Table of contents

1 Foreword ........................................................................................................................................ 5
   1.1 Notes on the documentation ....................................................................................................... 5
   1.2 Safety instructions ....................................................................................................................... 6

2 Introduction ....................................................................................................................................... 7
   2.1 Overview ...................................................................................................................................... 7

3 Packaging Machine State .................................................................................................................... 8
   3.1 DataTypes ...................................................................................................................................... 8
       3.1.1 E_PMLState ............................................................................................................................ 8
       3.1.2 E_PMLUnitMode .................................................................................................................... 9
   3.2 Function Blocks .............................................................................................................................. 10
       3.2.1 PS_PackML_StateMachine_Auto .......................................................................................... 10
       3.2.2 PS_PackML_StateMachine_Maintenance ............................................................................. 12
       3.2.3 PS_PackML_StateMachine_Manual ......................................................................................... 15
       3.2.4 PS_PackML_StateMachine_SemiAuto ................................................................................... 17
       3.2.5 PS_UnitModeManager .......................................................................................................... 20

4 Packaging Machine Functional Tag Description ................................................................................ 23
   4.1 Introduction .................................................................................................................................. 23
   4.2 Tag Types ...................................................................................................................................... 23
   4.3 Tag Details ..................................................................................................................................... 24
   4.4 DataTypes ..................................................................................................................................... 31
       4.4.1 Alarm ...................................................................................................................................... 31
       4.4.2 Common .................................................................................................................................. 34
       4.4.3 ST_PMLa ................................................................................................................................ 36
       4.4.4 ST_PMLc ................................................................................................................................ 37
       4.4.5 ST_PMLs ................................................................................................................................ 37
   4.5 Parameter List ................................................................................................................................. 38
   4.6 Global Constants ............................................................................................................................ 38
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:


with corresponding applications or registrations in various other countries.

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>! DANGER</td>
<td>Serious risk of injury! Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
<tr>
<td>! WARNING</td>
<td>Risk of injury! Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
<tr>
<td>! CAUTION</td>
<td>Personal injuries! Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td>NOTE</td>
<td>Damage to the environment or devices Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

Tip or pointer

This symbol indicates information that contributes to better understanding.
2 Introduction

2.1 Overview

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

**General Objectives of the lib**

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

**Packaging Machine Behavior Organization**

| PS_PackML_StateMachine_Auto | 10 |
| PS_PackML_StateMachine_Maintenance | 12 |
| PS_PackML_StateMachine_Manual | 15 |
| PS_PackML_StateMachine_SemiAuto | 17 |
| PS_UnitModeManager | 20 |

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1 Build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>
3 Packaging Machine State

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

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

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

Pack ML State Machine V3 Function Block Names:

- PS_Starting
- PS_Completing
- PS_Reseting
- 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

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

3.1 DataTypes

3.1.1 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

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

3.1.2 E_PMLUnitMode

E_PMLUnitMode

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

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>
3.2 Function Blocks

3.2.1 PS_PackML_StateMachine_Auto

Packaging Machine State Function Blocks, in updated form, provide a common interface to the PackML Machine State Model V3. It is expected that the application specific logic including the transitions between states is programmed in external function blocks, but the central logic of the state machine and the status representation should be handled by the Pack_ML_State_Machine Function block. Therefore, this FB comes with a recommendation how to combine with other logic.

The logic for transitions depends on the application, especially for those between manual, semi-automatic and automatic mode.
VAR_INPUT

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

Start: Execute State Machine from rising edge, to Starting

Hold: to Holding or Held

UnHold: to un-Holding

Suspend: to Suspending or Suspend

UnSuspend: to un-Suspending

Abort: to Aborting

Stop: to Stopping

Complete: to Resetting

Clear: to Clearing

Reset: to Resetting

StateComplete: Transition

VAR_OUTPUT

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

Status: Status Word representing the status of the State Machine.

ST_Start: True if State Machine is in state Starting

ST_Completing: True if State Machine is in state Completing

ST_Resetting: True if State Machine is in state Resetting.

ST_Holding: True if State Machine is in state Holding.

ST_UnHolding: True if State Machine is in state UnHolding.
ST_Suspending: True if State Machine is in state Suspending.

ST_UnSuspending: True if State Machine is in state UnSuspending.

ST_Clearing: True if State Machine is in state Clearing

ST_Stopping: True if State Machine is in state Stopping

ST_Aborting: True if State Machine is in state Aborting

ST_Execute: True if State Machine is in state Execute

ST_Complete: True if State Machine is in state Complete

ST_Idle: True if State Machine is in state Idle.

ST_Held: True if State Machine is in state Held

ST_Suspended: True if State Machine is in state Suspended

ST_Stopped: True if State Machine is in state Stopped

ST_Aborted: True if State Machine is in state Aborted

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

ErrorID: If the error output is set, this parameter supplies the error number.

ePMLState: Current PML state of the automatic state machine.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

3.2.2 PS_PackML_StateMachine_Maintenance

Packaging Machine State Function Blocks, in updated form, provide a common interface to the PackML Machine State Model V3. It is expected that the application specific logic including the transitions between states is programmed in external function blocks, but the central logic of the state machine and the status representation should be handled by the Pack_ML_State_Machine Function block. Therefore, this FB comes with a recommendation how to combine with other logic.
The logic for transitions depends on the application, especially for those between manual, semi-automatic and automatic mode.

### VAR_INPUT

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

**Start:** Execute State Machine from rising edge, to Starting

**Hold:** to Holding or Held

**UnHold:** to un-Holding

**Suspend:** to Suspending or Suspend

**UnSuspend:** to un-Suspending

**Abort:** to Aborting

**Stop:** to Stopping

**Complete:** to Resetting

**Clear:** to Clearing

**Reset:** to Resetting

**StateComplete:** Transition
VAR_OUTPUT

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

Status: Status Word representing the status of the State Machine.

ST_Starting: True if State Machine is in state Starting.

ST_Completing: True if State Machine is in state Completing

ST_Resetting: True if State Machine is in state Resetting.

ST_Holding: True if State Machine is in state Holding.

ST_UnHolding: True if State Machine is in state UnHolding.

ST_Suspending: True if State Machine is in state Suspending.

ST_UnSuspending: True if State Machine is in state UnSuspending.

ST_Clearing: True if State Machine is in state Clearing

ST_Stopping: True if State Machine is in state Stopping

ST_Aborting: True if State Machine is in state Aborting

ST_Execute: True if State Machine is in state Execute

ST_Complete: True if State Machine is in state Complete

ST_Idle: True if State Machine is in state Idle.

ST_Held: True if State Machine is in state Held

ST_Suspended: True if State Machine is in state Suspended

ST_Stopped: True if State Machine is in state Stopped

ST_Aborted: True if State Machine is in state Aborted

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

ErrorID: If the error output is set, this parameter supplies the error number.

ePMLState: Current PML state of the maintenance state machine.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

Packaging Machine State Function Blocks, in updated form, provide a common interface to the PackML Machine State Model V3. It is expected that the application specific logic including the transitions between states is programmed in external function blocks, but the central logic of the state machine and the status representation should be handled by the Pack_ML_State_Machine Function block. Therefore, this FB comes with a recommendation how to combine with other logic.

The logic for transitions depends on the application, especially for those between manual, semi-automatic and automatic mode.

VAR_INPUT

<table>
<thead>
<tr>
<th>Start</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hold</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

Figure: PS_Pack_ML_State_Model_Manual Mode
Start: Execute State Machine from rising edge, to Starting

Hold: to Holding or Held

UnHold: to un-Holding

Suspend: to Suspending or Suspend

UnSuspend: to un-Suspending

Abort: to Aborting

Stop: to Stopping

Complete: to Resetting

Clear: to Clearing

Reset: to Resetting

StateComplete: Transition

VAR_OUTPUT

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

Status: Status Word representing the status of the State Machine.

ST_Starting: True if State Machine is in state Starting.

ST_Completing: True if State Machine is in state Completing

ST_Resetting: True if State Machine is in state Resetting.

ST_Holding: True if State Machine is in state Holding.

ST_UnHolding: True if State Machine is in state UnHolding.

ST_Suspending: True if State Machine is in state Suspending.

ST_UnSuspending: True if State Machine is in state UnSuspending.
Packaging Machine State Function Blocks, in updated form, provide a common interface to the PackML Machine State Model V3. It is expected that the application specific logic including the transitions between states is programmed in external function blocks, but the central logic of the state machine and the status representation should be handled by the Pack_ML_State_Machine Function block. Therefore, this FB comes with a recommendation how to combine with other logic.

The logic for transitions depends on the application, especially for those between manual, semi-automatic and automatic mode.
VAR_INPUT

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

Start: Execute State Machine from rising edge, to Starting
Hold: to Holding or Held
UnHold: to un-Holding
Suspend: to Suspending or Suspend
UnSuspend: to un-Suspending
Abort: to Aborting
Stop: to Stopping
Complete: to Resetting
Clear: to Clearing
Reset: to Resetting
StateComplete: Transition

VAR_OUTPUT

VAR_OUTPUT
Status : WORD;
ST_Starting : BOOL;
END_VAR
Status Word representing the status of the State Machine.

**ST_Starting**: True if State Machine is in state Starting

**ST_Completing**: True if State Machine is in state Completing

**ST_Resetting**: True if State Machine is in state Resetting.

**ST_Holding**: True if State Machine is in state Holding.

**ST_UnHolding**: True if State Machine is in state UnHolding.

**ST_Suspending**: True if State Machine is in state Suspending.

**ST_UnSuspending**: True if State Machine is in state UnSuspending.

**ST_Clearing**: True if State Machine is in state Clearing

**ST_Stopping**: True if State Machine is in state Stopping

**ST_Aborting**: True if State Machine is in state Aborting

**ST_Execute**: True if State Machine is in state Execute

**ST_Complete**: True if State Machine is in state Complete

**ST_Idle**: True if State Machine is in state Idle.

**ST_Held**: True if State Machine is in state Held

**ST_Suspended**: True if State Machine is in state Suspended

**ST_Stopped**: True if State Machine is in state Stopped

**ST_Aborted**: True if State Machine is in state Aborted

**Error**: Becomes TRUE, as soon as an error occurs.

**ErrorID**: If the error output is set, this parameter supplies the error number.

**ePMLState**: Current PML state of the semi automatic state machine.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>
3.2.5 PS_UnitModeManager

Packaging machinery has unit modes other than “automatic”, as noted above. Each unit mode is defined by its own state model. In order to manage the change from one mode to the next a procedure known as a “mode manager” must be defined. The mode manager determines how, and in what state a machine may change unit modes; i.e., the mode manager includes interlocks that prevent the machine changing at inappropriate states.

The logic for transitions between Modes depends on the application, especially for those between manual, semi-automatic and automatic mode. Additionally, these mode changes may require interlocks by means of hardware or safety related equipment. Responsibility for proper application of mode switching resides with whoever implements such means.

See the below figure for example.

Figure: User Mode Operations State Model
VAR_INPUT

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

Execute: Mode change on rising edge.

eModeCommand: Requested Unit Mode. (E_PMLUnitMode [9])

ePMLState: Current PML state of the current mode. (E_PMLState [8])

VAR_OUTPUT

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

VAR_IN_OUT

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

eModeStatus: Current Unit Mode. (E_PMLUnitMode [9])

Done: True if mode change was successfully performed

Error: Signals that an error has occurred within Function Block, e.g. mode change not permitted

ErrorID: If the error output is set, this parameter supplies the error number.
0 = No error;
1 = mode switch not permitted. The ePMLState is not in idle, stopped, aborted, held, suspended or completed or the corresponding state does not exist in the requested mode.

VAR_IN_OUT

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

Machine_ID: To identify the machine executed by the state model.

Implementation

The mode change is restricted to certain modes, see implementation below.

Not all modes are yet implemented.
END_IF

ePMLUnitMode_MAINTENANCE:
  IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLState_IDLE) THEN
    eModeStatus := eModeCommand;
    Done := TRUE;
  ELSIF (ePMLState = ePMLState_HELD) AND ((eModeCommand = ePMLUnitMode_AUTOMATIC) OR (eModeCommand = ePMLUnitMode_SEMIAUTOMATIC)) THEN
    eModeStatus := eModeCommand;
    Done := TRUE;
  ELSE
    Error := TRUE;
    ErrorID := 1;
  END_IF

ePMLUnitMode_MANUAL:
  IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLState_IDLE) THEN
    eModeStatus := eModeCommand;
    Done := TRUE;
  ELSE
    Error := TRUE;
    ErrorID := 1;
  END_IF

ePMLUnitMode_SEMIAUTOMATIC:
  IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLState_IDLE) THEN
    eModeStatus := eModeCommand;
    Done := TRUE;
  ELSIF (ePMLState = ePMLState_SUSPENDED) OR (ePMLState = ePMLState_HELD) OR (ePMLState = ePMLState_COMPLETE))
    AND (eModeCommand = ePMLUnitMode_AUTOMATIC) THEN
    eModeStatus := eModeCommand;
    Done := TRUE;
  ELSIF (ePMLState = ePMLState_HELD) AND (eModeCommand = ePMLUnitMode_MAINTENANCE) THEN
    eModeStatus := eModeCommand;
    Done := TRUE;
  ELSE
    Error := TRUE;
    ErrorID := 1;
  END_IF

ePMLUnitMode_IDLE:
  IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLState_IDLE) THEN
    eModeStatus := eModeCommand;
    Done := TRUE;
  ELSE
    Error := TRUE;
    ErrorID := 1;
  END_IF

ePMLUnitMode_ESTOP:
  IF (ePMLState = ePMLState_STOPPED) OR (ePMLState = ePMLState_ABORTED) OR (ePMLState = ePMLState_IDLE) THEN
    eModeStatus := eModeCommand;
    Done := TRUE;
  ELSE
    Error := TRUE;
    ErrorID := 1;
  END_IF

ELSE
  eModeStatus := eModeCommand;
  Done := TRUE;
END_CASE
END_IF

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>
4 Packaging Machine Functional Tag Description

4.1 Introduction

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

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

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

4.2 Tag Types

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

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

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

Command Tags are prefixed by "PMLc".

Status Tags are prefixed by "PMLs".

Administration Tags are prefixed by "PMLa".

![Diagram showing interaction between information receiver and sender](image-url)
4.3 Tag Details

The following section is a summary listing of the tags. Tables below list the command, status and admin PackTags.
Command structure PMLc
## Tag Name & Data Type

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnitMode</td>
<td>PMLc.UnitMode</td>
</tr>
<tr>
<td>UnitModeChangeRequest</td>
<td>PMLc.UnitModeChangeRequest</td>
</tr>
<tr>
<td>ProcMode</td>
<td>PMLc.ProcMode</td>
</tr>
<tr>
<td>ProcModeChangeRequest</td>
<td>PMLc.ProcModeChangeRequest</td>
</tr>
<tr>
<td>CurMachSpeed</td>
<td>PMLc.CurMachSpeed</td>
</tr>
<tr>
<td>MatReady</td>
<td>PMLc.MatReady</td>
</tr>
<tr>
<td>RawMaterial</td>
<td>PMLc.MatReady.RawMaterial</td>
</tr>
<tr>
<td>CO2</td>
<td>PMLc.MatReady.CO2</td>
</tr>
<tr>
<td>Container</td>
<td>PMLc.MatReady.Container</td>
</tr>
<tr>
<td>Lubrication</td>
<td>PMLc.MatReady.Lubrication</td>
</tr>
<tr>
<td>Water</td>
<td>PMLc.MatReady.Water</td>
</tr>
<tr>
<td>ContainerClosures</td>
<td>PMLc.MatReady.ContainerClosures</td>
</tr>
<tr>
<td>Unused0</td>
<td>PMLc.MatReady.Unused0</td>
</tr>
<tr>
<td>Unused1</td>
<td>PMLc.MatReady.Unused1</td>
</tr>
<tr>
<td>Unused2</td>
<td>PMLc.MatReady.Unused2</td>
</tr>
<tr>
<td>Unused3</td>
<td>PMLc.MatReady.Unused3</td>
</tr>
<tr>
<td>Unused4</td>
<td>PMLc.MatReady.Unused4</td>
</tr>
<tr>
<td>Unused5</td>
<td>PMLc.MatReady.Unused5</td>
</tr>
<tr>
<td>Unused6</td>
<td>PMLc.MatReady.Unused6</td>
</tr>
<tr>
<td>Unused7</td>
<td>PMLc.MatReady.Unused7</td>
</tr>
<tr>
<td>Unused8</td>
<td>PMLc.MatReady.Unused8</td>
</tr>
<tr>
<td>Unused9</td>
<td>PMLc.MatReady.Unused9</td>
</tr>
<tr>
<td>MatLow</td>
<td>PMLc.MatLow</td>
</tr>
<tr>
<td>RawMaterial</td>
<td>PMLc.MatLow.RawMaterial</td>
</tr>
<tr>
<td>CO2</td>
<td>PMLc.MatLow.CO2</td>
</tr>
<tr>
<td>Container</td>
<td>PMLc.MatLow.Container</td>
</tr>
<tr>
<td>Lubrication</td>
<td>PMLc.MatLow.Lubrication</td>
</tr>
<tr>
<td>Water</td>
<td>PMLc.MatLow.Water</td>
</tr>
<tr>
<td>ContainerClosures</td>
<td>PMLc.MatLow.ContainerClosures</td>
</tr>
<tr>
<td>Unused0</td>
<td>PMLc.MatLow.Unused0</td>
</tr>
<tr>
<td>Unused1</td>
<td>PMLc.MatLow.Unused1</td>
</tr>
<tr>
<td>Unused2</td>
<td>PMLc.MatLow.Unused2</td>
</tr>
<tr>
<td>Unused3</td>
<td>PMLc.MatLow.Unused3</td>
</tr>
<tr>
<td>Unused4</td>
<td>PMLc.MatLow.Unused4</td>
</tr>
<tr>
<td>Unused5</td>
<td>PMLc.MatLow.Unused5</td>
</tr>
<tr>
<td>Unused6</td>
<td>PMLc.MatLow.Unused6</td>
</tr>
<tr>
<td>Unused7</td>
<td>PMLc.MatLow.Unused7</td>
</tr>
<tr>
<td>Unused8</td>
<td>PMLc.MatLow.Unused8</td>
</tr>
<tr>
<td>Unused9</td>
<td>PMLc.MatLow.Unused9</td>
</tr>
<tr>
<td>State</td>
<td>PMLc.State</td>
</tr>
<tr>
<td>StateChangeRequest</td>
<td>PMLc.StateChangeRequest</td>
</tr>
<tr>
<td>CntrlCmd</td>
<td>PMLc.CntrlCmd</td>
</tr>
<tr>
<td>Node[#]</td>
<td>PMLc.Node[#]</td>
</tr>
<tr>
<td>Number</td>
<td>PMLc.Node [#].Number</td>
</tr>
<tr>
<td>ControlCmdNumber</td>
<td>PMLc.Node[#].ControlCmdNumber</td>
</tr>
<tr>
<td>CmdValue</td>
<td>PMLc.Node[#].CmdValue</td>
</tr>
<tr>
<td>Parameter[#]</td>
<td>PMLc.Node[#].Parameter[#]</td>
</tr>
<tr>
<td>Id</td>
<td>PMLc.Node[#].Parameter[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLc.Node[#].Parameter[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLc.Node[#].Parameter[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLc.Node[#].Parameter[#].Value</td>
</tr>
<tr>
<td>ProcessVariables[#]</td>
<td>PMLc.ProcessVariables[#]</td>
</tr>
<tr>
<td>Id</td>
<td>PMLc.ProcessVariables[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLc.ProcessVariables[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLc.ProcessVariables[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLc.ProcessVariables[#].Value</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Product[#]</td>
<td>PMLc.Product[#]</td>
</tr>
<tr>
<td>ProductId</td>
<td>PMLc.Product[#].ProductId</td>
</tr>
<tr>
<td>ProcessVariables[#]</td>
<td>PMLc.Product[#].ProcessVariables[#]</td>
</tr>
<tr>
<td>Id</td>
<td>PMLc.Product[#].ProcessVariables[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLc.Product[#].ProcessVariables[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLc.Product[#].ProcessVariables[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLc.Product[#].ProcessVariables[#].Value</td>
</tr>
<tr>
<td>Ingredients[#]</td>
<td>PMLc.Product[#].Ingredients[#]</td>
</tr>
<tr>
<td>IngredientId</td>
<td>PMLc.Product[#].Ingredients[#].IngredientId</td>
</tr>
<tr>
<td>Parameter[#]</td>
<td>PMLc.Product[#].Ingredients[#].Parameter[#]</td>
</tr>
<tr>
<td>Id</td>
<td>PMLc.Product[#].Ingredients[#].Parameter[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLc.Product[#].Ingredients[#].Parameter[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLc.Product[#].Ingredients[#].Parameter[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLc.Product[#].Ingredients[#].Parameter[#].Value</td>
</tr>
<tr>
<td>Limits[#]</td>
<td>PMLc.Limits[#]</td>
</tr>
<tr>
<td>Id</td>
<td>PMLc.Limits[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLc.Limits[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLc.Limits[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLc.Limits[#].Value</td>
</tr>
<tr>
<td>TargetDownstreamNodeId</td>
<td>PMLc.TargetDownstreamNodeId</td>
</tr>
<tr>
<td>TargetUpstreamNodeId</td>
<td>PMLc.TargetUpstreamNodeId</td>
</tr>
<tr>
<td>ChangeNodeServicedUpstream</td>
<td>PMLc.ChangeNodeServicedUpstream</td>
</tr>
<tr>
<td>ChangeNodeServicedDownstream</td>
<td>PMLc.ChangeNodeServicedDownstream</td>
</tr>
</tbody>
</table>
Status structure PMLs
<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommandRejected</td>
<td>PMLs.CommandRejected</td>
</tr>
<tr>
<td>UnitModeCurrent</td>
<td>PMLs.UnitModeCurrent</td>
</tr>
<tr>
<td>UnitModeRequested</td>
<td>PMLs.UnitModeRequested</td>
</tr>
<tr>
<td>UnitModeChangeInProcess</td>
<td>PMLs.UnitModeChangeInProcess</td>
</tr>
<tr>
<td>ProcModeCurrent</td>
<td>PMLs.ProcModeCurrent</td>
</tr>
<tr>
<td>ProcModeRequested</td>
<td>PMLs.ProcModeRequested</td>
</tr>
<tr>
<td>ProcModeChangeInProcess</td>
<td>PMLs.ProcModeChangeInProcess</td>
</tr>
<tr>
<td>StateCurrent</td>
<td>PMLs.StateCurrent</td>
</tr>
<tr>
<td>StateRequested</td>
<td>PMLs.StateRequested</td>
</tr>
<tr>
<td>StateChangeInProcess</td>
<td>PMLs.StateChangeInProcess</td>
</tr>
<tr>
<td>StateChangeProgress</td>
<td>Status.StateChangeProgress</td>
</tr>
<tr>
<td>StateLastCompleted</td>
<td>PMLs.StateLastCompleted</td>
</tr>
<tr>
<td>SeqNumber</td>
<td>PMLs.SeqNumber</td>
</tr>
<tr>
<td>CurMachSpd</td>
<td>PMLs.CurMachSpd</td>
</tr>
<tr>
<td>MatReady</td>
<td>PMLs.MatReady</td>
</tr>
<tr>
<td>RawMaterial</td>
<td>PMLs.MatReady.RawMaterial</td>
</tr>
<tr>
<td>CO2</td>
<td>PMLs.MatReady.CO2</td>
</tr>
<tr>
<td>Container</td>
<td>PMLs.MatReady.Container</td>
</tr>
<tr>
<td>Lubrication</td>
<td>PMLs.MatReady.Lubrication</td>
</tr>
<tr>
<td>Water</td>
<td>PMLs.MatReady.Water</td>
</tr>
<tr>
<td>ContainerClosures</td>
<td>PMLs.MatReady.ContainerClosures</td>
</tr>
<tr>
<td>Unused0</td>
<td>PMLs.MatReady.Unused0</td>
</tr>
<tr>
<td>Unused1</td>
<td>PMLs.MatReady.Unused1</td>
</tr>
<tr>
<td>Unused2</td>
<td>PMLs.MatReady.Unused2</td>
</tr>
<tr>
<td>Unused3</td>
<td>PMLs.MatReady.Unused3</td>
</tr>
<tr>
<td>Unused4</td>
<td>PMLs.MatReady.Unused4</td>
</tr>
<tr>
<td>Unused5</td>
<td>PMLs.MatReady.Unused5</td>
</tr>
<tr>
<td>Unused6</td>
<td>PMLs.MatReady.Unused6</td>
</tr>
<tr>
<td>Unused7</td>
<td>PMLs.MatReady.Unused7</td>
</tr>
<tr>
<td>Unused8</td>
<td>PMLs.MatReady.Unused8</td>
</tr>
<tr>
<td>Unused9</td>
<td>PMLs.MatReady.Unused9</td>
</tr>
<tr>
<td>MatLow</td>
<td>PMLs.MatLow</td>
</tr>
<tr>
<td>RawMaterial</td>
<td>PMLs.MatLow.RawMaterial</td>
</tr>
<tr>
<td>CO2</td>
<td>PMLs.MatLow.CO2</td>
</tr>
<tr>
<td>Container</td>
<td>PMLs.MatLow.Container</td>
</tr>
<tr>
<td>Lubrication</td>
<td>PMLs.MatLow.Lubrication</td>
</tr>
<tr>
<td>Water</td>
<td>PMLs.MatLow.Water</td>
</tr>
<tr>
<td>ContainerClosures</td>
<td>PMLs.MatLow.ContainerClosures</td>
</tr>
<tr>
<td>Unused0</td>
<td>PMLs.MatLow.Unused0</td>
</tr>
<tr>
<td>Unused1</td>
<td>PMLs.MatLow.Unused1</td>
</tr>
<tr>
<td>Unused2</td>
<td>PMLs.MatLow.Unused2</td>
</tr>
<tr>
<td>Unused3</td>
<td>PMLs.MatLow.Unused3</td>
</tr>
<tr>
<td>Unused4</td>
<td>PMLs.MatLow.Unused4</td>
</tr>
<tr>
<td>Unused5</td>
<td>PMLs.MatLow.Unused5</td>
</tr>
<tr>
<td>Unused6</td>
<td>PMLs.MatLow.Unused6</td>
</tr>
<tr>
<td>Unused7</td>
<td>PMLs.MatLow.Unused7</td>
</tr>
<tr>
<td>Unused8</td>
<td>PMLs.MatLow.Unused8</td>
</tr>
<tr>
<td>Unused9</td>
<td>PMLs.MatLow.Unused9</td>
</tr>
<tr>
<td>MachDesignSpeed</td>
<td>PMLs.MachDesignSpeed</td>
</tr>
<tr>
<td>MachCycle</td>
<td>PMLs.MachCycle</td>
</tr>
<tr>
<td>ProdRatio</td>
<td>PMLs.ProdRatio</td>
</tr>
<tr>
<td>Dirty</td>
<td>PMLs.Dirty</td>
</tr>
<tr>
<td>Clean</td>
<td>PMLs.Clean</td>
</tr>
<tr>
<td>TimeToDirty</td>
<td>PMLs.TimeToDirty</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EquipmentAllocatedToUnitModelID</td>
<td>PMLs.EquipmentAllocatedToUnitModelID</td>
</tr>
<tr>
<td>MachineReusableForUnitModelID</td>
<td>PMLs.MachineReusableForUnitModelID</td>
</tr>
<tr>
<td>MachineReusableTimeLeft</td>
<td>PMLs.MachineReusableTimeLeft</td>
</tr>
<tr>
<td>MachineStoringProductId</td>
<td>PMLs.MachineStoringProductId</td>
</tr>
<tr>
<td>MachineTransferringProductID</td>
<td>PMLs.MachineTransferringProductID</td>
</tr>
<tr>
<td>Node[#]</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>PMLs.Node[#].Number</td>
</tr>
<tr>
<td>ControlCmdNumber</td>
<td>PMLs.Node[#].ControlCmdNumber</td>
</tr>
<tr>
<td>CmdValue</td>
<td>PMLs.Node[#].CmdValue</td>
</tr>
<tr>
<td>Parameter[#]</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>PMLs.Node[#].Parameter[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLs.Node[#].Parameter[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLs.Node[#].Parameter[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLs.Node[#].Parameter[#].Value</td>
</tr>
<tr>
<td>ProcessVariables[#]</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>PMLs.ProcessVariables[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLs.ProcessVariables[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLs.ProcessVariables[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLs.ProcessVariables[#].Value</td>
</tr>
<tr>
<td>Product[#]</td>
<td></td>
</tr>
<tr>
<td>ProductId</td>
<td>PMLs.Product[#].ProductId</td>
</tr>
<tr>
<td>ProcessVariables[#]</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>PMLs.Product[#].ProcessVariables[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLs.Product[#].ProcessVariables[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLs.Product[#].ProcessVariables[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLs.Product[#].ProcessVariables[#].Value</td>
</tr>
<tr>
<td>Ingredients[#]</td>
<td></td>
</tr>
<tr>
<td>IngredientId</td>
<td>PMLs.Product[#].Ingredients[#].IngredientId</td>
</tr>
<tr>
<td>Parameter[#]</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>PMLs.Product[#].Ingredients[#].Parameter[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLs.Product[#].Ingredients[#].Parameter[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLs.Product[#].Ingredients[#].Parameter[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLs.Product[#].Ingredients[#].Parameter[#].Value</td>
</tr>
<tr>
<td>Limits[#]</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>PMLs.Limits[#].Id</td>
</tr>
<tr>
<td>Name</td>
<td>PMLs.Limits[#].Name</td>
</tr>
<tr>
<td>Unit</td>
<td>PMLs.Limits[#].Unit</td>
</tr>
<tr>
<td>Value</td>
<td>PMLs.Limits[#].Value</td>
</tr>
</tbody>
</table>
Admin structure PMLa

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin</td>
<td>ST_PMLa</td>
</tr>
<tr>
<td>PMLa.Alarm[#]</td>
<td>ST_Alarm</td>
</tr>
<tr>
<td>PMLa.Alarm[#].Id</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].Value</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].Message</td>
<td>STRING</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeEvent</td>
<td>ST_Time-Stamp</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeEvent.Year</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeEvent.Month</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeEvent.Day</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeEvent.Hour</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeEvent.Minute</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeEvent.Second</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeEvent.mSec</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeAck</td>
<td>ST_Time-Stamp</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeAck.Year</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeAck.Month</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeAck.Day</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeAck.Hour</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeAck.Minute</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeAck.Second</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].TimeAck.mSec</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].ModeCurrentTime[#]</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].ModeCumulativeTime[#]</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].StateCurrentTime[#,#]</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].StateCumulativeTime[#,#]</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].ProdProcessed</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].DefectProd</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].ReworkProd</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].ResetTimersCounters</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].UpstreamMessage</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].DownstreamMessage</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].CurrentDownstreamNodeId[#]</td>
<td>DINT</td>
</tr>
<tr>
<td>PMLa.Alarm[#].CurrentUpstreamNodeId[#]</td>
<td>DINT</td>
</tr>
</tbody>
</table>

4.4 DataTypes

4.4.1 Alarm

4.4.1.1 E_AlarmID

The data type E_AlarmID defines the error cause.

```plaintext
TYPE E_AlarmID : ( 
    eAID_Undefined := 0,
    (** 1..32 Machine internal reason - safeties - OMAC defined ***)
    eAID_EStop_Pushed := 1,
    eAID_PerimeterProtectionFault,
    eAID_MainsTurnedOff,
    eAID_SafetyGateOrGuardDoorOpen,
    (* 5..32 reserved for future OMAC safety codes *)
```
(** Machine internal reason - operator actions - OMAC defined *)
eAID_CycleStopButtonPushed := 33,
eAID_StartButtonPushed,
eAID_ResetButtonPushed,
eAID_JogModeSelected,
eAID_AutomaticModeSelected,
eAID_ManualModeSelected,
eAID_SemiAutomaticModeSelected,
(* 40..64 reserved for future OMAC defined operator action codes *)

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

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

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

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

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

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

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

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

(** 4000..4499 out of service - OMAC defined *)
eAID_LineNotScheduled := 4000,
eAID_PlannedMaintenance,
eAID_MealAndRest,
eAID_Meetings,
eAID_Training,
Packaging Machine Functional Tag Description

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

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

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

**4.4.1.2 ST_Alarm**

This is the collection tags needed to describe alarm events.

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

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

**4.4.1.3 ST_TimeStamp**

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

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

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>
4.4.2 Common

4.4.2.1 E_CntrlCmd

Enumeration of possible commands for the machine state

```plaintext
TYPE E_CntrlCmd :
{
    eCntrlCmd_UNDEFINED,
    eCntrlCmd_RESET,
    eCntrlCmd_START,
    eCntrlCmd_STOP,
    eCntrlCmd_HOLD,
    eCntrlCmd_UNHOLD,
    eCntrlCmd_SUSPEND,
    eCntrlCmd_UNSUSPEND,
    eCntrlCmd_ABORT,
    eCntrlCmd_CLEAR,
    eCntrlCmd_COMPLETE,
};
END_TYPE
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

4.4.2.2 E_CurMachSpd

Enumeration to define different machine speeds

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

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

4.4.2.3 ST_Descriptor

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

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

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

#### 4.4.2.4 ST_Ingedit

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

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

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

#### 4.4.2.5 ST_Material

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

```plaintext
TYPE ST_Materials :
  STRUCT
    RawMaterial : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    CO2 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Container : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Lubrication : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Water : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    ContainerClosures : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    (* 10 more unused *)
    Unused0 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused1 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused2 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused3 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused4 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused5 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused6 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused7 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused8 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Unused9 : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
  END_STRUCT
END_TYPE
```

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

#### 4.4.2.6 ST_Node

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

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

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

### 4.4.2.7 ST_Product

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

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

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

### 4.4.3 ST_PMLa

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

```plaintext
TYPE ST_PMLa :
STRUCT
  Alarm : ARRAY [1..iMAX_ALARMS] OF ST_Alarm;(* ~ (OPC : 1: enabled for OPC ) *)
  ModeCurrentTime : ARRAY [1..iMAX_CURRENT_MODE] OF DINT;(* ~ (OPC : 1: enabled for OPC )*)
  ModeCummulativeTime : ARRAY [1..iMAX_CUMMULATIVE_MODE] OF DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  StateCurrentTime : ARRAY [1..iMAX_CURRENT_MODE, 0..iMAX_CURRENT_STATE] OF DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  StateCummulativeTime : ARRAY [1..iMAX_CURRENT_MODE, 0..iMAX_CURRENT_STATE] OF DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  ProdProcessed : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  DefectProd : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  ReworkProd : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  ResetTimersCounters : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  UpstreamMessage : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  DownstreamMessage : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  CurrentDownstreamNodeID : ARRAY [1..iMAX_CURR_NODE_IDS] OF DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  CurrentUpstreamNodeID : ARRAY [1..iMAX_CURR_NODE_IDS] OF DINT;(* ~ (OPC : 1: enabled for OPC ) *)
END_STRUCT
END_TYPE
```

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>
### 4.4.4 ST_PMLc

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

```
TYPE ST_PMLc :
  STRUCT
    UnitMode             : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    UnitModeChangeRequest: BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcMode             : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcModeChangeRequest: BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    CurMachSpeed         : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    MatReady             : ST_Materials;(* ~ (OPC : 1: enabled for OPC ) *)
    MatLow               : ST_Materials;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcMode             : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcModeChangeRequest: BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    CurMachSpeed         : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    State                : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    StateChangeRequest   : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    CntrlCmd             : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Node                 : ARRAY [1..iMAX_NODES] OF ST_Node;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcessVariables     : ARRAY [1..iMAX_PROCESS_VARS] OF ST_Descriptor;(* ~ (OPC : 1: enabled for OPC ) *)
    Product              : ARRAY [1..iMAX_PRODUCTS] OF ST_Product;(* ~ (OPC : 1: enabled for OPC ) *)
    Limits               : ARRAY [1..iMAX_LIMITS] OF ST_Descriptor;(* ~ (OPC : 1: enabled for OPC ) *)
    TargetDownstreamNodeID : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    TargetUpstreamNodeID  : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ChangeNodeServicedUpstream : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ChangeNodeServicedDownstream : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
  END_STRUCT
END_TYPE
```

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

### 4.4.5 ST_PMLs

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

```
TYPE ST_PMLs :
  STRUCT
    CommandRejected      : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    UnitModeCurrent      : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    UnitModeRequested    : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcModeCurrent      : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcModeRequested    : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProcModeChangeInProcess : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    StateCurrent         : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    StateRequested       : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    StateChangeInProcess : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    SeqNumber            : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    CurMachSpd           : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    MatReady             : ST_Materials;(* ~ (OPC : 1: enabled for OPC ) *)
    MatLow               : ST_Materials;(* ~ (OPC : 1: enabled for OPC ) *)
    MachDesignSpeed      : REAL;(* ~ (OPC : 1: enabled for OPC ) *)
    MachCycle            : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    ProdRatio            : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    Dirty                : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    Clean                : BOOL;(* ~ (OPC : 1: enabled for OPC ) *)
    TimeToDirty          : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    EquipmentAllocatedToUnitModelID : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    MachineReusableForUnitModelID : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
    MachineReusableTimeLeft : DINT;(* ~ (OPC : 1: enabled for OPC ) *)
```
MachineStoringProductID : DINT;(* (OPC : 1: enabled for OPC ) *)
MachineTransferringProductID : DINT;(* (OPC : 1: enabled for OPC ) *)

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

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>

**4.5 Parameter List**

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

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

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

**4.6 Global Constants**

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

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

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1, build 4018 onwards</td>
<td>PC (i386)</td>
<td>Tc3_PackML</td>
</tr>
</tbody>
</table>
More Information:
www.beckhoff.com/te1000