

BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc3_PackML_V2

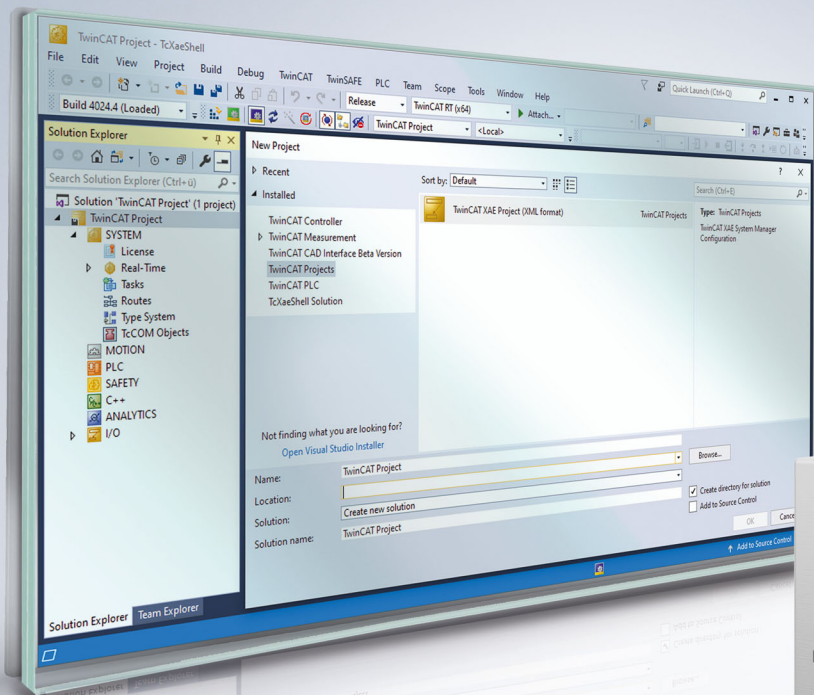


Table of contents

1	Foreword	5
1.1	Notes on the documentation	5
1.2	For your safety	5
1.3	Notes on information security.....	7
2	Packaging Machine State	8
2.1	Interfaces	8
2.1.1	I_UnitState	8
2.1.2	I_UnitStateWaiting	9
2.1.3	I_UnitStateActing	9
2.2	Data types	10
2.2.1	E_PMLCommand.....	10
2.2.2	E_PMLState	10
2.2.3	E_PMLProtectedUnitMode.....	11
2.2.4	ST_PMLSubUnitInfoRef.....	12
2.2.5	ST_PMLSubUnitInfo	12
2.2.6	ST_PMLStateMachineOptions.....	13
2.2.7	ST_AdminTimeOptions	13
2.3	Function Blocks.....	13
2.3.1	Packaging Machine State	13
2.3.2	General	23
2.3.3	Conversion	29
3	Packaging Machine Tags	33
3.1	Introduction	33
3.2	Tag Types	33
3.3	Tag Details	33
3.4	Data types	39
3.4.1	Alarm.....	39
3.4.2	Common.....	40
3.4.3	ST_PMLa	42
3.4.4	ST_PMLc	42
3.4.5	ST_PMLs	42
3.5	Global parameters.....	43
3.6	Global Constants.....	43
4	Example Tc3_PackML_V2	44
5	Support and Service	45

1 Foreword

1.1 Notes on the documentation

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

For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.

The qualified personnel is obliged to always use the currently valid documentation.

The responsible staff must ensure that the application or use of the products described satisfies all 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 notice.

No claims to modify 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 and licensed trademarks of Beckhoff Automation GmbH.

If third parties make use of designations or trademarks used in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

Patents

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

EtherCAT®

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The distribution and reproduction of this document as well as the use and communication of its contents without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

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 Packaging Machine State

The “Packaging Machine State” function blocks have a common interface to the existing “PackML Machine State Model” versions.

It is expected that application-specific logic such as state transitions are programmed in external function blocks and that the “Packaging Machine State” function block takes over the central logic of the state machine and the state display. For that reason there is a recommendation for this function block with regard to how it can be combined with other logic.

The state transition in a machine application is always application-specific. For that reason it is best to design linked “State” function blocks with PackML State Machine V3 in order to simplify the standardization. The “State” function blocks acquire application-specific signals and represent the transition logic to adjacent states (see PackML state model). The “State” function blocks supply feedback to PS_PackML_State_Machine_V3, as a result of which a standard state machine and a state message are possible. The “State” function blocks contain the machine execution code and the application-specific transition logic.

The “State” function blocks are listed below and are programmed by the application programmer in such a way that the integrity and functionality of the PackML State Machine are retained.

Names of the “PackML State Machine V3” function blocks:

- 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 include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.1 Interfaces

2.1.1 I_UnitState

This interface can be implemented in the unit function blocks of the application and makes available all methods of the Packaging State Model, which can then be filled as required with application code.

These methods are:

M_Aborted

M_Aborting
M_Clearing
M_Complete
M_Completing
M_Execute
M_Held
M_Holding
M_Idle
M_Resetting
M_Starting
M_StateComplete
M_Stopped
M_Stopping
M_Suspended
M_Suspending
M_Undefined
M_Unholding
M_Unsuspending

2.1.2 I_UnitStateWaiting

This interface can be implemented in the unit function blocks of the application and only provides the "Waiting" methods of the Packaging State Model, which can then be filled with application code as required.

These methods are:

M_Aborted
M_Complete
M_Held
M_Idle
M_Stopped
M_Suspended
M_Undefined

2.1.3 I_UnitStateActing

This interface can be implemented in the unit function blocks of the application and only provides the "Acting" methods of the Packaging State Model, which can then be filled with application code as required.

These methods are:

M_Aborting
M_Clearing
M_Completing

M_Execute
 M_Holding
 M_Resetting
 M_Starting
 M_StateComplete
 M_Stopping
 M_Suspending
 M_Unholding
 M_Unsuspending

2.2 Data types

2.2.1 E_PMLCommand

E_PMLCommand

```

TYPE E_PMLCommand :
(
  (* states according to PackTags v3.0 *)
  ePMLCommand_Undefined := 0,
  ePMLCommand_Reset    := 1,
  ePMLCommand_Start    := 2,
  ePMLCommand_Stop     := 3,
  ePMLCommand_Hold     := 4,
  ePMLCommand_Unhold   := 5,
  ePMLCommand_Suspend  := 6,
  ePMLCommand_Unsuspend := 7,
  ePMLCommand_Abort    := 8,
  ePMLCommand_Clear    := 9
);
END_TYPE
  
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.2.2 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 include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.2.3 E_PMLProtectedUnitMode

E_PMLProtectedUnitMode

```
TYPE E_PMLProtectedUnitMode :
(
  ePMLUnitMode_Invalid := 0,
  ePMLUnitMode_Production := 1,
  ePMLUnitMode_Maintenance := 2,
  ePMLUnitMode_Manual := 3
);
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.2.4 ST_PMLSubUnitInfoRef

ST_PMLSubUnitInfoRef

```

TYPE ST_PMLSubUnitInfoRef :
STRUCT
  pArray      : POINTER TO ST_PMLSubUnitInfo;
  nArraySize  : UDINT;
  nNoOfSubUnits : UDINT;
END_STRUCT
END_TYPE

```

pArray	Address of a one-dimensional array of the type ST_PMLSubUnitInfo. Each array element contains the state of a subordinated machine part. Example: stSubUnitInfo : ARRAY[1..10] OF ST_PMLSubUnitInfo; pArray := ADR(stSubUnitInfo);
nArraySize	Memory size of the one-dimensional array, which can be determined with the SIZEOF function. Example: nArraySize := SIZEOF(stSubUnitInfo);
nNoOfSubUnits	Number of relevant subordinated machine parts.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.2.5 ST_PMLSubUnitInfo

ST_PMLSubUnitInfo

```

TYPE ST_PMLSubUnitInfo :
STRUCT
  bActive : BOOL;
  eState  : E_PMLState;
END_STRUCT
END_TYPE

```

bActive	Signals that this subordinated machine part is active and follows the state presets of the state machine.
eState	Enumeration that reflects the current state of the subordinated machine part.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.2.6 ST_PMLStateMachineOptions

ST_PMLStateMachineOptions

```

TYPE ST_PMLStateMachineOptions :
STRUCT
END_STRUCT
END_TYPE
    
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.2.7 ST_AdminTimeOptions

ST_AdminTimeOptions

```

TYPE ST_AdminTimeOptions :
STRUCT
    UseExternalTime          : BOOL;
    ExternalPackMLTime      : ARRAY [0..6] OF DINT;
END_STRUCT
END_TYPE
    
```

UseExternalTime	If this flag is set to TRUE, the time set at the input ExternalPackMLTime is used instead of the system time information.
ExternalPackMLTime	Externally set time

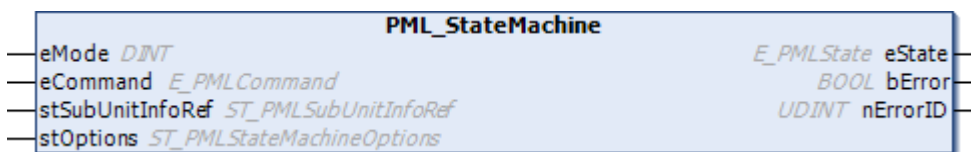
Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3 Function Blocks

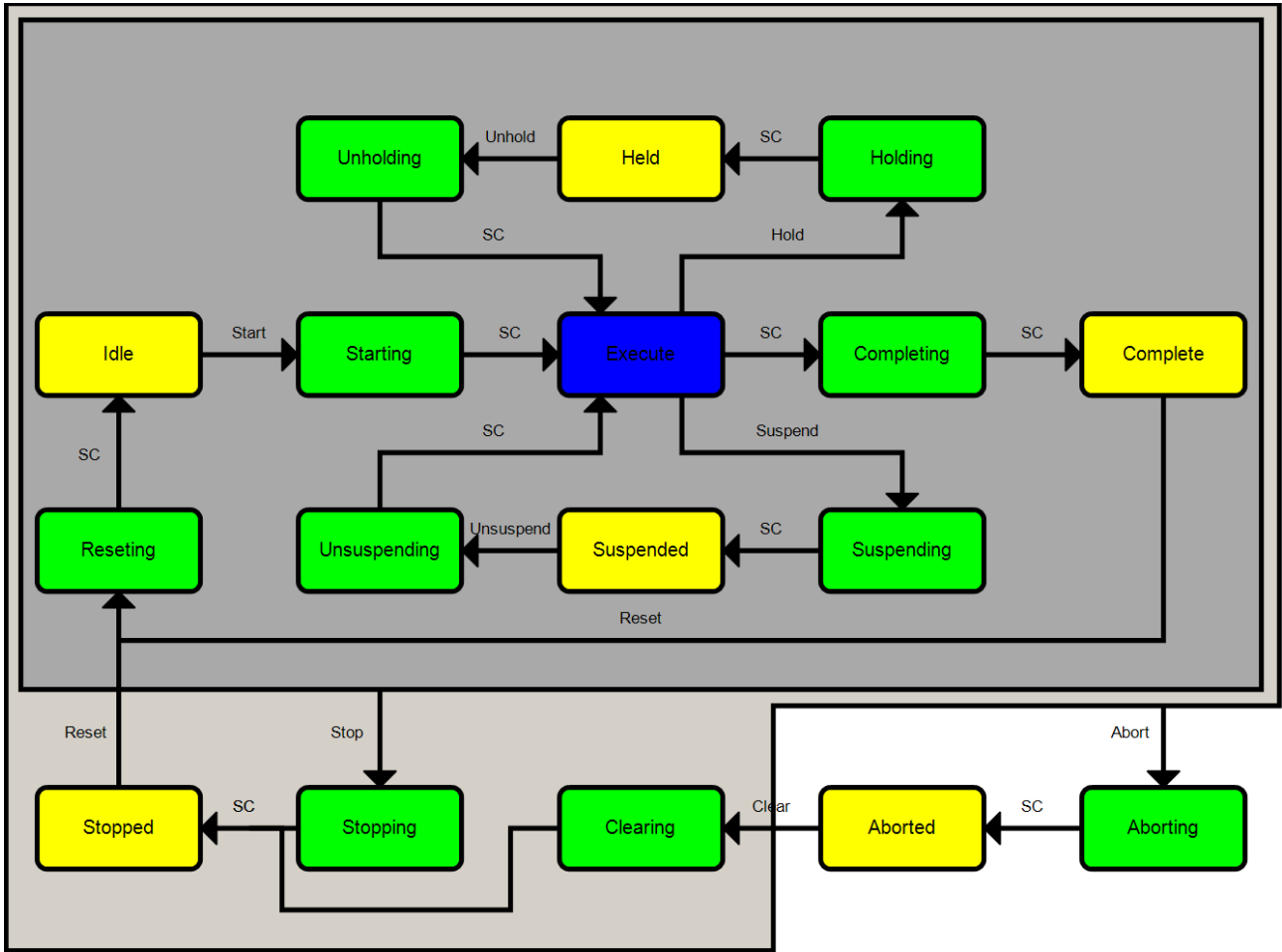
2.3.1 Packaging Machine State

2.3.1.1 PML_StateMachine

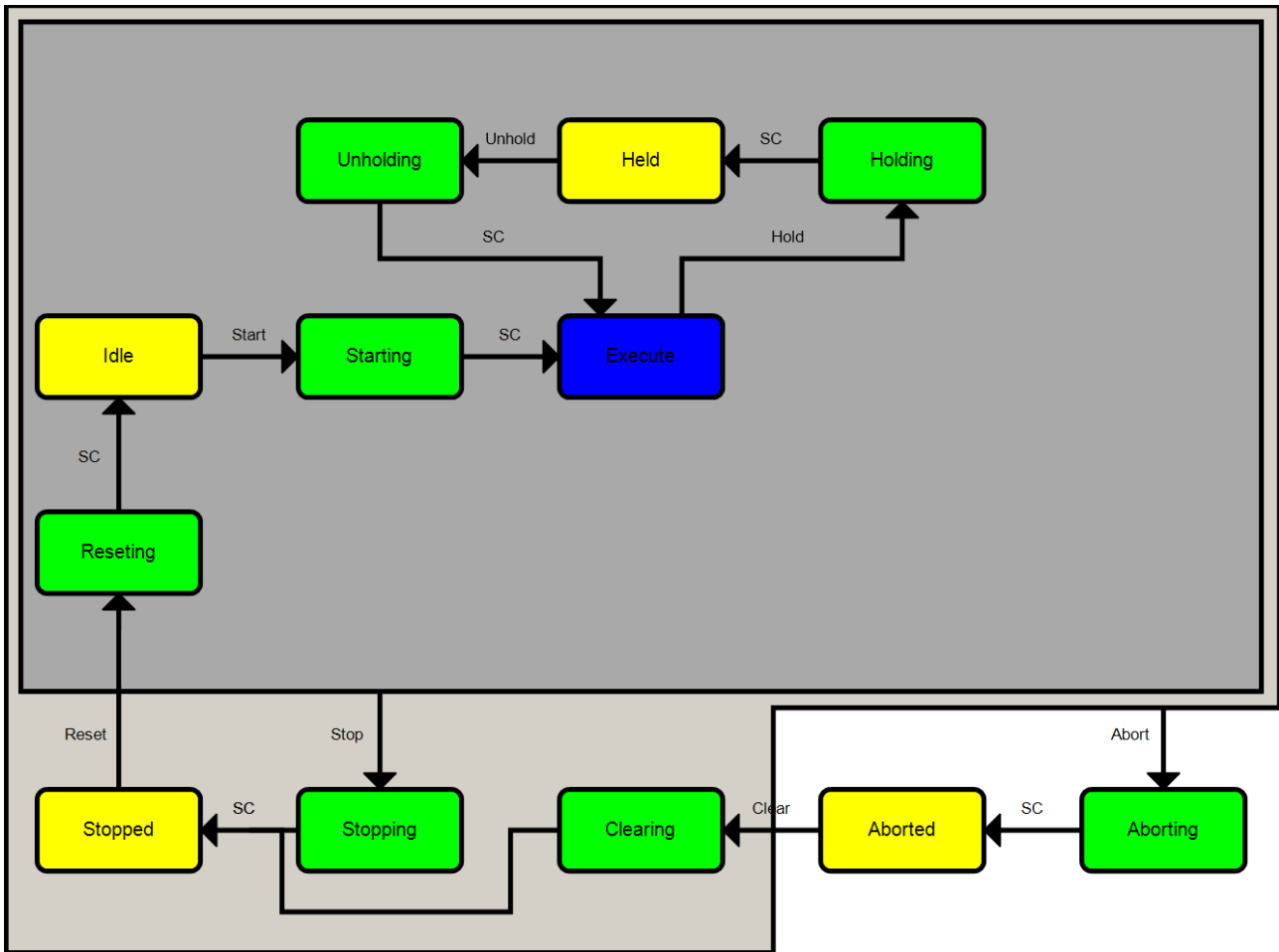


In the updated form the PML_StateMachine function block has a common interface with the PackML Machine State Model V3. It is assumed that application-specific logic, such as state transitions, is programmed in external function blocks and that the PML_StateMachine function block deals with the central logic of the state machine and the state representation. The Machine State Model has a different appearance due to the currently active UnitMode (eMode). Three basic modules are preconfigured for this (E_PMLProtectedUnitMode [▶ 11]).

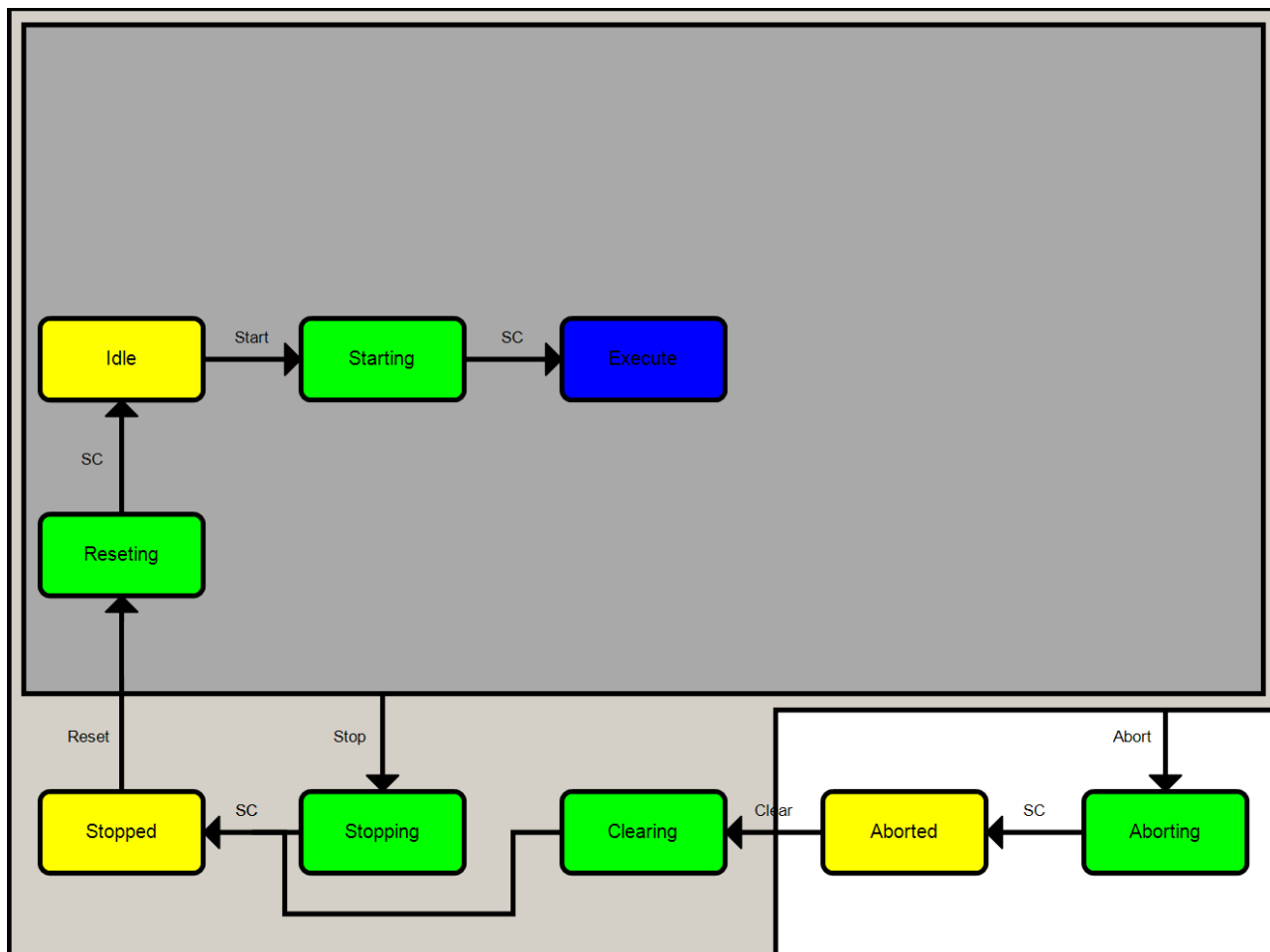
ePMLProtUnitMode_Production



ePMLProtUnitMode_Maintenance



ePMLProtUnitMode_Manual



Furthermore, other user-specific models can be created in a simple manner with the aid of the function block PML_UnitModeConfig [▶ 17] and are thus very flexible in use.

The logic for transitions, in particular between production, maintenance and manual mode, depends on the application. The states in which UnitMode changes are permissible for the basic models are described more precisely in the description of the function block PML_UnitModeManager [▶ 20].

Inputs

```

VAR_INPUT
    eMode          : DINT;
    eCommand       : E_PMLCommand;
    stSubUnitInfoRef : ST_PMLSubUnitInfoRef;
    stOptions      : ST_PMLStateMachineOptions;
END_VAR
    
```

Name	Type	Description
eMode	DINT	Current PML UnitMode
eCommand	E_PMLCommand	Enumeration [▶ 10] with the various PML commands of the function block.
stSubUnitInfoRef	ST_PMLSubUnitInfoRef	Structure [▶ 12] that points to an array of the current PML states of subordinated machine units
stOptions	ST_PMLStateMachineOptions	Not used at present

🔌 Outputs

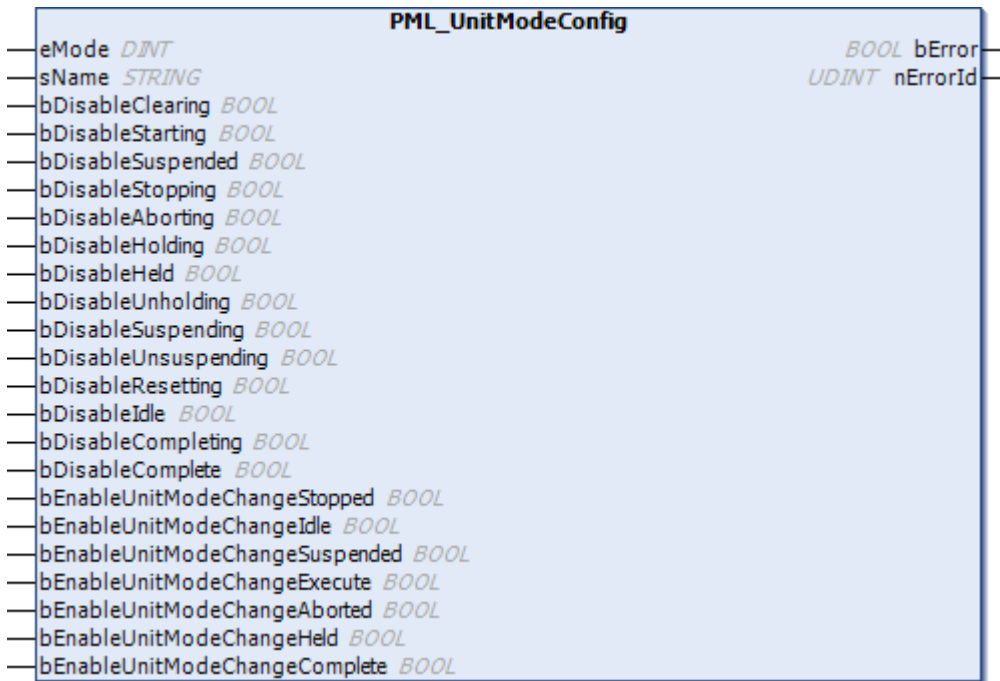
```
VAR_OUTPUT
  eState      : E_PMLState;
  bError      : BOOL;
  nErrorId    : UDINT;
END_VAR
```

Name	Type	Description
eState	E_PMLState	Enumeration [▶_10] that delivers the current PML state of the automatic state machine.
bError	BOOL	Becomes TRUE as soon as an error occurs.
nErrorId	UDINT	Supplies the error number when the bError output is set.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.1.2 PML_UnitModeConfig



Machines may have unit modes other than "Production", "Maintenance" and "Manual". This function block enables the user to configure further models (UnitModes).

The number of the new model, the existing states and the states in which a model change is possible can be freely defined.

🔌 Inputs

```
VAR_INPUT
  eMode      : DINT;
  sName      : STRING;
  bDisableClearing : BOOL;
  bDisableStarting  : BOOL;
  bDisableSuspended : BOOL;
  bDisableStopping  : BOOL;
  bDisableAborting  : BOOL;
  bDisableHolding   : BOOL;
  bDisableHeld      : BOOL;
```

```

bDisableUnholding      : BOOL;
bDisableSuspending     : BOOL;
bDisableUnsuspending  : BOOL;
bDisableResetting     : BOOL;
bDisableIdle          : BOOL;
bDisableCompleting    : BOOL;
bDisableComplete      : BOOL;
bEnableUnitModeChangeStopped : BOOL;
bEnableUnitModeChangeIdle : BOOL;
bEnableUnitModeChangeSuspended : BOOL;
bEnableUnitModeChangeExecute : BOOL;
bEnableUnitModeChangeAborted : BOOL;
bEnableUnitModeChangeHeld : BOOL;
bEnableUnitModeChangeComplete : BOOL;

```

```
END_VAR
```

Name	Type	Description
eMode	DINT	Number of the new PML UnitMode [4..31]
sName	STRING	Name of the new PML UnitMode
bDisableClearing	BOOL	Deactivates the "Clearing" PML state.
bDisableStarting	BOOL	Deactivates the "Starting" PML state.
bDisableSuspended	BOOL	Deactivates the "Suspended" PML state. The deactivation of the static state also causes the "Suspending" and "Unsuspending" PML states to be deactivated.
bDisableStopping	BOOL	Deactivates the "Stopping" PML state.
bDisableAborting	BOOL	Deactivates the "Aborting" PML state.
bDisableHolding	BOOL	Deactivates the "Holding" PML state.
bDisableHeld	BOOL	Deactivates the "Held" PML state. The deactivation of the static state also causes the "Holding" and "Unholding" PML states to be deactivated.
bDisableUnholding	BOOL	Deactivates the "Unholding" PML state.
bDisableSuspending	BOOL	Deactivates the "Suspending" PML state.
bDisableUnsuspending	BOOL	Deactivates the "Unsuspending" PML state.
bDisableResetting	BOOL	Deactivates the "Resetting" PML state.
bDisableIdle	BOOL	Deactivates the "Idle" PML state. The deactivation of the static state also causes the "Resetting" PML state to be deactivated.
bDisableCompleting	BOOL	Deactivates the "Completing" PML state.
bDisableComplete	BOOL	Deactivates the "Complete" PML state. The deactivation of the static state also causes the "Completing" PML state to be deactivated.
bEnableUnitModeChangeStopped	BOOL	Enables a mode change in the "Stopped" PML state.
bEnableUnitModeChangeIdle	BOOL	Enables a mode change in the "Idle" PML state.
bEnableUnitModeChangeSuspended	BOOL	Enables a mode change in the "Suspended" PML state.
bEnableUnitModeChangeExecute	BOOL	Enables a mode change in the "Execute" PML state.
bEnableUnitModeChangeAborted	BOOL	Enables a mode change in the "Aborted" PML state.
bEnableUnitModeChangeHeld	BOOL	Enables a mode change in the "Held" PML state.
bEnableUnitModeChangeComplete	BOOL	Enables a mode change in the "Complete" PML state.

 **Outputs**

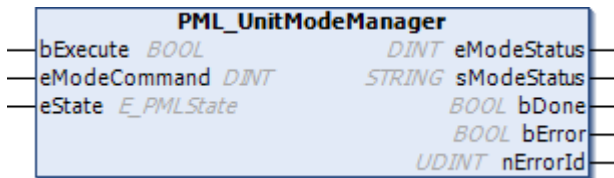
```
VAR_OUTPUT
  bError      : BOOL;
  nErrorID    : UDINT;
END_VAR
```

Name	Type	Description
bError	BOOL	Becomes TRUE as soon as an error occurs.
nErrorId	UDINT	Supplies the error number when the bError output is set.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.1.3 PML_UnitModeManager

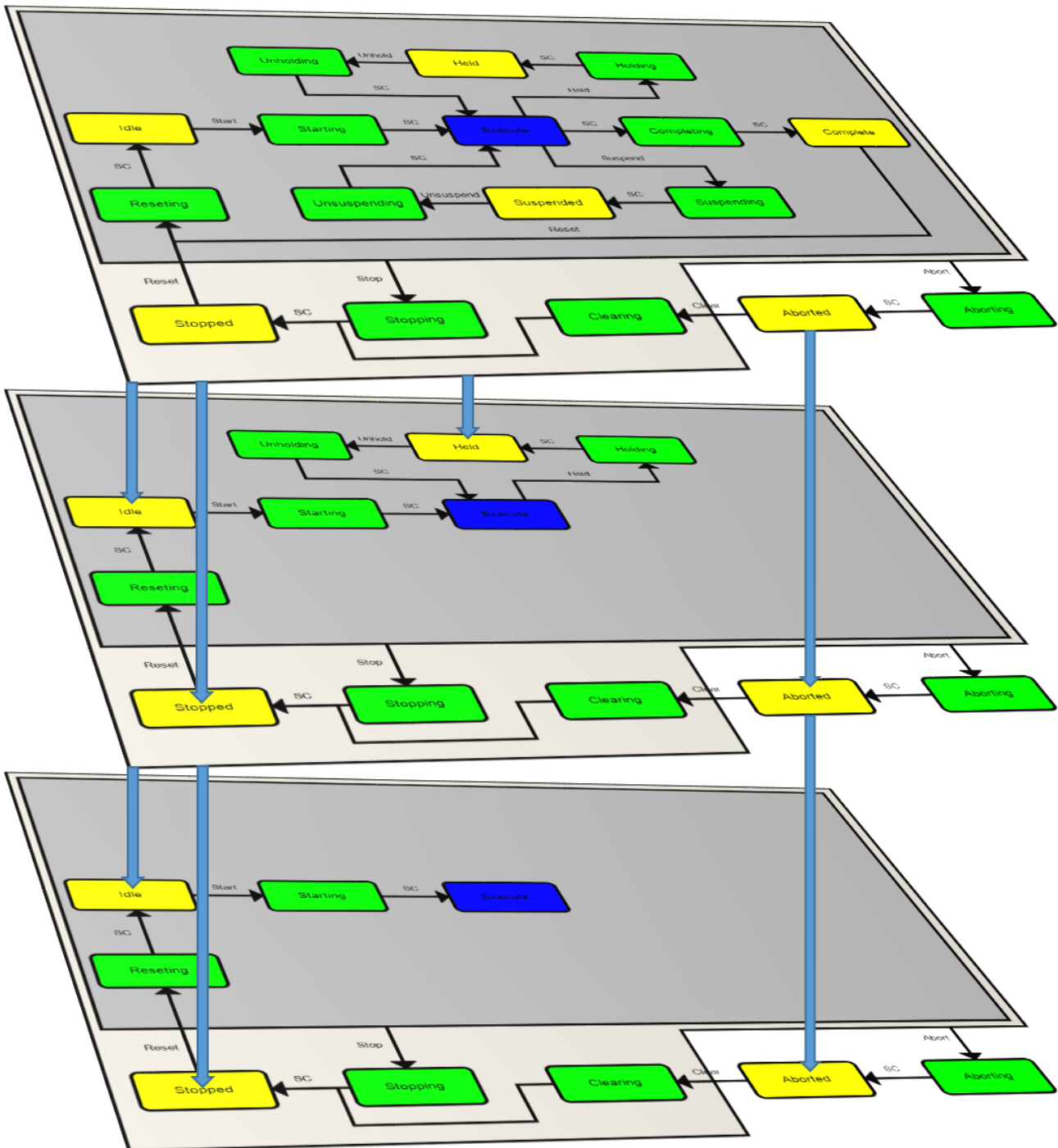


Machines have other system modes apart from “Production”. Each unit mode is defined by its own state model. A “Mode Manager” must be defined for transitions between the modes. The “Mode Manager” decides how and in which state a machine can change unit modes; i.e. built-in barriers prevent the machine from changing to unsuitable states. These barriers are permanently defined for the “Production”, “Maintenance” and “Manual” base modes, as the illustration below shows. This can be individually specified for other modes defined via the [PML_UnitModeConfig](#) [► 17] function block.

⚠ WARNING

Adhere to proper mode changes

The logic for transitions between the modes depends on the application, especially for transitions between “Manual” and “Production” mode. 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.



Inputs

```

VAR_INPUT
    bExecute      : BOOL;
    eModeCommand  : DINT;
    ePMLState     : E_PMLState;
END_VAR
    
```

Name	Type	Description
bExecute	BOOL	Mode change on rising edge
eModeCommand	DINT	Requested mode
ePMLState	E_PMLState	Enumeration [▶_10] that delivers the current PML state of the automatic state machine.

Outputs

```

VAR_OUTPUT
  eModeStatus      : DINT;
  sModeStatus      : STRING;
  bDone            : BOOL;
  bError           : BOOL;
  bErrorID         : UDINT;
END_VAR

```

Name	Type	Description
eModeStatus	DINT	Current PML UnitMode
sModeStatus	STRING	Name of the current PML UnitMode
bDone	BOOL	Becomes TRUE as soon as the mode change has been successfully carried out.
bError	BOOL	Becomes TRUE as soon as an error occurs.
nErrorID	UDINT	Supplies the error number when the bError output is set.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2 General

2.3.2.1 PML_AdminAlarm

This function block assists the user with the entry, acknowledgement and deletion of Alarms, Warnings and StopReasons of the Admin-PackTags. The function block provides different methods for this.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.1.1 M_SetAlarm

This method inserts an alarm in the Admin-Tags. Alarm[].Trigger is set to TRUE and the value from Admin.PlcDateTime is entered in Alarm[].DateTime. The other values are taken from the transferred alarm structure. The method returns TRUE if the alarm was entered successfully



So that a valid timestamp can be entered, the function block PML_AdminTime should be called cyclically in the program.

Syntax

```
METHOD M_SetAlarm : BOOL
VAR_IN_OUT
    stAdmin      : ST_PMLa;
END_VAR
VAR_INPUT
    stAlarm      : ST_Alarm;
END_VAR
```

Sample call:

```
AlarmInserted := fbAdminAlarm.M_SetAlarm(stAdmin := PackTags.Admin, stAlarm := Alarm);
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.1.2 M_AcknowledgeAlarm

This method acknowledges an alarm in the Admin-Tags. Alarm[].Trigger is set to FALSE and the value from Admin.PlcDateTime is entered in Alarm[].AckDateTime. The method returns TRUE if the alarm was found and acknowledged successfully. Acknowledging the alarm does not delete it. The alarm remains in the Alarm array until an M_ClearAlarm has been called, then it is moved to the AlarmHistory array. If the AlarmHistory array is already full of entries, the oldest entry is deleted as a result.



So that a valid timestamp can be entered, the function block PML_AdminTime should be called cyclically in the program.

Syntax

```
METHOD M_AcknowledgeAlarm : BOOL
VAR_IN_OUT
    stAdmin      : ST_PMLa;
END_VAR
VAR_INPUT
```

```

stAlarm      : ST_Alarm;
END_VAR

```

Sample call:

```
AlarmAcknowledged := fbAdminAlarm.M_AcknowledgeAlarm(stAdmin := PackTags.Admin, stAlarm := Alarm);
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.1.3 M_ClearAlarm

This method deletes an alarm from the Admin-Tags. Alarm[].Trigger is set to FALSE. The method returns TRUE if the alarm was deleted successfully. The alarm remains in the Alarm array until an M_AcknowledgeAlarm has been called, then it is moved to the AlarmHistory array. If the AlarmHistory array is already full of entries, the oldest entry is deleted as a result.



So that a valid timestamp can be entered, the function block PML_AdminTime should be called cyclically in the program.

Syntax

```

METHOD M_ClearAlarm : BOOL
VAR_IN_OUT
  stAdmin      : ST_PMLa;
END_VAR
VAR_INPUT
  stAlarm      : ST_Alarm;
END_VAR

```

Sample call:

```
AlarmCleared := fbAdminAlarm.M_ClearAlarm(stAdmin := PackTags.Admin, stAlarm := Alarm);
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.1.4 M_SetWarning

This method inserts a warning in the Admin-Tags. Warning[].Trigger is set to TRUE and the value from Admin.PlcDateTime is entered in Warning[].DateTime. The other values are taken from the transferred warning structure. The method returns TRUE if the warning was entered successfully. If the Warning array is already full of entries, the oldest entry is deleted as a result.



So that a valid timestamp can be entered, the function block PML_AdminTime should be called cyclically in the program.

Syntax

```

METHOD M_SetWarning : BOOL
VAR_IN_OUT
  stAdmin      : ST_PMLa;
END_VAR
VAR_INPUT
  stWarning    : ST_Alarm;
END_VAR

```


Sample call:

```
WarningInserted := fbAdminAlarm.M_SetWarning(stAdmin := PackTags.Admin, stWarning := Warning);
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.1.5 M_AcknowledgeWarning

This method acknowledges a warning in the Admin-Tags. Warning[].Trigger is set to FALSE and the value from Admin.PlcDateTime is entered in Warning[].AckDateTime. The method returns TRUE if the warning was found and acknowledged successfully. The warning remains in the Warning array until it is pushed out of the array by the next warning.



So that a valid timestamp can be entered, the function block PML_AdminTime should be called cyclically in the program.

Syntax

```
METHOD M_AcknowledgeWarning : BOOL
VAR_IN_OUT
    stAdmin          : ST_PMLa;
END_VAR
VAR_INPUT
    stWarning        : ST_Alarm;
END_VAR
```

Sample call:

```
WarningAcknowledged := fbAdminAlarm.M_AcknowledgeWarning(stAdmin := PackTags.Admin, stWarning := Warning);
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.1.6 M_ClearWarning

This method deletes a warning from the Admin-Tags. Warning[].Trigger is set to FALSE. The method returns TRUE if the warning was deleted successfully. The warning remains in the Warning array until it is pushed out of the array by the next warning.



So that a valid timestamp can be entered, the function block PML_AdminTime should be called cyclically in the program.

Syntax

```
METHOD M_ClearWarning : BOOL
VAR_IN_OUT
    stAdmin          : ST_PMLa;
END_VAR
VAR_INPUT
    stWarning        : ST_Alarm;
END_VAR
```

Sample call:

```
WarningCleared := fbAdminAlarm.M_ClearWarning(stAdmin := PackTags.Admin, stWarning := Warning);
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.1.7 M_SetStopReason

This method inserts a StopReason in the Admin-Tags. StopReason[].Trigger is set to TRUE and the value from Admin.PlcDateTime is entered in StopReason[].DateTime. The other values are taken from the transferred StopReason structure. The method returns TRUE if the StopReason was entered successfully. If the StopReason array is already full of entries, the oldest entry is deleted as a result.



So that a valid timestamp can be entered, the function block PML_AdminTime should be called cyclically in the program.

Syntax

```
METHOD M_SetStopReason : BOOL
VAR_IN_OUT
  stAdmin      : ST_PMLa;
END_VAR
VAR_INPUT
  stStopReason : ST_Alarm;
END_VAR
```

Sample call:

```
StopReasonInserted := fbAdminAlarm.M_SetStopReason
(stAdmin := PackTags.Admin, stStopReason := StopReason);
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.1.8 M_AcknowledgeStopReason

This method acknowledges a StopReason in the Admin-Tags. StopReason[].Trigger is set to FALSE and the value from Admin.PlcDateTime is entered in StopReason[].AckDateTime. The method returns TRUE if the StopReason was found and acknowledged successfully. The StopReason remains in the StopReason array until it is pushed out of the array by the next StopReason.



So that a valid timestamp can be entered, the function block PML_AdminTime should be called cyclically in the program.

Syntax

```
METHOD M_AcknowledgeAlarm : BOOL
VAR_IN_OUT
  stAdmin      : ST_PMLa;
END_VAR
VAR_INPUT
  stStopReason : ST_Alarm;
END_VAR
```

Sample call:

```
StopReasonAcknowledged := fbAdminAlarm.M_AcknowledgeStopReason(stAdmin := PackTags.Admin, stStopReason := StopReason);
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.1.9 M_ClearStopReason

This method deletes a StopReason from the Admin-Tags. StopReason[].Trigger is set to FALSE. The method returns TRUE if the StopReason was deleted successfully. The StopReason remains in the StopReason array until it is pushed out of the array by the next StopReason.



So that a valid timestamp can be entered, the function block PML_AdminTime should be called cyclically in the program.

Syntax

```
METHOD M_ClearAlarm : BOOL
VAR_IN_OUT
  stAdmin      : ST_PMLa;
END_VAR
VAR_INPUT
  stStopReason : ST_Alarm;
END_VAR
```

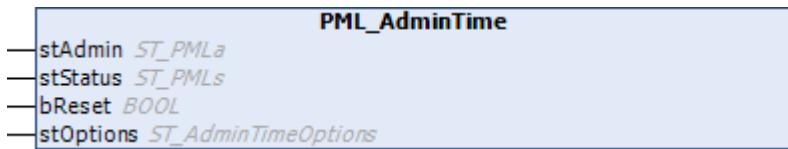
Sample call:

```
StopReasonCleared := fbAdminAlarm.M_ClearStopReason(stAdmin := PackTags.Admin, stStopReason := StopReason);
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.2.2 PML_AdminTime



This function block should be called cyclically and it then fills the following Admin-PackTags:

- PlcDateTime
- AccTimeSinceReset
- ModeCurrentTime[]
- ModeCumulativeTime[]
- StateCurrentTime[][]
- StateCumulativeTime[][]

The length of time that the machine was in different states for is thus recorded. In the further process, this allows conclusions to be drawn about the machine efficiency. To ensure that the times are calculated correctly, it is a prerequisite that the *UnitCurrent* and *StateCurrent* Status-PackTags have already been written coherently.

Inputs

```
VAR_INPUT
    bReset      : BOOL;
    stOptions   : ST_AdminTimeOptions;
END_VAR
```

Name	Type	Description
bReset	BOOL	A signal at this input resets the recorded times
stOptions	ST_AdminTimeOptions	Additional options of the function block

Inputs/Outputs

```
VAR_IN_OUT
    stAdmin     : ST_PMLa;
    stStatus    : ST_PMLs;
END_VAR
```

Name	Type	Description
stAdmin	ST_PMLa	Transfer of the Admin-PackTags
stStatus	ST_PMLs	Transfer of the Status-PackTags

Requirements

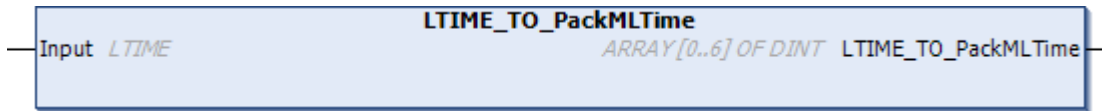
Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.3 Conversion

2.3.3.1 Time

These function convert time values into the PackML-compliant array.

2.3.3.1.1 LTIME_TO_PackMLTime



This function converts a time value in LTIME format into the PackML-compliant array.

FUNCTION LTIME_TO_PackMLTime : ARRAY [0..6] OF DINT;

🔧 Inputs

```

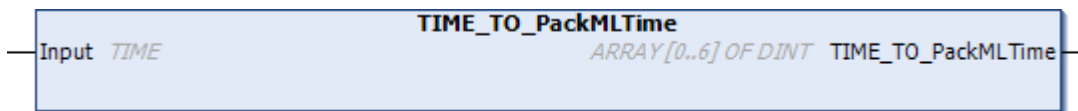
VAR_INPUT
  Input      : LTIME;
END_VAR
  
```

Name	Type	Description
Input	LTIME	The time value to be converted.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.3.1.2 TIME_TO_PackMLTime



This function converts a time value in TIME format into the PackML-compliant array.

FUNCTION TIME_TO_PackMLTime : ARRAY [0..6] OF DINT;

🔧 Inputs

```

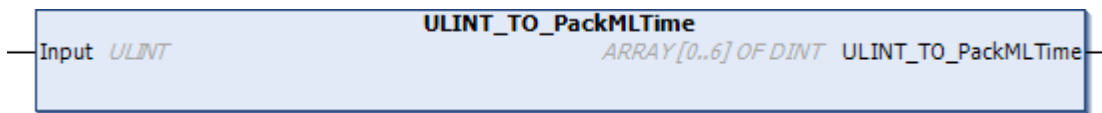
VAR_INPUT
  Input      : TIME;
END_VAR
  
```

Name	Type	Description
Input	TIME	The time value to be converted.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.3.1.3 ULINT_TO_PackMLTime



This function converts a time value in ULINT format into the PackML-compliant array.

FUNCTION ULINT_TO_PackMLTime : ARRAY [0..6] OF DINT;

Inputs

```
VAR_INPUT
  Input      : ULINT;
END_VAR
```

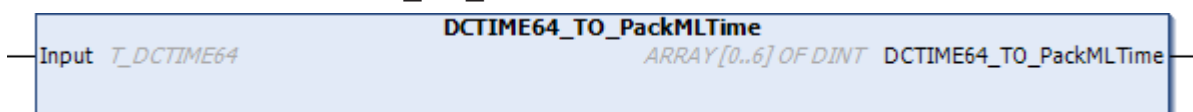
Name	Type	Description
Input	ULINT	The time value to be converted.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.3.2 Timestamp

2.3.3.2.1 DCTIME64_TO_PackMLTime



This function converts a time in DCTIME64 format into the PackML-compliant array.

FUNCTION DCTIME64_TO_PackMLTime : ARRAY [0..6] OF DINT;

Inputs

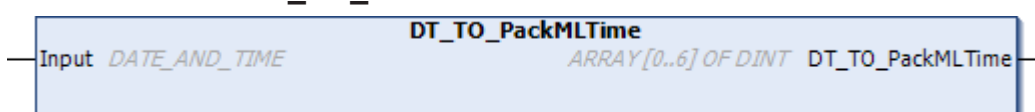
```
VAR_INPUT
  Input      : DCTIME64;
END_VAR
```

Name	Type	Description
Input	DCTIME64	The time to be converted

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.3.2.2 DT_TO_PackMLTime



This function converts a time in DT format into the PackML-compliant array.

FUNCTION DT_TO_PackMLTime : ARRAY [0..6] OF DINT;

 **Inputs**

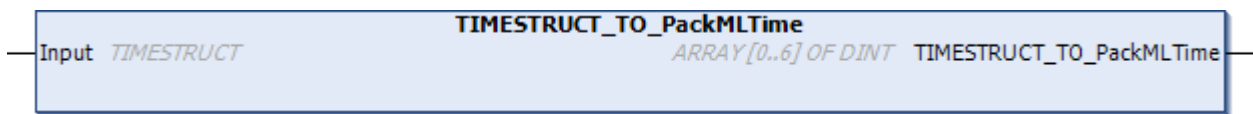
```
VAR_INPUT
    Input      : DT;
END_VAR
```

Name	Type	Description
Input	DT	The time to be converted

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.3.2.3 TIMESTRUCT_TO_PackMLTime



This function converts a time in TIMESTRUCT format into the PackML-compliant array.

FUNCTION TIMESTRUCT_TO_PackMLTime : ARRAY [0..6] OF DINT;

 **Inputs**

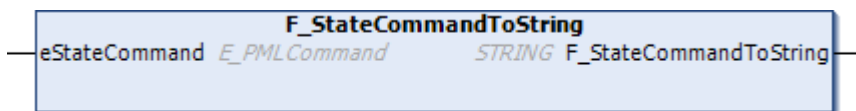
```
VAR_INPUT
    Input      : TIMESTRUCT;
END_VAR
```

Name	Type	Description
Input	TIMESTRUCT	The time to be converted

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.3.3 F_StateCommandToString



This function outputs the name of a state command as a string.

FUNCTION F_StateCommandToString : STRING;

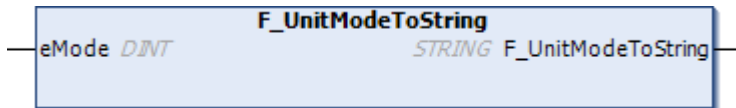
 **Inputs**

```
VAR_INPUT
    eStateCommand : E_PMLCommand;
END_VAR
```

Name	Type	Description
eStateCommand	E_PMLCommand	The state command for which the name is to be determined

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.3.4 F_UnitModeToString

This function returns the name of a Unit Mode as a string.

FUNCTION F_UnitModeToString : STRING;

Inputs

```
VAR_INPUT
  eMode          : DINT;
END_VAR
```

Name	Type	Description
eMode	DINT	The Unit Mode for which the name is to be determined

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3 Packaging Machine Tags

3.1 Introduction

PackTags provides a uniform set of naming conventions for data elements that are used in the procedural elements of the Base State Model. As described, the Base State Model provides a uniform set of machine states, so that all automated machines can be considered in the same way. PackTags are data elements provided with names for the interoperable data exchange between automated machines with open architectures. This documentation contains the key names of the data elements, data types, values, fields and data structures, if applicable. PackTags are used for machine-to-machine communication, e.g. between a bottle filler and a cap fitter. PackTags can also be used for the exchange of data between a machine and superordinated information systems such as Manufacturing Operations Management and Enterprise Information Systems.

The documentation describes all PackTags for the navigation through a state model and for the definition and actuation of the system control mode. Furthermore, this documentation defines a list of PackTags, which may describe important machine information. All PackTags must be used in order to conform to the principles of integrated connectivity with systems with the same implementation.

The tags required are those that are needed for the function of the automated machine or for the connectivity to control or remote systems.

3.2 Tag Types

PackTags are broken down into three groups: Command, Status and Administration. Command and State tags contain data for interfacing the machine with the line control for coordination or for downloading recipes/parameters. Command tags are transferred as “information recipients” to the machine program and “consumed” by it. State tags are created and read by the machine program. Administration tags contain data, which are collected by higher-level systems for machine performance analysis or operator information.

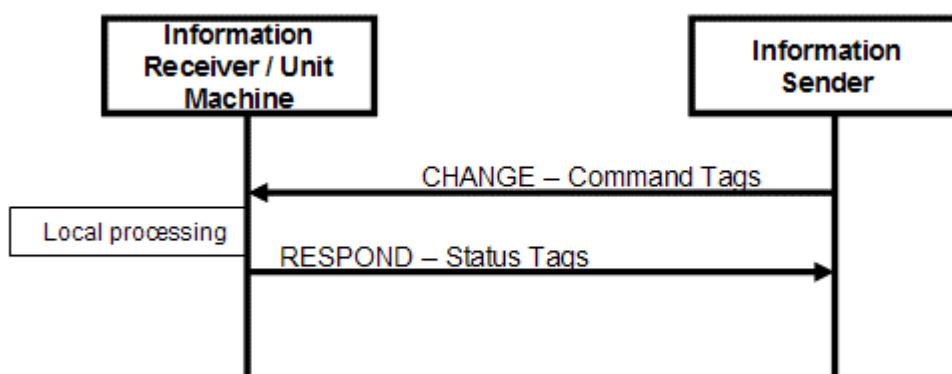
The grouping of data should take place in adjacent registers, in order to optimize the communication.

Information data are usually transferred via OPC in an Ethernet-based communication network.

The prefix of Command tags is “PMLc”.

The prefix of State tags is “PMLs”.

The prefix of Administration tags is “PMLa”.



3.3 Tag Details

The following section provides an overview of the tags. Command, state and administration PackTags are listed in the following tables.

Command structure PMLc

				Tag Name	Data Type
PMLc				PMLc	ST_PMLc
	UnitMode			PMLc.UnitMode	DINT
	UnitModeChangeRequest			PMLc.UnitModeChangeRequest	BOOL
	MachSpeed			PMLc.MachSpeed	REAL
	MaterialInterlock			PMLc.MaterialInterlock	DINT
	CntrlCmd			PMLc.CntrlCmd	DINT
	CmdChangeRequest			PMLc.CmdChangeRequest	BOOL
	RemoteInterface[#]			PMLc.RemoteInterface[#]	ST_Interface
		Number		PMLc.RemoteInterface[#].Number	DINT
		ControlCmdNumber		PMLc.RemoteInterface[#].ControlCmdNumber	DINT
		CmdValue		PMLc.RemoteInterface[#].CmdValue	DINT
		Parameter[#]		PMLc.RemoteInterface[#].Parameter[#]	ST_Descriptor
			Id	PMLc.RemoteInterface[#].Parameter[#].Id	DINT
			Name	PMLc.RemoteInterface[#].Parameter[#].Name	STRING
			Unit	PMLc.RemoteInterface[#].Parameter[#].Unit	STRING(5)
			Value	PMLc.RemoteInterface[#].Parameter[#].Value	REAL
	Parameter[#]			PMLc.Parameter[#]	ST_Descriptor
		Id		PMLc..Parameter[#].Id	DINT
		Name		PMLc..Parameter[#].Name	STRING
		Unit		PMLc..Parameter[#].Unit	STRING(5)
		Value		PMLc..Parameter[#].Value	REAL
	Product[#]			PMLc.Product[#]	ST_Product
		ProductId		PMLc.Product[#]	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(5)
			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(5)
			Value	PMLc.Product[#].Ingredients[#].Parameter[#].Value	REAL

State structure PMLs

				Tag Name	Data Type
PMLs				PMLs	ST_PMLs
	UnitModeCurrent			PMLs.UnitModeCurrent	DINT
	UnitModeRequested			PMLs.UnitModerequested	DINT
	UnitModeChangeInProcesses			PMLs.UnitModeChangeInProcess	BOOL
	StateCurrent			PMLs.StateCurrent	DINT
	StateRequested			PMLs.StateRequested	DINT
	StateChangeInProcess			PMLs.StateChangeInProcess	BOOL
	MachineSpeed			PMLs.MachineSpeed	REAL
	CurMachineSpeed			PMLs.CurMachineSpeed	REAL
	MaterialInterlock			PMLs.MaterialInterlock	DINT
	EquipmentInterlock			PMLs.EquipmentInterlock	ST_Equipment
		Blocked		PMLs.EquipmentInterlock.Blocked	BOOL
		Starved		PMLs.EquipmentInterlock.Starved	BOOL
	RemoteInterface[#]			PMLs.RemoteInterface[#]	ST_Interface
		Number		PMLs.RemoteInterface[#].Number	DINT
		ControlCmdNumber		PMLs.RemoteInterface[#].ControlCmdNumber	DINT
		CmdValue		PMLs.RemoteInterface[#].CmdValue	DINT
		Parameter[#]		PMLs.RemoteInterface[#].Parameter[#]	ST_Descriptor
			Id	PMLs.RemoteInterface[#].Parameter[#].Id	DINT
			Name	PMLs.RemoteInterface[#].Parameter[#].Name	STRING
			Unit	PMLs.RemoteInterface[#].Parameter[#].Unit	STRING(5)
			Value	PMLs.RemoteInterface[#].Parameter[#].Value	REAL
	Parameter[#]			PMLs.Parameter[#]	ST_Descriptor
		Id		PMLs..Parameter[#].Id	DINT
		Name		PMLs..Parameter[#].Name	STRING
		Unit		PMLs..Parameter[#].Unit	STRING(5)
		Value		PMLs..Parameter[#].Value	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(5)
			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(5)
			Value	PMLc.Product[#].Ingredients[#].Parameter[#].Value	REAL

PMLa administration structure

				Tag Name	Data Type
PMLa				Admin	ST_PMLa
	Parameter[#]			PMLa.Parameter[#]	ST_Descriptor
		Id		PMLa..Parameter[#].Id	DINT
		Name		PMLa..Parameter[#].Name	STRING
		Unit		PMLa..Parameter[#].Unit	STRING(5)
		Value		PMLa..Parameter[#].Value	REAL
	Alarm[#]			PMLa.Alarm[#]	ST_Alarm
		Trigger		PMLa.Alarm[#].Trigger	BOOL
		Id		PMLa.Alarm[#].Id	DINT
		Value		PMLa.Alarm[#].Value	DINT
		Message		PMLa.Alarm[#].Message	STRING
		Category		PMLa.Alarm[#].Category	DINT
		DateTime		PMLa.Alarm[#].DateTime	ARRAY [0..6] OF DINT
			Year	PMLa.Alarm[#].DateTime[0]	DINT
			Month	PMLa.Alarm[#].DateTime[1]	DINT
			Day	PMLa.Alarm[#].DateTime.[2]	DINT
			Hour	PMLa.Alarm[#].DateTime[3]	DINT
			Minute	PMLa.Alarm[#].DateTime[4]	DINT
			Second	PMLa.Alarm[#].DateTime[5]	DINT
			mSec	PMLa.Alarm[#].DateTime[6]	DINT
		AckDateTime		PMLa.Alarm[#].AckDateTime	ARRAY [0..6] OF DINT
			Year	PMLa.Alarm[#].AckDateTime[0]	DINT
			Month	PMLa.Alarm[#].AckDateTime[1]	DINT
			Day	PMLa.Alarm[#].AckDateTime[2]	DINT
			Hour	PMLa.Alarm[#].AckDateTime[3]	DINT
			Minute	PMLa.Alarm[#].AckDateTime[4]	DINT
			Second	PMLa.Alarm[#].AckDateTime[5]	DINT
			mSec	PMLa.Alarm[#].AckDateTime[6]	DINT
	AlarmExtent			PMLa.AlarmExtent	DINT
	AlarmHistory[#]			PMLa.AlarmHistory[#]	ST_Alarm
		Trigger		PMLa.AlarmHistory[#].Trigger	BOOL
		Id		PMLa.AlarmHistory[#].Id	DINT
		Value		PMLa.AlarmHistory[#].Value	DINT
		Message		PMLa.AlarmHistory[#].Message	STRING
		Category		PMLa.AlarmHistory[#].Category	DINT
		DateTime		PMLa.AlarmHistory[#].DateTime	ARRAY [0..6] OF DINT
			Year	PMLa.AlarmHistory[#].DateTime[0]	DINT
			Month	PMLa.AlarmHistory[#].DateTime[1]	DINT
			Day	PMLa.AlarmHistory[#].DateTime[2]	DINT
			Hour	PMLa.AlarmHistory[#].DateTime[3]	DINT
			Minute	PMLa.AlarmHistory[#].DateTime[4]	DINT
			Second	PMLa.AlarmHistory[#].DateTime[5]	DINT
			mSec	PMLa.AlarmHistory[#].DateTime[6]	DINT
		AckDateTime		PMLa.AlarmHistory[#].AckDateTime	ARRAY [0..6] OF DINT
			Year	PMLa.AlarmHistory[#].AckDateTime[0]	DINT
			Month	PMLa.AlarmHistory[#].AckDateTime[1]	DINT
			Day	PMLa.AlarmHistory[#].AckDateTime[2]	DINT
			Hour	PMLa.AlarmHistory[#].AckDateTime[3]	DINT
			Minute	PMLa.AlarmHistory[#].AckDateTime[4]	DINT
			Second	PMLa.AlarmHistory[#].AckDateTime[5]	DINT
			mSec	PMLa.AlarmHistory[#].AckDateTime[6]	DINT
	AlarmHistoryExtent			PMLa.AlarmHistoryExtent	DINT
	StopReason[#]			PMLa.StopReason[#]	ST_Alarm
		Trigger		PMLa.StopReason[#].Trigger	BOOL
		Id		PMLa.StopReason[#].Id	DINT

		Value		PMLa.StopReason[#].Value	DINT
		Message		PMLa.StopReason[#].Message	STRING
		Category		PMLa.StopReason[#].Category	DINT
		DateTime		PMLa.StopReason[#].DateTime	ARRAY [0..6] OF DINT
			Year	PMLa.StopReason[#].DateTime[0]	DINT
			Month	PMLa.StopReason[#].DateTime[1]	DINT
			Day	PMLa.StopReason[#].DateTime[2]	DINT
			Hour	PMLa.StopReason[#].DateTime[3]	DINT
			Minute	PMLa.StopReason[#].DateTime[4]	DINT
			Second	PMLa.StopReason[#].DateTime[5]	DINT
			mSec	PMLa.StopReason[#].DateTime[6]	DINT
		AckDateTime		PMLa.StopReason[#].AckDateTime	ARRAY [0..6] OF DINT
			Year	PMLa.StopReason[#].AckDateTime[0]	DINT
			Month	PMLa.StopReason[#].AckDateTime[1]	DINT
			Day	PMLa.StopReason[#].AckDateTime[2]	DINT
			Hour	PMLa.StopReason[#].AckDateTime[3]	DINT
			Minute	PMLa.StopReason[#].AckDateTime[4]	DINT
			Second	PMLa.StopReason[#].AckDateTime[5]	DINT
			mSec	PMLa.StopReason[#].AckDateTime[6]	DINT
	StopReasonExtent			PMLa.StopReasonExtent	DINT
	Warning[#]			PMLa.Warning[#]	ST_Alarm
		Trigger		PMLa.Warning [#].Trigger	BOOL
		Id		PMLa.Warning[#].Id	DINT
		Value		PMLa.Warning[#].Value	DINT
		Message		PMLa.Warning[#].Message	STRING
		Category		PMLa.Warning[#].Category	DINT
		DateTime		PMLa.Warning[#].DateTime	ARRAY [0..6] OF DINT
			Year	PMLa.Warning[#].DateTime[0]	DINT
			Month	PMLa.Warning[#].DateTime[1]	DINT
			Day	PMLa.Warning[#].DateTime[2]	DINT
			Hour	PMLa.Warning[#].DateTime[3]	DINT
			Minute	PMLa.Warning[#].DateTime[4]	DINT
			Second	PMLa.Warning[#].DateTime[5]	DINT
			mSec	PMLa.Warning[#].DateTime[6]	DINT
		AckDateTime		PMLa.Warning[#].AckDateTime	ARRAY [0..6] OF DINT
			Year	PMLa.Warning[#].AckDateTime[0]	DINT
			Month	PMLa.Warning[#].AckDateTime[1]	DINT
			Day	PMLa.Warning[#].AckDateTime[2]	DINT
			Hour	PMLa.Warning[#].AckDateTime[3]	DINT
			Minute	PMLa.Warning[#].AckDateTime[4]	DINT
			Second	PMLa.Warning[#].AckDateTime[5]	DINT
			mSec	PMLa.Warning[#].AckDateTime[6]	DINT
	WarningExtent			PMLa.WarningExtent	DINT
	ModeCurrentTime[#]			PMLa.ModeCurrentTime[#]	DINT
	ModeCumulativeTime[#]			PMLa.ModeCumulativeTime[#]	DINT
	StateCurrentTime[#, #]			PMLa.StateCurrentTime[#, #]	DINT
	StateCumulativeTime[#, #]			PMLa.StateCumulativeTime[#, #]	DINT
	ProdConsumedCount[#]			PMLa.ProdConsumedCount[#]	
		Id		PMLa.ProdConsumedCount[#].Id	DINT
		Name		PMLa.ProdConsumedCount[#].Name	STRING
		Unit		PMLa.ProdConsumedCount[#].Unit	STRING(5)
		Count		PMLa.ProdConsumedCount[#].Count	DINT
		AccCount		PMLa.ProdConsumedCount[#].AccCount	DINT
	ProdProcessedCount[#]			PMLa.ProdProcessedCount[#]	
		Id		PMLa.ProdProcessedCount[#].Id	DINT

		Name		PMLa.ProdProcessedCount[#].Name	STRING
		Unit		PMLa.ProdProcessedCount[#].Unit	STRING(5)
		Count		PMLa.ProdProcessedCount[#].Count	DINT
		AccCount		PMLa.ProdProcessedCount[#].AccCount	DINT
	ProdDefectiveCount[#]			PMLa.ProdDefectiveCount[#]	
		Id		PMLa.ProdDefectiveCount[#].Id	DINT
		Name		PMLa.ProdDefectiveCount[#].Name	STRING
		Unit		PMLa.ProdDefectiveCount[#].Unit	STRING(5)
		Count		PMLa.ProdDefectiveCount[#].Count	DINT
		AccCount		PMLa.ProdDefectiveCount[#].AccCount	DINT
	AccTimeSinceReset			PMLa.AccTimeSinceReset	DINT
	MachDesignSpeed			PMLa.MachDesignSpeed	REAL
	StatesDisabled			PMLa.StatesDisabled	DINT
	PlcDateTime			PMLa.PlcDateTime	ARRAY [0..6] OF DINT
		Year		PMLa.PlcDateTime[0]	DINT
		Month		PMLa.PlcDateTime[1]	DINT
		Day		PMLa.PlcDateTime[2]	DINT
		Hour		PMLa.PlcDateTime[3]	DINT
		Minute		PMLa.PlcDateTime[4]	DINT
		Second		PMLa.PlcDateTime[5]	DINT
		mSec		PMLa.PlcDateTime[6]	DINT

3.4 Data types

3.4.1 Alarm

3.4.1.1 ST_Alarm

Collection of tags for the description of alarm events.

```

TYPE ST_Alarm :
STRUCT
  Trigger          : BOOL;
  Id               : DINT;
  Value            : DINT;
  Message          : STRING;
  Category         : DINT;
  DateTime         : ARRAY [0..6] OF DINT;
  AckDateTime      : ARRAY [0..6] OF DINT;
END_STRUCT
END_TYPE
    
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.1.2 ST_DateAndTime

This structure is used for saving the date and time of an event or for the acknowledgement of an event.

```

TYPE ST_DateAndTime :
STRUCT
  Year             : DINT;
  Month            : DINT;
  Day              : DINT;
  Hour             : DINT;
  Minute           : DINT;
  Second           : DINT;
  mSec             : DINT;
END_STRUCT
    
```

```
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.2 Common

3.4.2.1 ST_Count

Collection of tags for the description of parameters in the machine.

```
TYPE ST_Count :
STRUCT
  Id          : DINT;
  Name       : STRING;
  Unit       : STRING(5);
  Count      : DINT;
  AccCount   : DINT;
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.2.2 ST_Descriptor

Collection of tags for the description of parameters in the machine.

```
TYPE ST_Descriptor :
STRUCT
  Id          : DINT;
  Name       : STRING;
  Unit       : STRING(5);
  Value      : REAL;
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.2.3 ST_Equipment

Collection of tags for the description of parameters in the machine.

```
TYPE ST_Descriptor :
STRUCT
  Blocked      : BOOL;
  Starved      : BOOL;
END_STRUCT
END_TYPE
```


Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.2.4 ST_Ingredient

Collection of tags for the description of the raw materials required for the product.

```

TYPE ST_Ingredient :
STRUCT
    IngredientId      : DINT;
    Parameter         : ARRAY [1..MaxIngredientParameters] OF ST_Descriptor;
END_STRUCT
END_TYPE
    
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.2.5 ST_Interface

Collection of tags for the description of materials in the machine.

```

TYPE ST_Interface :
STRUCT
    Number            : DINT;
    ControlCmdNumber  : DINT;
    CmdValue          : DINT;
    Parameter         : ARRAY [1..MaxInterfaceParameters] OF ST_Descriptor;
END_STRUCT
END_TYPE
    
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.2.6 ST_Product

Collection of tags for the description of the product manufactured on the machine.

```

TYPE ST_Product :
STRUCT
    ProductId         : DINT;
    ProcessVariables  : ARRAY [1..MaxProductProcessVariables] OF ST_Descriptor;
    Ingredients       : ARRAY [1..MaxIngredients] OF ST_Ingredient;
END_STRUCT
END_TYPE
    
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.3 ST_PMLa

Collection of all Administration tags of the PackTag structure.

```

TYPE ST_PMLa :
STRUCT
  Parameter          : ARRAY [1..MaxAdminParameters] OF ST_Descriptor;
  Alarm              : ARRAY [1..MaxAlarms] OF ST_Alarm;
  AlarmExtent        : DINT := MaxAlarms;
  AlarmHistory        : ARRAY [1..MaxHistoryAlarms] OF ST_Alarm;
  AlarmHistoryExtent : DINT := MaxHistoryAlarms;
  StopReason          : ARRAY [1..MaxStopReasons] OF ST_Alarm;
  StopReasonExtent   : DINT := MaxStopReasons;
  Warning             : ARRAY [1..MaxWarnings] OF ST_Alarm;
  WarningExtent       : DINT := MaxWarnings;
  ModeCurrentTime     : ARRAY [1..MaxUnitMode] OF DINT;
  ModeCummulativeTime : ARRAY [1..MaxUnitMode] OF DINT;
  StateCurrentTime    : ARRAY [1..MaxUnitMode, 0..MaxMachineState] OF DINT;
  StateCummulativeTime : ARRAY [1..MaxUnitMode, 0..MaxMachineState] OF DINT;
  ProdConsumedCount   : ARRAY [1..MaxConsumedCounts] OF ST_Count;
  ProdProcessedCount  : ARRAY [1..MaxProductCounts] OF ST_Count;
  ProdDefectiveCount  : ARRAY [1..MaxProductCounts] OF ST_Count;
  AccTimeSinceReset   : DINT;
  MachDesignSpeed     : REAL;
  StatesDisabled      : DINT;
  PlcDateTime         : ARRAY [0..6] OF DINT;
END_STRUCT
END_TYPE

```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.4 ST_PMLc

Collection of all Command tags of the PackTag structure.

```

TYPE ST_PMLc :
STRUCT
  UnitMode           : DINT;
  UnitModeChangeRequest : BOOL;
  MachSpeed          : REAL;
  MaterialInterlock  : DINT;
  CntrlCmd           : DINT;
  CmdChangeRequest   : BOOL;
  RemoteInterface     : ARRAY [1..MaxCommandRemoteInterfaces] OF ST_Interface;
  Parameter          : ARRAY [1..MaxCommandParameters] OF ST_Descriptor;
  Product            : ARRAY [1..MaxCommandProducts] OF ST_Product;
END_STRUCT
END_TYPE

```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.4.5 ST_PMLs

Collection of all state tags of the PackTag structure.

```

TYPE ST_PMLs :
STRUCT
  UnitModeCurrent     : DINT;
  UnitModeRequested   : DINT;
  UnitModeChangeInProgress : BOOL;
  StateCurrent        : DINT;
  StateRequested      : DINT;
  StateChangeInProgress : BOOL;

```

```

MachineSpeed          : REAL;
CurMachineSpeed      : REAL;
MaterialInterlock     : DINT;
EquipmentInterlock    : ST_Equipment;
RemoteInterface       : ARRAY [1..MaxStatusRemoteInterfaces] OF ST_Interface;
Parameter            : ARRAY [1..MaxStatusParameters] OF ST_Descriptor;
Product              : ARRAY [1..MaxStatusProducts] OF ST_Product;
END_STRUCT
END_TYPE
    
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.5 Global parameters

Parameters for the construction of the packaging machine tag structures. These can be adapted when inserting the library for the current project.

```

(* PMLc / PMLs *)
  MaxInterfaceParameter      : INT := 5
  MaxProductProcessVariables : INT := 10
  MaxIngredients             : INT := 10
  MaxIngredientParameters    : INT := 10

(* PMLc *)
  MaxCommandRemoteInterfaces : INT := 2
  MaxCommandParameters      : INT := 10
  MaxCommandProducts        : INT := 5

(* PMLs *)
  MaxStatusRemoteInterfaces  : INT := 2
  MaxStatusParameters       : INT := 10
  MaxStatusProducts         : INT := 5

(* PMLa *)
  MaxAdminParameters        : INT := 10
  MaxAlarms                 : INT := 10
  MaxHistoryAlarms         : INT := 10
  MaxStopReasons           : INT := 10
  MaxWarnings               : INT := 10
  MaxConsumedCounts        : INT := 10
  MaxProductCounts         : INT := 10
    
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

3.6 Global Constants

Constants for the construction of the packaging machine tag structures. These cannot be changed.

```

(* PMLa *)
  MaxUnitMode              : INT := 31
  MaxMachineState         : INT := 17
    
```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

4 Example Tc3_PackML_V2

Based on a visualized sorting unit, the sample illustrates how the Tc3_PackML_V2 library can be used as the basis for a machine control system:

https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_PackML_V2/Resources/3414434955/.zip

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

