BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc3_PackML_V2

2021-02-22 | Version: 1.2

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®, 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 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations 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 engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

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

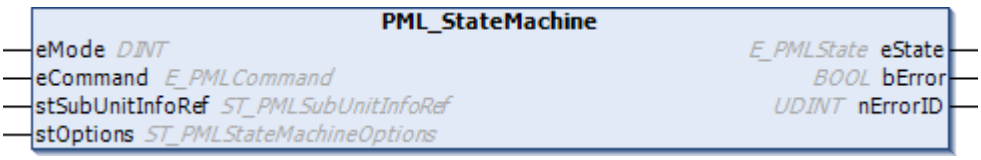
Requirements

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

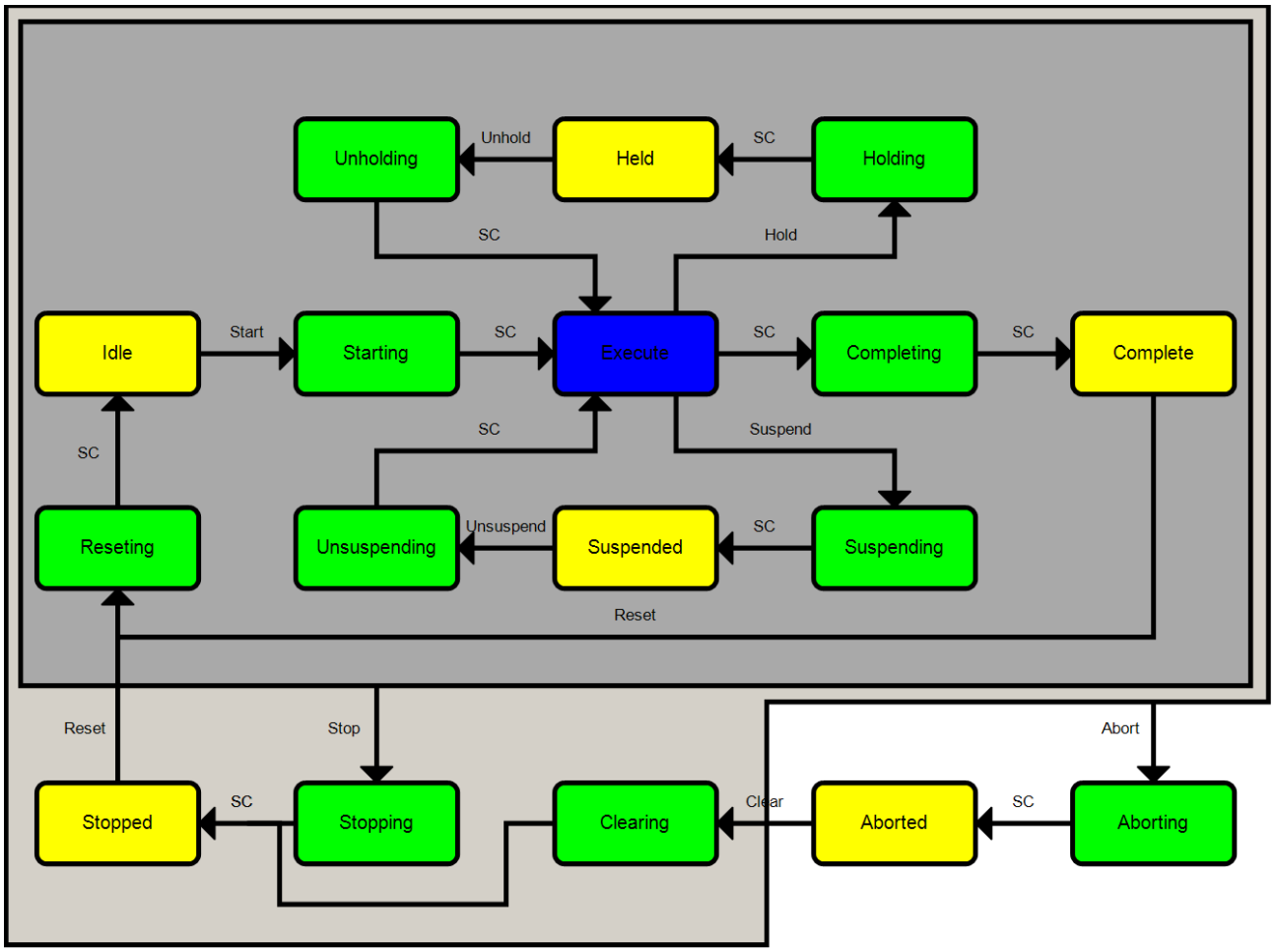
2.3.1 Packaging Machine State

2.3.1.1 PML_StateMachine

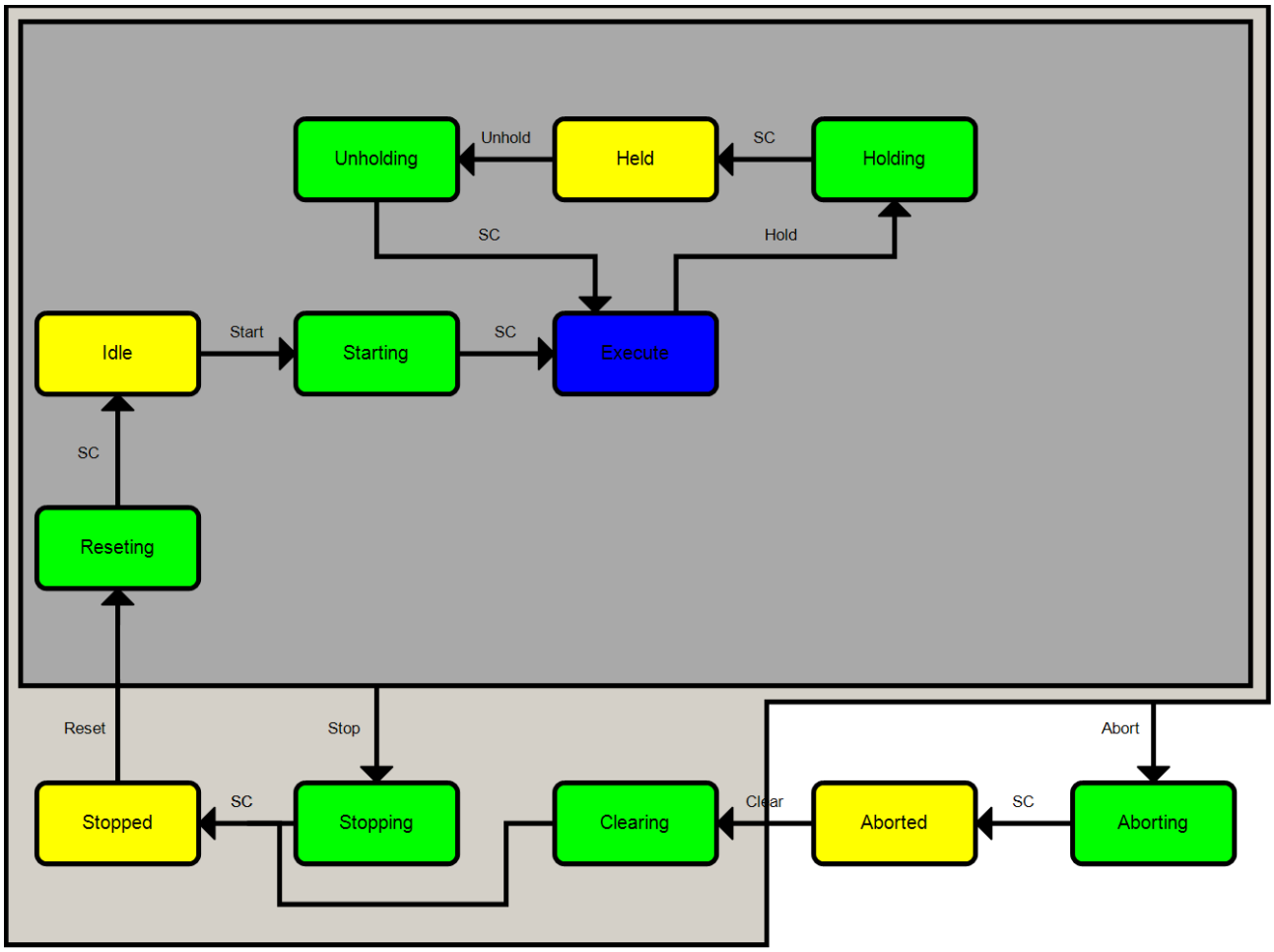


In the updated form the PML_StateMachine function block has a common interface with the PackML Machine State Model V3. It is assumed that application-specific logic, such as state transitions, is programmed in external function blocks and that the PML_StateMachine function block deals with the central 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 [▶ 10]).

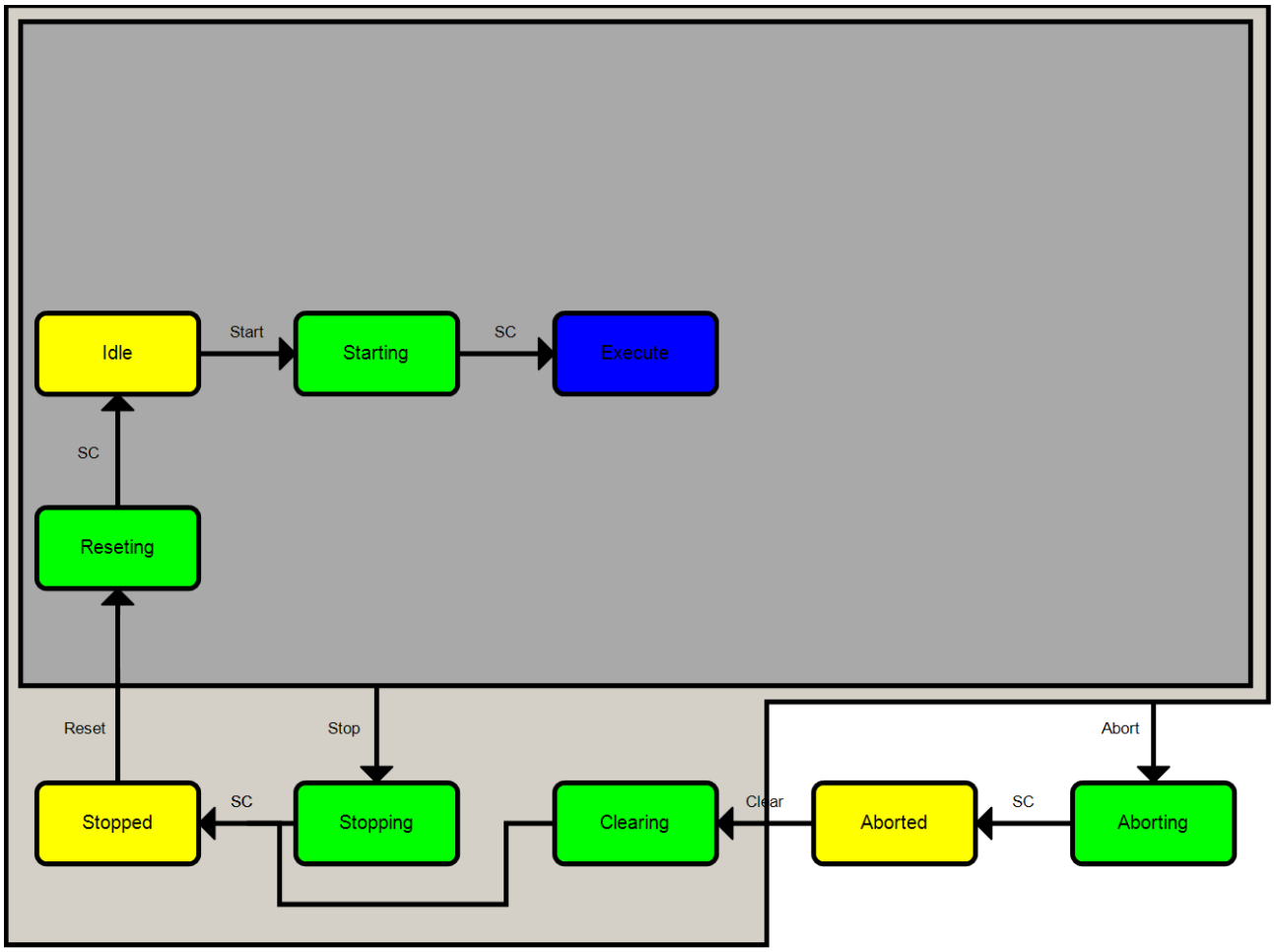
ePMLProtUnitMode_Production



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 [▶ 16] 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 [▶ 19].

Inputs

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

eMode: Current PML UnitMode.

eCommand: Enumeration [▶ 9] with the various PML commands of the function block.

stSubUnitInfoRef: Structure [▶ 11] that points to an array of the current PML states of subordinated machine units

stOptions: Not used at present

Outputs

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

eState: Enumeration [▶ 9] that delivers the current PML state of the automatic state machine.

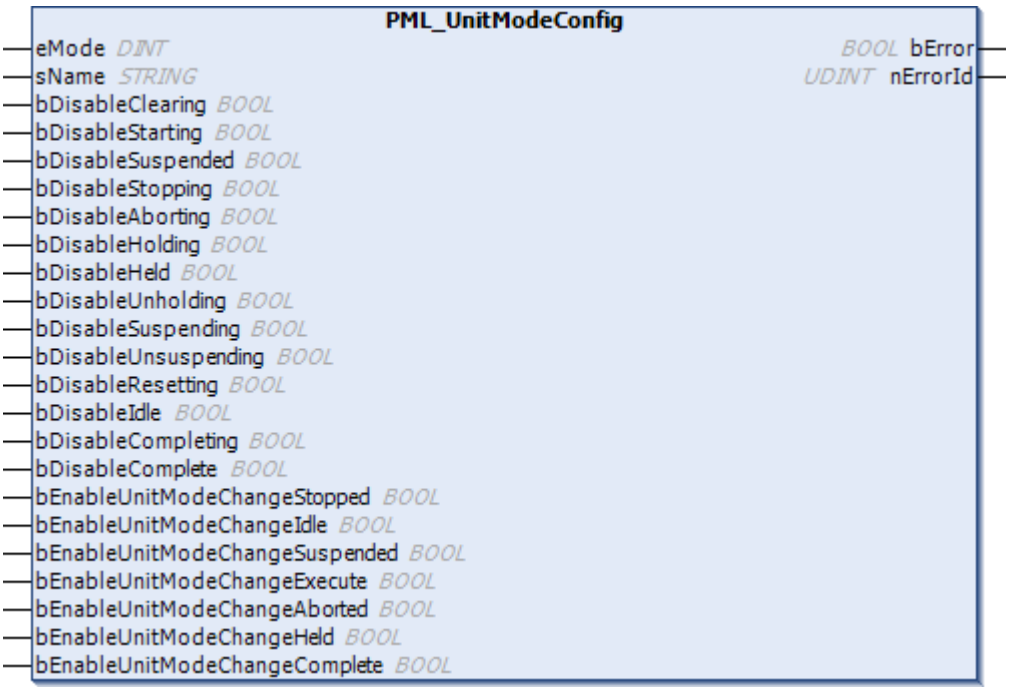
bError: Becomes TRUE, as soon as an error occurs.

nErrorID: Supplies the error number when the output bError 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
```

eMode: Number of the new PML UnitMode [4..31].

sName: Name of the new PML UnitMode.

bDisableClearing: Deactivates the PMLState "Clearing".

bDisableStarting: Deactivates the PMLState "Starting".

bDisableSuspended: Deactivates the PMLState "Suspended".
The deactivation of the static state also causes the PMLStates "Suspending" and "Unsuspending" to be deactivated.

bDisableStopping: Deactivates the PMLState "Stopping".

bDisableAborting: Deactivates the PMLState "Aborting".

bDisableHolding: Deactivates the PMLState "Holding".

bDisableHeld: Deactivates the PMLState "Held".
The deactivation of the static state also causes the PMLStates "Holding" and "Unholding" to be deactivated.

bDisableUnholding: Deactivates the PMLState "Unholding".

bDisableSuspending: Deactivates the PMLState "Suspending".

bDisableUnsuspending: Deactivates the PMLState "Unsuspending".

bDisableResetting: Deactivates the PMLState "Resetting".

bDisableIdle: Deactivates the PMLState "Idle".
The deactivation of the static state also causes the PMLState "Resetting" to be deactivated.

bDisableCompleting: Deactivates the PMLState "Completing".

bDisableComplete: Deactivates the PMLState "Complete".
The deactivation of the static state also causes the PMLState "Completing" to be deactivated

bEnableUnitModeChangeStopped: Enables a mode change in the PMLState "Stopped".

bEnableUnitModeChangeIdle: Enables a mode change in the PMLState "Idle".

bEnableUnitModeChangeSuspended: Enables a mode change in the PMLState "Suspended".

bEnableUnitModeChangeExecute: Enables a mode change in the PMLState "Execute".

bEnableUnitModeChangeAborted: Enables a mode change in the PMLState "Aborted".

bEnableUnitModeChangeHeld: Enables a mode change in the PML state "Held".

bEnableUnitModeChangeComplete: Enables a mode change in the PMLState "Complete".

Outputs

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

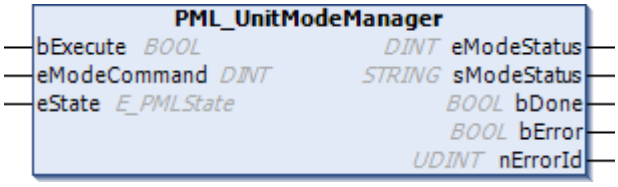
bError: Becomes TRUE, as soon as an error occurs.

nErrorID: Supplies the error number when the output bError is set.

Requirements

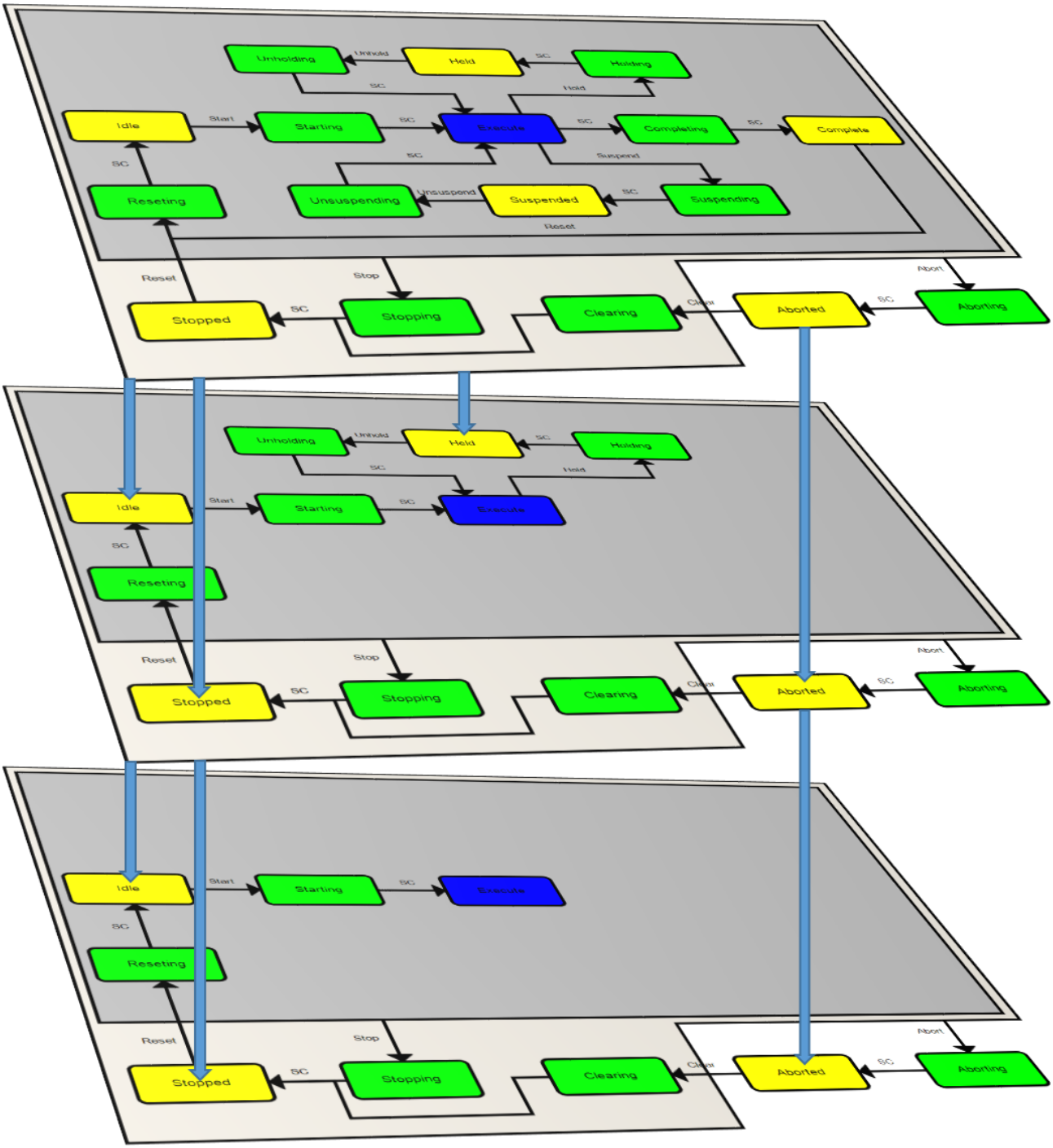
Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

2.3.1.3 PML_UnitModeManager



Machines have system modes other than “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 base modes “Production”, “Maintenance” and “Manual”, as the illustration below shows. This can be individually specified for other modes defined via the PML_UnitModeConfig [▶ 16] function block.

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
```

bExecute: On a rising edge at this input, an attempt is made to perform the mode change.

eModeCommand: Requested mode.

ePMLState: Enumeration [▶ 9] 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
```

eModeStatus: Current PML UnitMode.

sModeStatus: Name of the current PML UnitMode.

bDone: Becomes TRUE as soon as the mode change has been successfully carried out.

bError: Becomes TRUE, as soon as an error occurs.

nErrorID: 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 := War
ning);
```

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
```

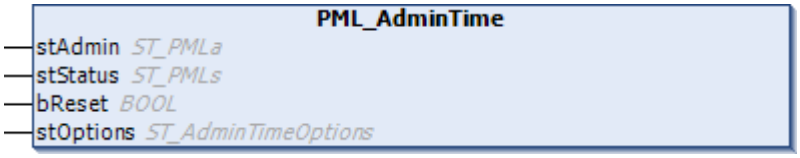
Sample call:

```
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*[]
- *ModeCummulativeTime*[]
- *StateCurrentTime*[][]
- *StateCummulativeTime*[][]

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

Inputs

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

bReset: A signal at this input resets the recorded times.

stOptions: Additional options of the function block.

Inputs/outputs

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

stAdmin: Transfer of the Admin-PackTags

stStatus: Transfer of the Status-PackTags

Requirements

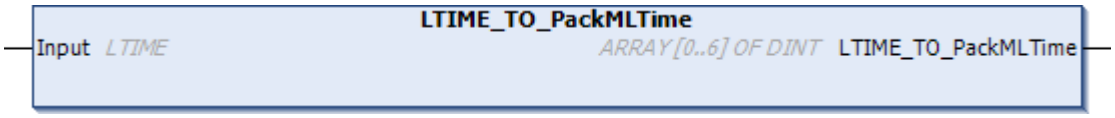
Development environment	Target system type	PLC libraries to include
TwinCAT v3.1 Build 4018 and higher	PC (i386)	Tc3_PackML_V2

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;

VAR_INPUT

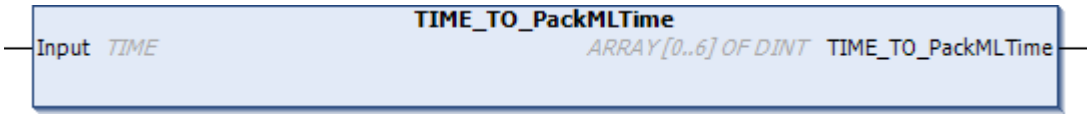
```
VAR_INPUT
    Input         : LTIME;
END_VAR
```

Input: 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;

VAR_INPUT

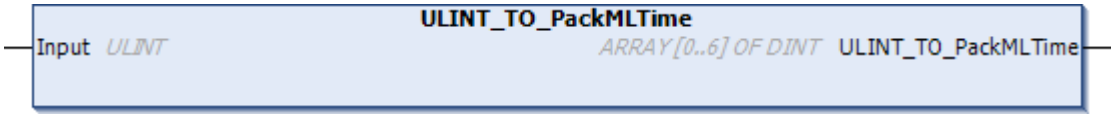
```
VAR_INPUT
    Input         : TIME;
END_VAR
```

Input: 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;

VAR_INPUT

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

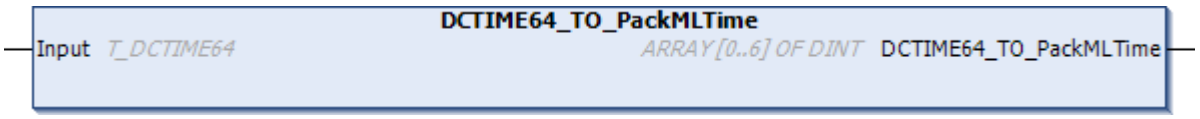
Input: The time value to be converted.

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;

VAR_INPUT

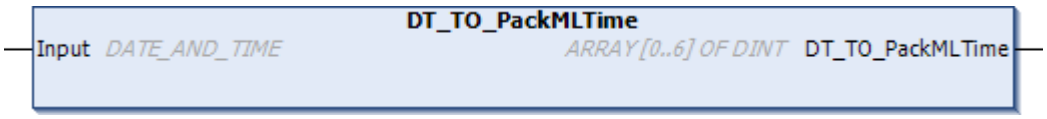
```
VAR_INPUT
    Input         : DCTIME64;
END_VAR
```

Input: 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;

VAR_INPUT

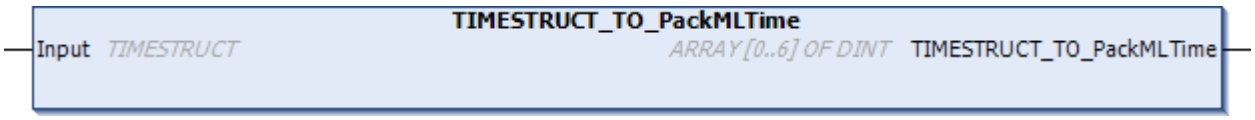
```
VAR_INPUT
    Input          : DT;
END_VAR
```

Input: 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;

VAR_INPUT

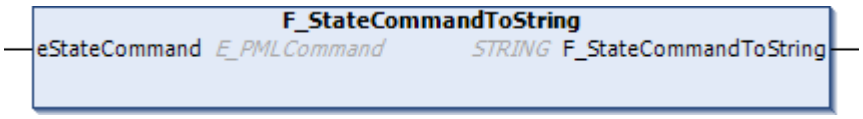
```
VAR_INPUT
    Input          : TIMESTRUCT;
END_VAR
```

Input: 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;

VAR_INPUT

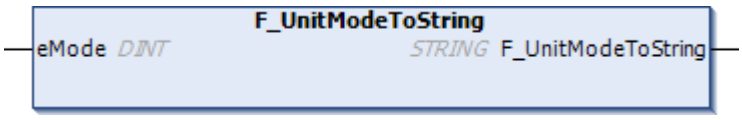
```
VAR_INPUT
    eStateCommand          : E_PMLCommand;
END_VAR
```

eStateCommand: 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;

VAR_INPUT

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

eMode: 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.

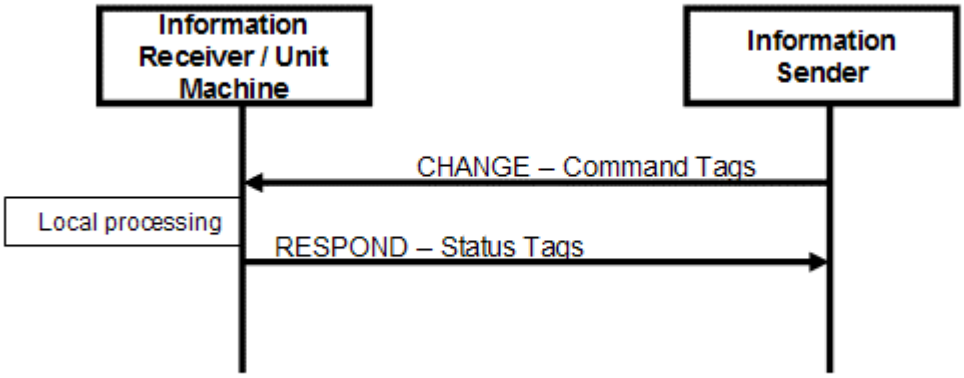
The grouping of data should take place in adjacent registers, in order to optimize the communication.

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[#]	DINT
		ControlCmd-Number			PMLc.RemoteInterface[#]	DINT
		CmdValue			PMLc.RemoteInterface[#]	DINT
		Parameter[#]			PMLc.RemoteInterface[#].Parameter[#]	ST_Descriptor
			Id		PMLc.RemoteInterface[#].Parameter[#].Id	DINT
			Name		PMLc.RemoteInterface[#].Parame-ter[#].Name	STRING
			Unit		PMLc.RemoteInterface[#].Parameter[#].Unit	STRING(5)
			Value		PMLc.RemoteInterface[#].Parame-ter[#].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
		ProcessVari-ables[#]			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
			Ingredien-tId		PMLc.Product[#].Ingredients[#].IngredientId	DINT
			Parame-ter[#]		PMLc.Product[#].Ingredients[#].Parameter[#]	ST_Descriptor
				Id	PMLc.Product[#].Ingredients[#].Parame-ter[#].Id	DINT
				Name	PMLc.Product[#].Ingredients[#].Parame-ter[#].Name	STRING
				Unit	PMLc.Product[#].Ingredients[#].Parame-ter[#].Unit	STRING(5)
				Value	PMLc.Product[#].Ingredients[#].Parame-ter[#].Value	REAL

State structure PMLs

					Tag Name	Data Type
PMLs					PMLs	ST_PMLs
	UnitModeCurrent				PMLs.UnitModeCurrent	DINT
	UnitModeRequested				PMLs.UnitModerequested	DINT
	UnitModeChangeInPro-cess				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[#]	DINT
		ControlCmd-Number			PMLs.RemoteInterface[#]	DINT
		CmdValue			PMLs.RemoteInterface[#]	DINT
		Parameter[#]			PMLs.RemoteInterface[#].Parameter[#]	ST_Descriptor
			Id		PMLs.RemoteInterface[#].Parameter[#].Id	DINT
			Name		PMLs.RemoteInterface[#].Parame-ter[#].Name	STRING
			Unit		PMLs.RemoteInterface[#].Parameter[#].Unit	STRING(5)
			Value		PMLs.RemoteInterface[#].Parame-ter[#].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[#]	DINT
		ProcessVari-ables[#]			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
			Ingredien-tId		PMLc.Product[#].Ingredients[#].IngredientId	DINT
			Parame-ter[#]		PMLc.Product[#].Ingredients[#].Parameter[#]	ST_Descriptor
				Id	PMLc.Product[#].Ingredients[#].Parame-ter[#].Id	DINT
				Name	PMLc.Product[#].Ingredients[#].Parame-ter[#].Name	STRING
				Unit	PMLc.Product[#].Ingredients[#].Parame-ter[#].Unit	STRING(5)
				Value	PMLc.Product[#].Ingredients[#].Parame-ter[#].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
	ModeCummulativeTime[#]				PMLa.ModeCummulativeTime[#]	DINT
	StateCurrentTime[#,#]				PMLa.StateCurrentTime[#,#]	DINT
	StateCummulative-Time[#,#]				PMLa.StateCummulativeTime[#,#]	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_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;
    UnitModeChangeInProcess   : BOOL;
    StateCurrent              : DINT;
    StateRequested            : DINT;
    StateChangeInProcess      : 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 Appendix

4.1 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/zip/9007202669175947.zip

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