulicLevel,
    eAID_HydraulicTemperature,
    (* 273..512 reserved for future OMAC defined internal machine related codes *)

    (*** 513..999 Machine internal reason - general information - OMAC defined *)
    eAID_CounterPresetReached               := 513,
    eAID_ProductSelected,
    eAID_LocalSlowSpeedRequested,
    eAID_LocalMediumSpeedRequested,
    eAID_LocalHighSpeedRequested,
    eAID_LocalSurgeSpeedRequested,
    eAID_RemoteSpeedRequested,
    eAID_DriveWarning,
    eAID_ServoWarning,
    (* 522..998 reserved for future OMAC defined general information related codes *)
    eAID_CatchAllUndefinedInternalReason    := 999,

    (*** 1000..1999 Machine internal reason - vendor defined *)
    (* 1000..1999 vendor defined area for machine internal items *)

    (*** 2000..2499 Machine upstream process reason - OMAC defined *)
    eAID_InfeedNotTurnedOn                  := 2000,
    eAID_InfeedOverload,
    eAID_LowPrimeMaterial,
    eAID_HighPrimeMaterial,
    (* 2004..2498 reserved for future OMAC defined upstream reason *)
    eAID_CatchAllUndefinedUpstreamReason    := 2499,

    (*** 2500..2999 Machine upstream process reason - vendor defined *)
    (* 2500..2999 vendor defined area for upstream items *)

    (*** 3000..3499 Machine downstream process reason - OMAC defined *)
    eAID_DischargeNotTurnedOn               := 3000,
    eAID_DischargeOverload,
    eAID_DischargeBlockedReason,
    eAID_DischargeCycleStopReason,
    eAID_DischargeImmediateStopReason,
    (* 3004..3498 reserved for future OMAC defined downstream reason *)
    eAID_CatchAllUndefinedDownstreamReason  := 3499,

    (*** 3500..3999 Machine upstream process reason - vendor defined *)
    (* 3500..3999 vendor defined area for downstream items *)

    (*** 4000..4499 out of service - OMAC defined *)
    eAID_LineNotScheduled                   := 4000,
    eAID_PlannedMaintenance,
    eAID_MealsAndRest,
    eAID_Meetings,
    eAID_Training,
    eAID_NoPackagingMaterials,
    eAID_RemoteStopRequested,
```

```
    eAID_MachineNotSelected,
    eAID_Changeover,
    eAID_Lubrication,
    eAID_ProductCountPresetReached,
    eAID_SetupSelected,
    eAID_NoIncommingProduct,
    eAID_WaitingForElectrialService,
    eAID_WaitingForMechanicalService
    (* 4012..4499 reserved for future OMAC defined downstream reason *)

    (*** 4500..4999 out of service - vendor defined *)
    (* 4500..4999 vendor defined area for out of service items *)
);
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.1.2 ST_Alarm

This is the collection tags needed to describe alarm events.

```
TYPE ST_Alarm :
STRUCT
    Id                  : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Value               : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Message             : STRING;(* ~ (OPC : 1: enabled for OPC ) *)
    TimeEvent           : ST_TimeStamp;(* ~ (OPC : 1: enabled for OPC ) *)
    TimeAck             : ST_TimeStamp;(* ~ (OPC : 1: enabled for OPC ) *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.1.3 ST_TimeStamp

This structure is used to store the time and date of an event or the acknowledge of an event.

```
TYPE ST_TimeStamp :
STRUCT
    Year                : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Month               : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Day                 : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Hour                : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Minute              : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Second              : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    mSec                : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.2 Common

4.4.2.1 E_CntrlCmd

Enumeration of possible commands for the machine state

```
TYPE E_CntrlCmd :
(
    eCntrlCmd_UNDEFINED,
    eCntrlCmd_RESET,
    eCntrlCmd_START,
    eCntrlCmd_STOP,
    eCntrlCmd_HOLD,
    eCntrlCmd_UNHOLD,
    eCntrlCmd_SUSPEND,
    eCntrlCmd_UNSUSPEND,
    eCntrlCmd_ABORT,
    eCntrlCmd_CLEAR,
    eCntrlCmd_COMPLETE,
);
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.2.2 E_CurMachSpd

Enumeration to define different machine speeds

```
TYPE E_CurMachSpd :
(
    eCurMachSpd_UNDEFINED,
    eCurMachSpd_JOG,
    eCurMachSpd_PRIME,
    eCurMachSpd_PRELUBE,
    eCurMachSpd_MAINTENANCE,
    eCurMachSpd_SLOW,
    eCurMachSpd_MEDIUM,
    eCurMachSpd_HIGH,
    eCurMachSpd_SURGE,
    eCurMachSpd_TRACKING,
    eCurMachSpd_ANALOG_CTRL_ONLY := 99
);
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.2.3 ST_Descriptor

This is a collection of tags that are used to describe parameters in the machine unit.

```
TYPE ST_Descriptor :
STRUCT
    Id                  : DINT;   (* ~ (OPC : 1: enabled for OPC ) *)
    Name                : STRING; (* ~ (OPC : 1: enabled for OPC ) *)
    Unit                : STRING; (* ~ (OPC : 1: enabled for OPC ) *)
    Value               : REAL;   (* ~ (OPC : 1: enabled for OPC ) *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.2.4 ST_Ingedient

This is a collection of tags used to describe the raw materials that are needed for the product.

```
TYPE ST_Ingredient :
STRUCT
    IngredientId        : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Parameter           : ARRAY [1..iMAX_INGREDIENT_PARAMS] OF ST_Descriptor;(* ~ (OPC : 1: enabled
for OPC ) *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.2.5 ST_Material

This is a collection of tags that are used to describe different materials in the machine unit.

```
TYPE ST_Materials :
STRUCT
    RawMaterial         : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    CO2                 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Container           : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Lubrication         : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Water               : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    ContainerClosures   : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)

    (* 10 more unused *)
    Unused0             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused1             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused2             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused3             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused4             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused5             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused6             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused7             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused8             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused9             : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.2.6 ST_Node

This is a collection of tags that are used to describe communication command values between machines using the PackTag structure.

```
TYPE ST_Node :
STRUCT
    Number              : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ControlCmdNumber    : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    CmdValue            : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Parameter           : ARRAY [1..iMAX_NODE_PARAMS] OF ST_Descriptor; (* ~ (OPC : 1: enabled for
OPC ) *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.2.7 ST_Product

This is a collection of tags used to describe the product that the machine is making.

```
TYPE ST_Product :
STRUCT
    ProductId           : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcessVariables    : ARRAY [1..iMAX_PROD_PROCESS_VARS] OF ST_Descriptor; (* ~ (OPC : 1: enabled
for OPC ) *)
    Ingredients         : ARRAY [1.. iMAX_INGREDIENTS] OF ST_Ingredient; (* ~ (OPC : 1: enabled for
OPC ) *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.3 ST_PMLa

This is the collection of all administration tags in the PackTag structure.

```
TYPE ST_PMLa :
STRUCT
    Alarm                     : ARRAY [1..iMAX_ALARMS] OF ST_Alarm;(* ~ (OPC : 1: enabled for OPC )
*)
    ModeCurrentTime           : ARRAY [1..iMAX_CURRENT_MODE] OF DINT;(* ~ (OPC : 1: enabled for
OPC ) *)
    ModeCummulativeTime       : ARRAY [1..iMAX_CUMMULATIVE_MODE] OF DINT;(* ~ (OPC : 1: enabled for
OPC ) *)
    StateCurrentTime          : ARRAY [1..iMAX_CURRENT_MODE, 0..iMAX_CURRENT_STATE] OF DINT;(* ~
(OPC : 1: enabled for OPC ) *)
    StateCummulativeTime      : ARRAY [1..iMAX_CURRENT_MODE, 0..iMAX_CURRENT_STATE] OF DINT;(* ~
(OPC : 1: enabled for OPC ) *)

    ProdProcessed             : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    DefectProd                : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ReworkProd                : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ResetTimersCounters       : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    UpstreamMessage           : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    DownstreamMessage         : DINT;(* ~ (OPC : 1: enabled for OPC ) *)

    CurrentDownstreamNodeID   : ARRAY [1..iMAX_CURR_NODE_IDS] OF DINT;(* ~ (OPC : 1: enabled for
OPC ) *)
    CurrentUpstreamNodeID     : ARRAY [1..iMAX_CURR_NODE_IDS] OF DINT;(* ~ (OPC : 1: enabled for
OPC ) *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.4 ST_PMLc

This is the collection of all command tags in the PackTag structure.

```
TYPE ST_PMLc :
STRUCT
    UnitMode                      : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    UnitModeChangeRequest         : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcMode                      : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcModeChangeRequest         : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    CurMachSpeed                  : DINT;(* ~ (OPC : 1: enabled for OPC ) *)

    MatReady                      : ST_Materials;(* ~ (OPC : 1: enabled for OPC ) *)
    MatLow                        : ST_Materials;(* ~ (OPC : 1: enabled for OPC ) *)
```

```
    State                           : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    StateChangeRequest              : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)

    CntrlCmd                        : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Node                            : ARRAY [1..iMAX_NODES] OF ST_Node;(* ~ (OPC : 1: enabled for
OPC ) *)
    ProcessVariables                : ARRAY [1..iMAX_PROCESS_VARS] OF ST_Descriptor;(* ~ (OPC : 1:
enabled for OPC ) *)
    Product                         : ARRAY [1..iMAX_PRODUCTS] OF ST_Product;(* ~ (OPC : 1: enabled
for OPC ) *)
    Limits                          : ARRAY [1..iMAX_LIMITS] OF ST_Descriptor;(* ~ (OPC : 1: enabled
for OPC ) *)

    TargetDownstreamNodeID          : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    TargetUpstreamNodeID            : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ChangeNodeServicedUpstream      : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ChangeNodeServicedDownstream    : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.4.5 ST_PMLs

This is the collection of all status tags in the PackTag structure.

```
TYPE ST_PMLs :
STRUCT
    CommandRejected                 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    UnitModeCurrent                 : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    UnitModeRequested               : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    UnitModeChangeInProcess         : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcModeCurrent                 : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcModeRequested               : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcModeChangeInProcess         : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    StateCurrent                    : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    StateRequested                  : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    StateChangeInProcess            : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    StateChangeProgress             : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    StateLastCompleted              : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    SeqNumber                       : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    CurMachSpd                      : DINT;(* ~ (OPC : 1: enabled for OPC ) *)

    MatReady                        : ST_Materials;(* ~ (OPC : 1: enabled for OPC ) *)
    MatLow                          : ST_Materials;(* ~ (OPC : 1: enabled for OPC ) *)

    MachDesignSpeed                 : REAL;(* ~ (OPC : 1: enabled for OPC ) *)
    MachCycle                       : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProdRatio                       : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Dirty                           : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Clean                           : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    TimeToDirty                     : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    EquipmentAllocatedToUnitModelID: DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    MachineReusableForUnitModelID   : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    MachineReusableTimeLeft         : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    MachineStoringProductID         : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    MachineTransferringProductID    : DINT;(* ~ (OPC : 1: enabled for OPC ) *)

    Node                            : ARRAY [1..iMAX_NODES] OF ST_Node;(* ~ (OPC : 1: enabled for
OPC ) *)
    ProcessVariables                : ARRAY [1..iMAX_PROCESS_VARS] OF ST_Descriptor;(* ~ (OPC : 1:
enabled for OPC ) *)
    Product                         : ARRAY [1..iMAX_PRODUCTS] OF ST_Product;(* ~ (OPC : 1: enabled
for OPC ) *)
    Limits                          : ARRAY [1..iMAX_LIMITS] OF ST_Descriptor;(* ~ (OPC : 1: enabled
for OPC ) *)

END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.5 Parameter List

Parameter for the Packaging Machine Tag Structures. While inserting the library they can be customized for the current project.

```
(* PMLc / PMLs *)
iMAX_NODE_PARAMS                : INT := 10
iMAX_NODES                      : INT := 10
iMAX_PROCESS_VARS               : INT := 10
iMAX_PROD_PROCESS_VARS          : INT := 10
iMAX_INGREDIENT_PARAMS          : INT := 10
iMAX_INGREDIENT                 : INT := 10
iMAX_LIMITS                     : INT := 10
iMAX_PRODUCTS                   : INT := 10

(* PMLa *)
iMAX_ALARMS                     : INT := 10
iMAX_CURR_NODE_IDS              : INT := 10
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

4.6 Global Constants

Constants for the Packaging Machine Tag Structures, they can’t be changed.

```
(* PMLa *)
iMAX_CURRENT_MODE               : INT := 10
iMAX_CUMMULATIVE_MODE           : INT := 10
iMAX_CURRENT_STATE              : INT := 17
iMAX_CUMMULATIVE_STATE          : INT := 17
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1, build 4018 onwards	PC (i386)	Tc3_PackML

5 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

More Information:
www.beckhoff.com/te1000

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com