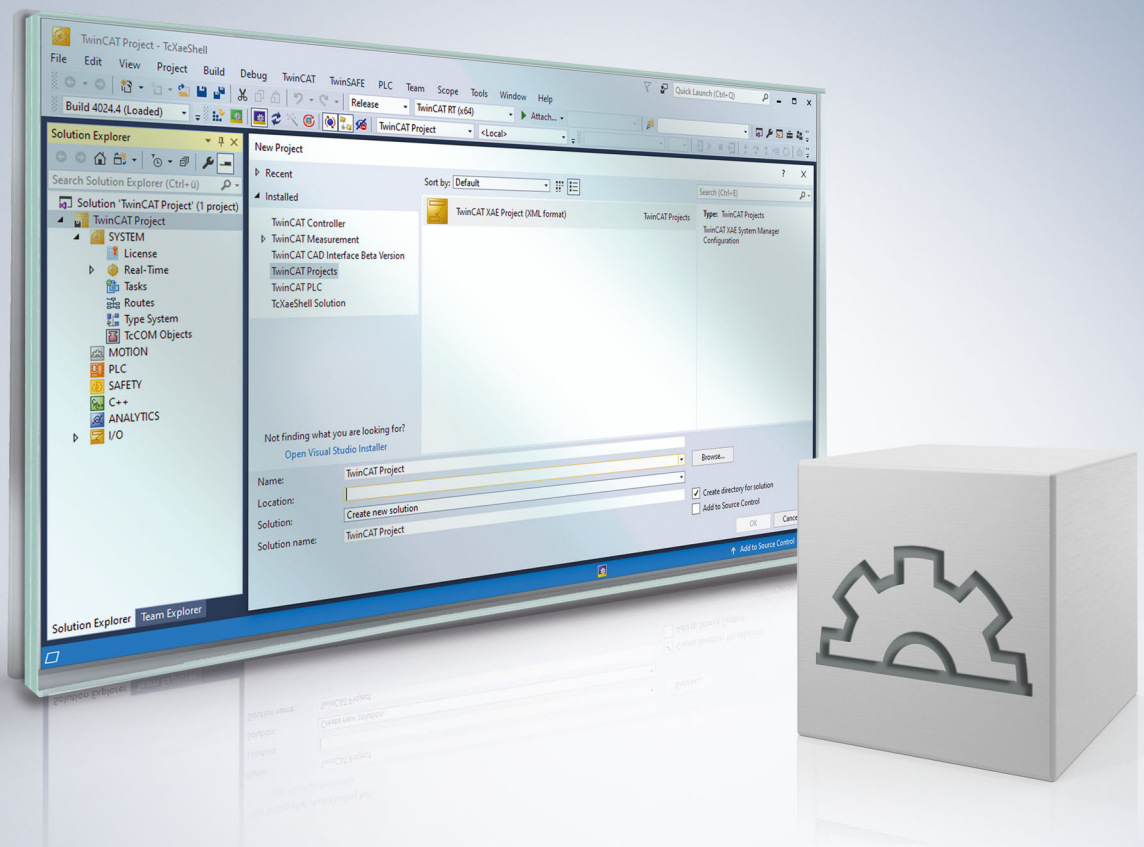
BECKHOFF New Automation Technology

Handbuch | DE

# TE1000

## TwinCAT 3 | PLC-Bibliothek: Tc3_PackML_V2




28.09.2023 | Version: 1.2.3

# Inhaltsverzeichnis

# 1 Vorwort

## 1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

### Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

### Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

### Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

### Copyright

## 1.2 Zu Ihrer Sicherheit

### Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

### Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

### Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

### Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

### Warnungen vor Personenschäden

**⚠ GEFAHR**

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

**⚠ WARNUNG**

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

**⚠ VORSICHT**

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

### Warnung vor Umwelt- oder Sachschäden

***HINWEIS***

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

### Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

## 1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

# 2 Packaging Machine State

Die „Packaging Machine State“-Funktionsbausteine haben eine gemeinsame Schnittstelle zu den existierenden „PackML Machine State Model“-Ausführungen.

Es wird erwartet, dass anwendungsspezifische Logik, wie Zustandsübergänge, in externen Funktionsbausteinen programmiert ist und der „Packaging Machine State“-Funktionsbaustein die zentrale Logik der Zustandsmaschine und die Zustandsdarstellung übernimmt. Deswegen gibt es für diesen Funktionsbaustein eine Empfehlung, wie er mit anderer Logik kombiniert werden kann.

Der Zustandsübergang in einer Maschinenanwendung ist immer anwendungsspezifisch. Deswegen gestalten Sie am besten mit PackML State Machine V3 verknüpfte „State“-Funktionsbausteine, um die Standardisierung zu vereinfachen. Die „State“-Funktionsbausteine erfassen anwendungsspezifische Signale und stellen die Übergangslogik zu angrenzenden Zuständen dar (siehe PackML Zustandsmodell). Die „State“-Funktionsbausteine geben Feedback an PS_PackML_State_Machine_V3, wodurch eine Standard-Zustandsmaschine und Zustandsmeldung möglich wird. Die „State“-Funktionsbausteine enthalten den Maschinenausführungscode und die anwendungsspezifische Übergangslogik.

„State“-Funktionsbausteine sind unten gelistet und werden vom Anwendungsprogrammierer so programmiert, dass Integrität und Funktionalität der PackML State Machine gewahrt bleiben.

**Namen der „PackML State Machine V3“-Funktionsbausteine:**

- 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

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.1 Interfaces

### 2.1.1 I_UnitState

Dieses Interface kann in den Unit-Bausteinen der Applikation implementiert werden und stellt alle Methoden des Packaging State Models zur Verfügung, die dann je nach Bedarf mit Applikations-Code gefüllt werden können.

Diese Methoden sind:

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

Dieses Interface kann in den Unit-Bausteinen der Applikation implementiert werden und stellt nur die „Waiting“-Methoden des Packaging State Models zur Verfügung, die dann je nach Bedarf mit Applikations-Code gefüllt werden können.

Diese Methoden sind:

M_Aborted

M_Complete

M_Held

M_Idle

M_Stopped

M_Suspended

M_Undefined

### 2.1.3 I_UnitStateActing

Dieses Interface kann in den Unit-Bausteinen der Applikation implementiert werden und stellt nur die „Acting“-Methoden des Packaging State Models zur Verfügung, die dann je nach Bedarf mit Applikations-Code gefüllt werden können.

Diese Methoden sind:

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 Datentypen

## 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
```

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | 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
```

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | 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
```

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | 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 | Adresse eines eindimensionalen Arrays vom Typ ST_PMLSubUnitInfo. Jedes Array-Element enthält den Zustand eines unterlagerten Maschinenteils.<br>Beispiel:<br>stSubUnitInfo : ARRAY[1..10] OF ST_PMLSubUnitInfo;<br>pArray := ADR(stSubUnitInfo); |
| nArraySize | Speichergröße des eindimensionalen Arrays, die mit der SIZEOF-Funktion ermittelt werden kann.<br>Beispiel:<br>nArraySize := SIZEOF(stSubUnitInfo); |
| nNoOfSubUnits | Anzahl der relevanten unterlagerten Maschinenteile. |

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | 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 | Signalisiert, dass dieser unterlagerte Maschinenteil aktiv ist und den Zustandsvorgaben der Zustandsmaschine folgt. |
| eState | Enumeration, die den aktuellen Zustand des unterlagerten Maschinenteils wiedergibt. |

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.2.6 ST_PMLStateMachineOptions

### ST_PMLStateMachineOptions

```
TYPE ST_PMLStateMachineOptions :
STRUCT
END_STRUCT
END_TYPE
```

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | 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 | Wird dieses Flag TRUE gesetzt, so wird anstelle der Zeitinformation des Systems nun die am Eingang ExternalPackMLTime vorgegebene Zeit verwendet. |
| ExternalPackMLTime | Extern vorgegebene Zeit |

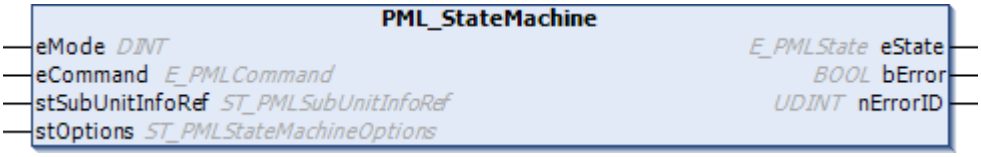
### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

# 2.3 Funktionsbausteine

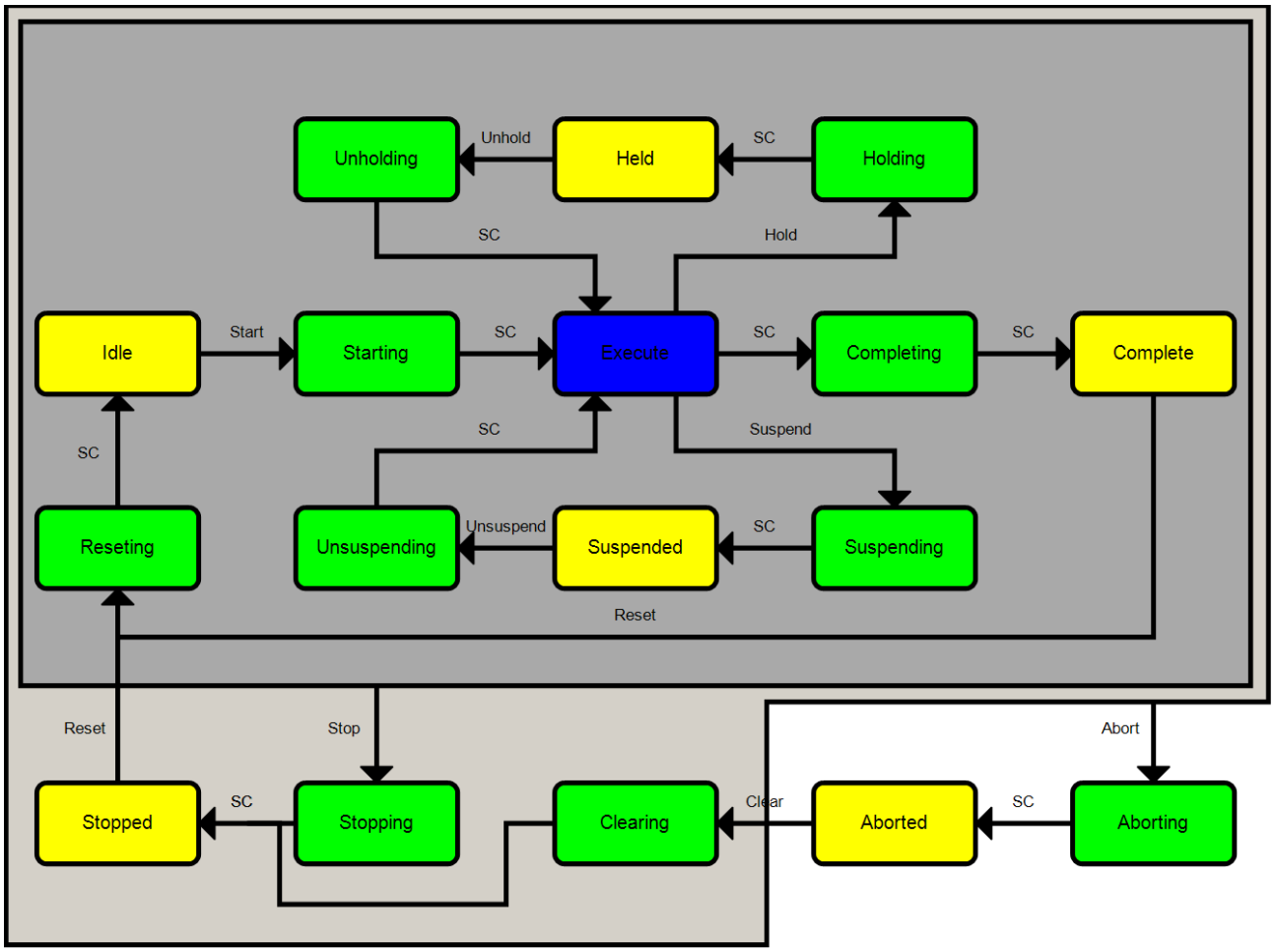
## 2.3.1 Packaging Machine State

### 2.3.1.1 PML_StateMachine

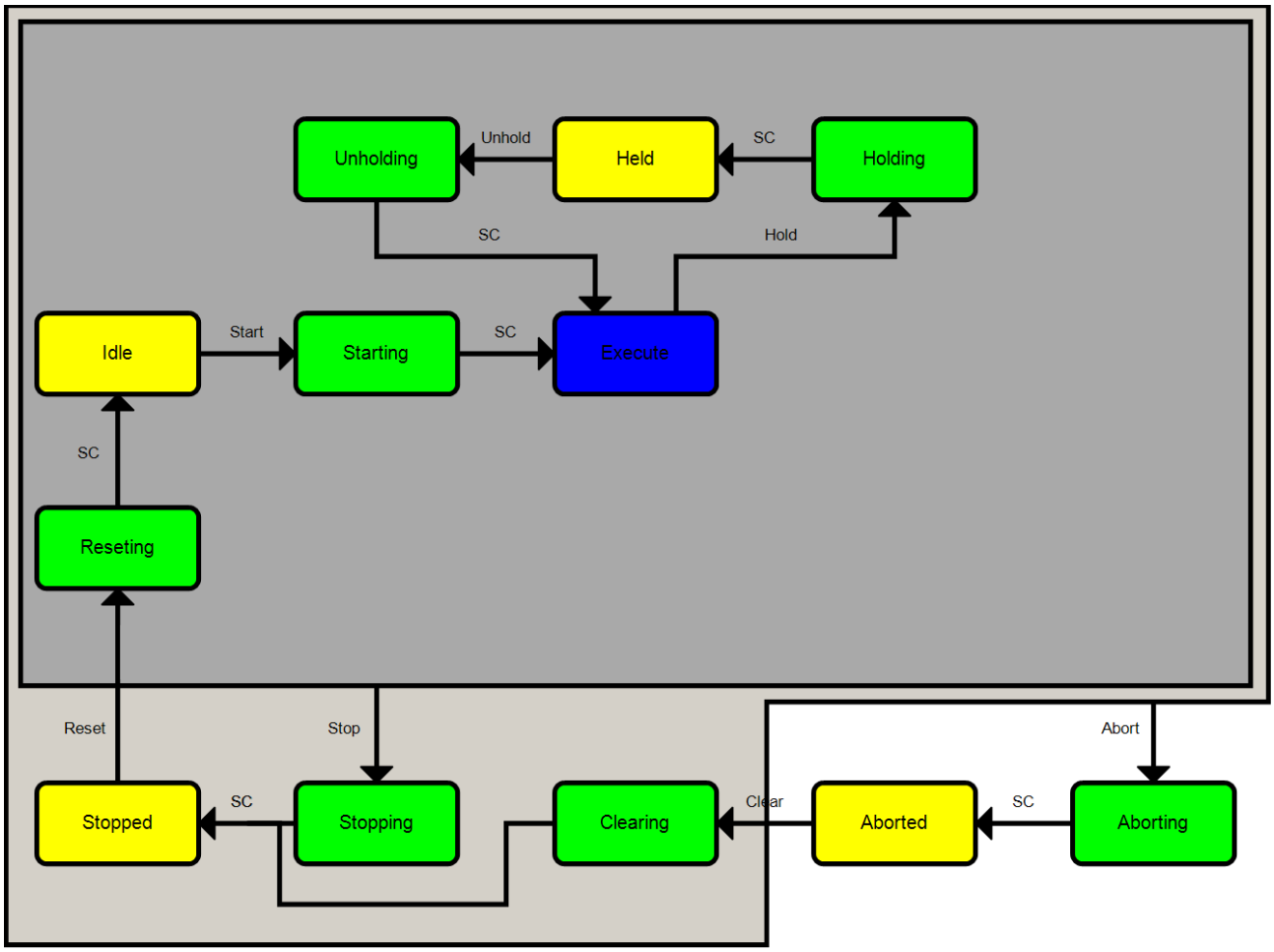



Der PML_StateMachine-Funktionsbaustein hat in der aktualisierten Form eine gemeinsame Schnittstelle zum PackML Machine State Model V3. Es wird erwartet, dass anwendungsspezifische Logik, wie Zustandsübergänge, in externen Funktionsbausteinen programmiert wird und der PML_StateMachine-Funktionsbaustein die zentrale Logik der Zustandsmaschine und die Zustandsdarstellung übernimmt. Aufgrund des aktuell aktiven UnitMode (eMode) stellt sich das Machine State Model unterschiedlich dar. Dazu sind drei Grundmodelle vorkonfiguriert (E_PMLProtectedUnitMode [▶ 11]).

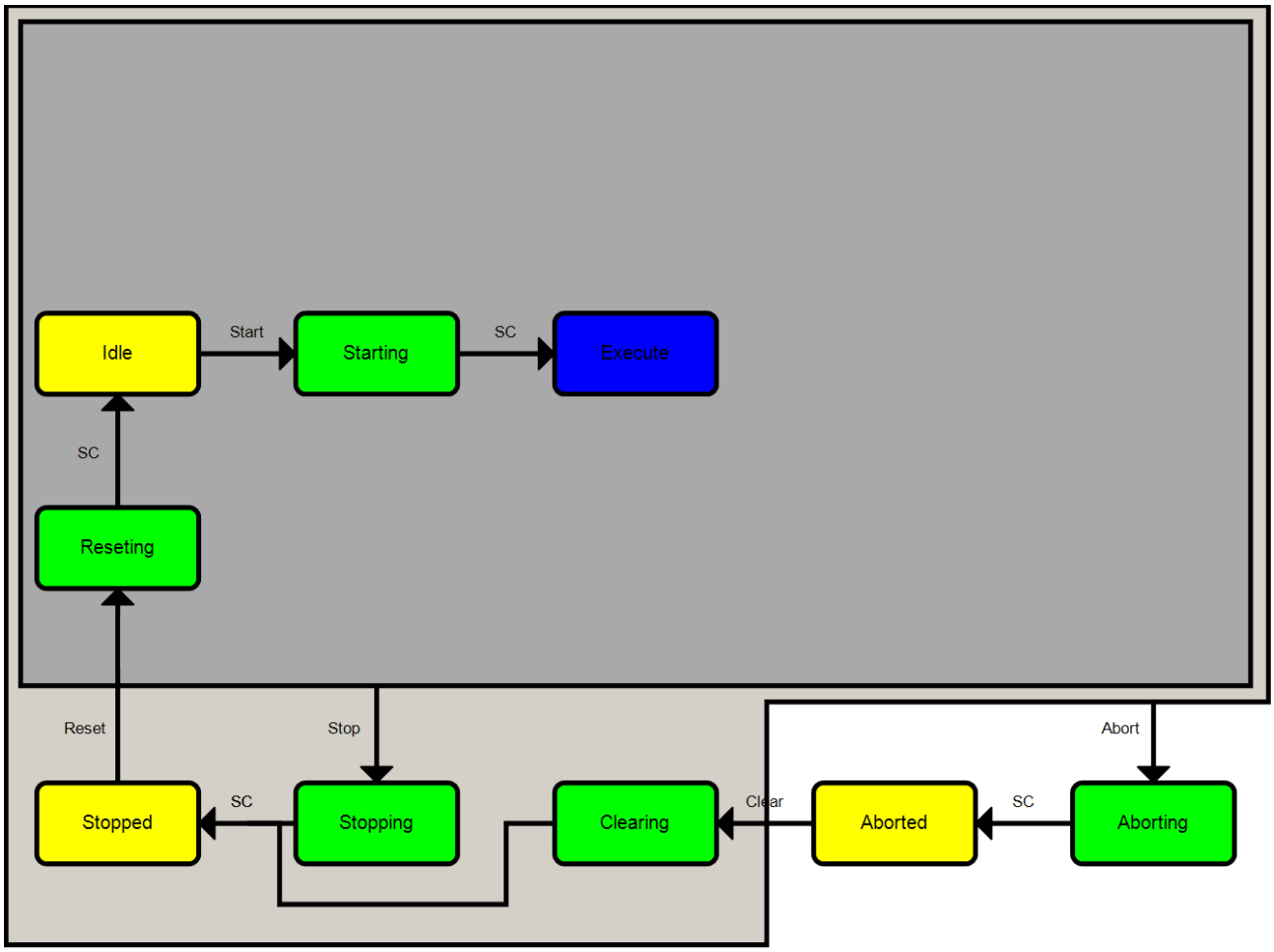
**ePMLProtUnitMode_Production**



**ePMLProtUnitMode_Maintenance**



**ePMLProtUnitMode_Manual**




Weiterhin können weitere anwenderspezifische Modelle in einfacher Weise mit Hilfe des Funktionsbaustein PML_UnitModeConfig [▶ 17] erstellt werden und sind so sehr flexibel einsetzbar.

Die Logik für Übergänge, insbesondere zwischen Production, Maintenance und Manual Mode, ist abhängig von der Anwendung. In welchen Zuständen UnitMode-Wechsel für die Grundmodelle zulässig sind, ist in der Beschreibung des Funktionsbaustein PML_UnitModeManager [▶ 20] genauer beschrieben.

### Eingänge

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

| Name | Typ | Beschreibung |
|---|---|---|
| eMode | DINT | Aktueller PML-UnitMode |
| eCommand | E_PMLCommand | Enumeration [▶ 10] mit den verschiedenen PML-Kommandos des Bausteins. |
| stSubUnitInfoRef | ST_PMLSubUnitInfoRef | Struktur [▶ 12] zum Verweis auf ein Array der aktuellen PML-States von unterlagerten Maschineneinheiten |
| stOptions | ST_PMLStateMachineOptions | Momentan nicht verwendet |

### Ausgänge

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

| Name | Typ | Beschreibung |
|---|---|---|
| eState | E_PMLState | Enumeration [▶ 10], die den aktuellen PML-Zustand der automatischen Zustandsmaschine liefert. |
| bError | BOOL | Wird TRUE, sobald ein Fehler eintritt. |
| nErrorId | UDINT | Liefert bei einem gesetzten bError-Ausgang die Fehlernummer. |

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.1.2 PML_UnitModeConfig


Maschinen haben unter Umständen noch andere Anlagenmodi als „Production“, „Maintenance“ und „Manual“. Dieser Baustein ermöglicht es dem Anwender weitere Modelle (UnitModes) zu konfigurieren.

Dabei können die Nummer des neuen Modells, die vorhandenen Zustände und die Zustände, in denen ein Modell-Wechsel möglich ist, frei definiert werden.

### Eingänge

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

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

| Name | Typ | Beschreibung |
|---|---|---|
| eMode | DINT | Nummer der neuen PML-UnitMode [4..31] |
| sName | STRING | Name des neuen PML-UnitMode |
| bDisableClearing | BOOL | Deaktiviert den PMLState „Clearing“. |
| bDisableStarting | BOOL | Deaktiviert den PMLState „Starting“. |
| bDisableSuspended | BOOL | Deaktiviert den PMLState „Suspended“.<br>Durch das Deaktivieren des statischen Zustandes werden die PMLState „Suspending“ und „Unsuspending“ ebenfalls deaktiviert. |
| bDisableStopping | BOOL | Deaktiviert den PMLState „Stopping“. |
| bDisableAborting | BOOL | Deaktiviert den PMLState „Aborting“. |
| bDisableHolding | BOOL | Deaktiviert den PMLState „Holding“. |
| bDisableHeld | BOOL | Deaktiviert den PMLState „Held“.<br>Durch das Deaktivieren des statischen Zustandes werden die PMLState „Holding“ und „Unholding“ ebenfalls deaktiviert. |
| bDisableUnholding | BOOL | Deaktiviert den PMLState „Unholding“. |
| bDisableSuspending | BOOL | Deaktiviert den PMLState „Suspending“. |
| bDisableUnsuspending | BOOL | Deaktiviert den PMLState „Unsuspending“. |
| bDisableResetting | BOOL | Deaktiviert den PMLState „Resettinging“. |
| bDisableIdle | BOOL | Deaktiviert den PMLState „Idle“. Durch das Deaktivieren des statischen Zustandes wird der PMLState „Resetting“ ebenfalls deaktiviert. |
| bDisableCompleting | BOOL | Deaktiviert den PMLState „Completing“. |
| bDisableComplete | BOOL | Deaktiviert den PMLState „Complete“. Durch das Deaktivieren des statischen Zustandes wird der PMLState „Completing“ ebenfalls deaktiviert. |
| bEnableUnitModeChangeStopped | BOOL | Gibt einen Modus-Wechsel im PMLState „Stopped“ frei. |
| bEnableUnitModeChangeIdle | BOOL | Gibt einen Modus-Wechsel im PMLState „Idle“ frei. |
| bEnableUnitModeChangeSuspended | BOOL | Gibt einen Modus-Wechsel im PMLState „Suspended“ frei. |
| bEnableUnitModeChangeExecute | BOOL | Gibt einen Modus-Wechsel im PMLState „Execute“ frei. |
| bEnableUnitModeChangeAborted | BOOL | Gibt einen Modus-Wechsel im PMLState „Aborted“ frei. |
| bEnableUnitModeChangeHeld | BOOL | Gibt einen Modus-Wechsel im PMLState „Held“ frei. |
| bEnableUnitModeChangeComplete | BOOL | Gibt einen Modus-Wechsel im PMLState „Complete“ frei. |

### Ausgänge

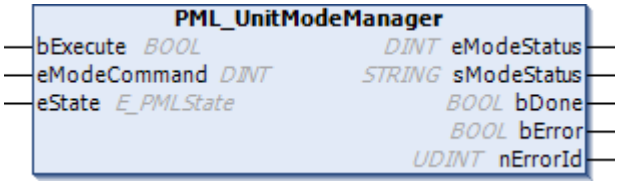
```
VAR_OUTPUT
    bError      : BOOL;
    nErrorID    : UDINT;
END_VAR
```

| Name | Typ | Beschreibung |
|---|---|---|
| bError | BOOL | Wird TRUE, sobald ein Fehler eintritt. |
| nErrorId | UDINT | Liefert bei einem gesetzten bError-Ausgang die Fehlernummer. |

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 2.3.1.3 PML_UnitModeManager

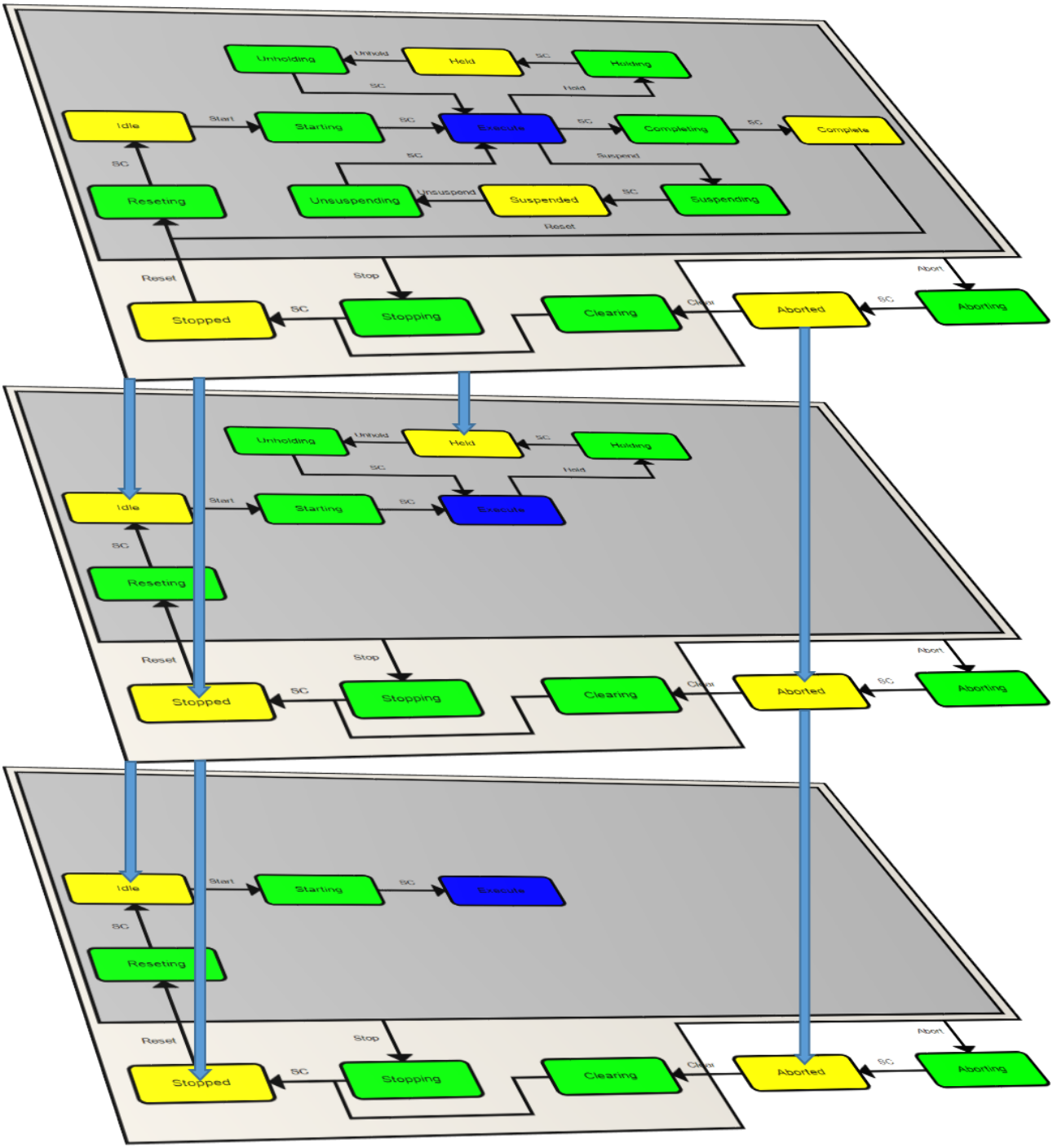



Maschinen haben andere Anlagenmodi als „Production“. Jeder Anlagenmodus ist durch ein eigenes Zustandsmodell definiert. Für Übergänge zwischen den Modi muss ein „Mode Manager“ definiert werden. Der „Mode Manager“ entscheidet, wie und in welchen Zustand eine Maschine Anlagenmodi ändern kann, d.h. eingebaute Sperren verhindern, dass die Maschine in ungeeignete Zustände wechselt. Für die Grundmodi „Production“, „Maintenance“ und „Manual“ sind diese Sperren fest definiert, wie die Darstellung unten zeigt. Für weitere über den PML_UnitModeConfig [▶ 17]-Baustein definierte Modi kann dies individuell festgelegt werden.

**⚠ WARNUNG**

**Ordnungsgemäße Modusänderungen einhalten**

Die Logik für Übergänge zwischen den Modi ist abhängig von der Anwendung, insbesondere für Übergänge zwischen „Manual“- und „Production“-Modus. Für solche Modusänderungen können darüber hinaus Hardwaresperren oder Sicherheitsausrüstung erforderlich sein. Die Verantwortung für ordnungsgemäße Moduswechsel liegt bei demjenigen, der sie implementiert.




## Eingänge

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

| Name | Typ | Beschreibung |
|---|---|---|
| bExecute | BOOL | Moduswechsel bei steigender Flanke |
| eModeCommand | DINT | Angeforderter Modus |
| ePMLState | E_PMLState | Enumeration [▶ 10], die den aktuellen PML-Zustand der automatischen Zustandsmaschine liefert. |

### Ausgänge

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

| Name | Typ | Beschreibung |
|---|---|---|
| eModeStatus | DINT | Aktueller PML-UnitMode |
| sModeStatus | STRING | Name des aktuellen PML-UnitMode |
| bDone | BOOL | Wird TRUE, sobald der Modus-Wechsel erfolgreich durchgeführt wurde. |
| bError | BOOL | Wird TRUE, sobald ein Fehler eintritt. |
| nErrorID | UDINT | Liefert bei einem gesetzten bError-Ausgang die Fehlernummer. |

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.2 General

### 2.3.2.1 PML_AdminAlarm

Dieser Baustein unterstützt den Anwender beim Eintragen, Quittieren und Löschen von Alarm, Warning und StopReason.der Admin-PackTags. Dazu stellt der Baustein verschiedene Methoden zur bereit.

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

#### 2.3.2.1.1 M_SetAlarm

Diese Methode fügt einen Alarm in den Admin-Tags ein. Alarm[].Trigger wird TRUE gesetzt und in Alarm[].DateTime der Wert aus Admin.PlcDateTime eingetragen. Die anderen Werte werden aus der übergebenen Alarm-Struktur übernommen. Konnte der Alarm erfolgreich eingetragen werden liefert die Methode TRUE zurück

Damit ein gültiger Zeitstempel eingetragen werden kann, sollte der Baustein PML_AdminTime zyklisch im Programm aufgerufen werden.

**Syntax**

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

Beispielaufruf:

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

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

#### 2.3.2.1.2 M_AcknowledgeAlarm

Diese Methode quittiert einen Alarm in den Admin-Tags. Alarm[].Trigger wird FALSE gesetzt und in Alarm[].AckDateTime der Wert aus Admin.PlcDateTime eingetragen. Konnte der Alarm erfolgreich gefunden und quittiert werden liefert die Methode TRUE zurück. Die Quittierung des Alarms löscht diesen nicht. Der Alarm verbleibt im Alarm-Array bis auch ein Aufruf M_ClearAlarm erfolgt ist, dann wird dieser in das AlarmHistory-Array verschoben. Sollte das AlarmHistory-Array bereits mit Einträgen gefüllt sein wird der älteste Eintrag dadurch gelöscht.

Damit ein gültiger Zeitstempel eingetragen werden kann, sollte der Baustein PML_AdminTime zyklisch im Programm aufgerufen werden.

**Syntax**

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

```
  stAlarm          : ST_Alarm;
END_VAR
```

Beispielaufruf:

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.2.1.3 M_ClearAlarm

Diese Methode löscht einen Alarm in den Admin-Tags. Alarm[].Trigger wird FALSE gesetzt. Konnte der Alarm erfolgreich gelöscht werden liefert die Methode TRUE zurück. Der Alarm verbleibt im Alarm-Array bis auch ein Aufruf M_AcknowledgeAlarm erfolgt ist, dann wird dieser in das AlarmHistory-Array verschoben. Sollte das AlarmHistory-Array bereits mit Einträgen gefüllt sein wird der älteste Eintrag dadurch gelöscht.

Damit ein gültiger Zeitstempel eingetragen werden kann, sollte der Baustein PML_AdminTime zyklisch im Programm aufgerufen werden.

### Syntax

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

Beispielaufruf:

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.2.1.4 M_SetWarning

Diese Methode fügt ein Warning in den Admin-Tags ein. Warning[].Trigger wird TRUE gesetzt und in Warning[].DateTime der Wert aus Admin.PlcDateTime eingetragen. Die anderen Werte werden aus der übergebenen Warning-Struktur übernommen. Konnte das Warning erfolgreich eingetragen werden liefert die Methode TRUE zurück. Sollte das Warning-Array bereits mit Einträgen gefüllt sein wird der älteste Eintrag dadurch gelöscht.

Damit ein gültiger Zeitstempel eingetragen werden kann, sollte der Baustein PML_AdminTime zyklisch im Programm aufgerufen werden.

### Syntax

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

Beispielaufruf:

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.2.1.5 M_AcknowledgeWarning

Diese Methode quittiert ein Warning in den Admin-Tags. Warning[].Trigger wird FALSE gesetzt und in Warning[].AckDateTime der Wert aus Admin.PlcDateTime eingetragen. Konnte das Warning erfolgreich gefunden und quittiert werden liefert die Methode TRUE zurück. Das Warning verbleibt im Warning-Array, bis es durch nachfolgende Warning aus dem Array verdrängt wird.

Damit ein gültiger Zeitstempel eingetragen werden kann, sollte der Baustein PML_AdminTime zyklisch im Programm aufgerufen werden.

### Syntax

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

Beispielaufruf:

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.2.1.6 M_ClearWarning

Diese Methode löscht ein Warning in den Admin-Tags. Warning[].Trigger wird FALSE gesetzt. Konnte das Warning erfolgreich gelöscht werden liefert die Methode TRUE zurück. Das Warning verbleibt im Warning-Array, bis es durch nachfolgende Warning aus dem Array verdrängt wird.

Damit ein gültiger Zeitstempel eingetragen werden kann, sollte der Baustein PML_AdminTime zyklisch im Programm aufgerufen werden.

### Syntax

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

Beispielaufruf:

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.2.1.7 M_SetStopReason

Diese Methode fügt ein StopReason in den Admin-Tags ein. StopReason[].Trigger wird TRUE gesetzt und in StopReason[].DateTime der Wert aus Admin.PlcDateTime eingetragen. Die anderen Werte werden aus der übergebenen StopReason-Struktur übernommen. Konnte das StopReason erfolgreich eingetragen werden liefert die Methode TRUE zurück. Sollte das StopReason-Array bereits mit Einträgen gefüllt sein wird der älteste Eintrag dadurch gelöscht.

Damit ein gültiger Zeitstempel eingetragen werden kann, sollte der Baustein PML_AdminTime zyklisch im Programm aufgerufen werden.

### Syntax

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

Beispielaufruf:

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.2.1.8 M_AcknowledgeStopReason

Diese Methode quittiert ein StopReason in den Admin-Tags. StopReason[].Trigger wird FALSE gesetzt und in StopReason[].AckDateTime der Wert aus Admin.PlcDateTime eingetragen. Konnte das StopReason erfolgreich gefunden und quittiert werden liefert die Methode TRUE zurück. Das StopReason verbleibt im StopReason -Array, bis es durch nachfolgende StopReason aus dem Array verdrängt wird.

Damit ein gültiger Zeitstempel eingetragen werden kann, sollte der Baustein PML_AdminTime zyklisch im Programm aufgerufen werden.

### Syntax

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

Beispielaufruf:

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.2.1.9 M_ClearStopReason

Diese Methode löscht ein StopReason in den Admin-Tags. StopReason[].Trigger wird FALSE gesetzt. Konnte das StopReason erfolgreich gelöscht werden liefert die Methode TRUE zurück. Das StopReason verbleibt im StopReason -Array, bis es durch nachfolgende StopReason aus dem Array verdrängt wird.

Damit ein gültiger Zeitstempel eingetragen werden kann, sollte der Baustein PML_AdminTime zyklisch im Programm aufgerufen werden.

### Syntax

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

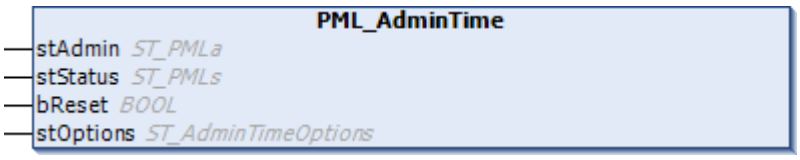
Beispielaufruf:

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 2.3.2.2 PML_AdminTime




Dieser Funktionsbaustein sollte zyklisch aufgerufen werden und füllt dann die folgenden Admin-PackTags:

- PlcDateTime
- AccTimeSinceReset
- ModeCurrentTime[ ]
- ModeCummulativeTime[ ]
- StateCurrentTime[ ][ ]
- StateCummulativeTime[ ][ ]

Damit wird aufgezeichnet, für welche Zeit die Maschine sich in den verschiedenen States befunden hat. Dadurch können im Weiteren Rückschlüsse über Maschineneffizienz gewonnen werden. Damit die Zeiten korrekt berechnet werden, wird vorausgesetzt, dass die Status-PackTags *UnitCurrent* und *StateCurrent* bereits sinnvoll beschrieben werden.

#### Eingänge

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

| Name | Typ | Beschreibung |
|---|---|---|
| bReset | BOOL | Ein Signal an diesem Eingang setzt die aufgezeichneten Zeiten zurück. |
| stOptions | ST_AdminTimeOptions | Zusätzliche Optionen des Bausteins |

#### Ein/Ausgänge

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

| Name | Typ | Beschreibung |
|---|---|---|
| stAdmin | ST_PMLa | Übergabe der Admin-PackTags |
| stStatus | ST_PMLs | Übergabe der Status-PackTags |

#### Voraussetzungen

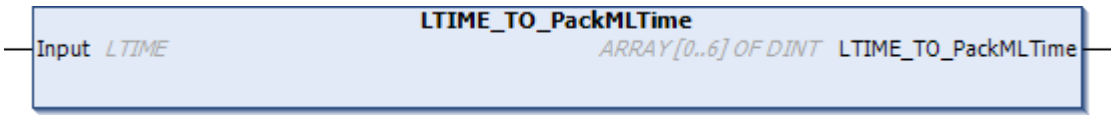
| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.3 Conversion

### 2.3.3.1 Time

Diese Funktionen konvertieren Zeitwerte in das PackML-konforme Array.

#### 2.3.3.1.1 LTIME_TO_PackMLTime




Die Funktion konvertiert einen Zeitwert im LTIME-Format in das PackML-konforme Array.

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

**Eingänge**

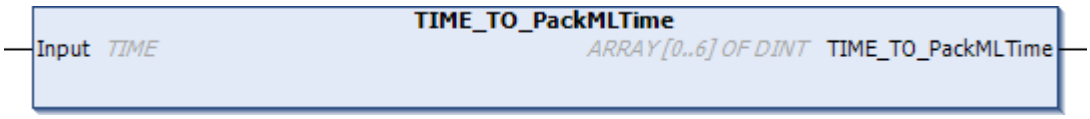
```
VAR_INPUT
    Input          : LTIME;
END_VAR
```

| Name | Typ | Beschreibung |
|---|---|---|
| Input | LTIME | Der zu konvertierende Zeitwert |

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

#### 2.3.3.1.2 TIME_TO_PackMLTime




Die Funktion konvertiert einen Zeitwert im TIME-Format in das PackML-konforme Array.

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

**Eingänge**

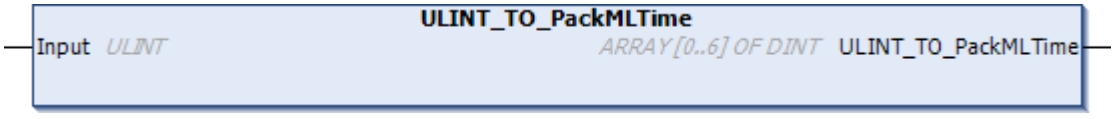
```
VAR_INPUT
    Input          : TIME;
END_VAR
```

| Name | Typ | Beschreibung |
|---|---|---|
| Input | TIME | Der zu konvertierende Zeitwert |

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

#### 2.3.3.1.3 ULINT_TO_PackMLTime



Die Funktion konvertiert einen Zeitwert im ULINT-Format in das PackML-konforme Array.

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

### Eingänge

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

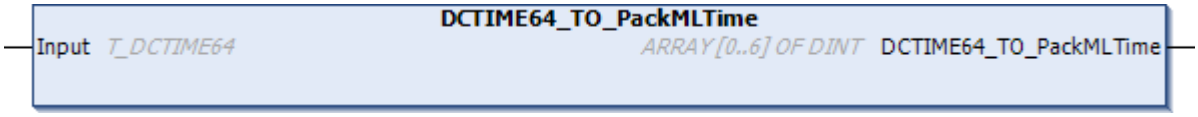
| Name | Typ | Beschreibung |
|---|---|---|
| Input | ULINT | Der zu konvertierende Zeitwert |

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 2.3.3.2 Timestamp

### 2.3.3.2.1 DCTIME64_TO_PackMLTime




Die Funktion konvertiert einen Zeitpunkt im DCTIME64-Format in das PackML-konforme Array.

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

### Eingänge

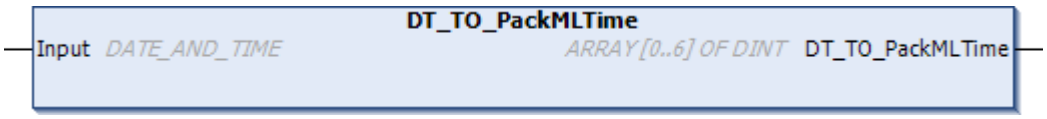
```
VAR_INPUT
    Input          : DCTIME64;
END_VAR
```

| Name | Typ | Beschreibung |
|---|---|---|
| Input | DCTIME64 | Der zu konvertierende Zeitpunkt |

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 2.3.3.2.2 DT_TO_PackMLTime




Die Funktion konvertiert einen Zeitpunkt im DT-Format in das PackML-konforme Array.

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

### Eingänge

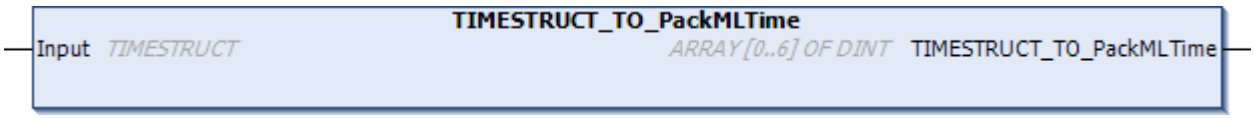
```
VAR_INPUT
    Input          : DT;
END_VAR
```

| Name | Typ | Beschreibung |
|---|---|---|
| Input | DT | Der zu konvertierende Zeitpunkt |

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 2.3.3.2.3 TIMESTRUCT_TO_PackMLTime




Die Funktion konvertiert einen Zeitpunkt im TIMESTRUCT-Format in das PackML-konforme Array.

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

**Eingänge**

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

| Name | Typ | Beschreibung |
|---|---|---|
| Input | TIMESTRUCT | Der zu konvertierende Zeitpunkt |

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 2.3.3.3 F_StateCommandToString


Die Funktion gibt den Namen eines State-Kommandos als Zeichenkette aus.

**FUNCTION F_StateCommandToString : STRING;**

**Eingänge**

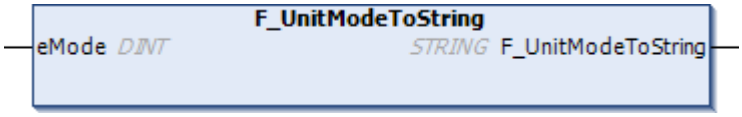
```
VAR_INPUT
    eStateCommand          : E_PMLCommand;
END_VAR
```

| Name | Typ | Beschreibung |
|---|---|---|
| eStateCommand | E_PMLCommand | Das State-Kommando zu dem der Name ermittelt werden soll |

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 2.3.3.4 F_UnitModeToString




Die Funktion liefert den Namen eines Unit Mode als Zeichenkette zurück.

**FUNCTION F_UnitModeToString : STRING;**

#### Eingänge

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

| Name | Typ | Beschreibung |
|---|---|---|
| eMode | DINT | Der Unit Mode zu dem der Name ermittelt werden soll. |

#### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

# 3 Packaging Machine Tags

## 3.1 Einleitung

PackTags stellen einen einheitlichen Satz Namenskonventionen für Datenelemente bereit, die in den prozeduralen Elementen des Basiszustandsmodells verwendet werden. Wie beschrieben stellt das Base State Model (Basiszustandsmodell) einen einheitlichen Satz Maschinenzustände bereit, so dass alle automatisierten Maschinen gleich betrachtet werden können. PackTags sind mit Namen versehene Datenelemente für interoperablen Datenaustausch bei automatisierten Maschinen offener Architekturen. In dieser Dokumentation finden Sie die wesentlichen Namen der Datenelemente, Datentyp, Werte, Bereiche und gegebenenfalls Datenstrukturen. PackTags werden für die Maschine-zu-Maschine-Kommunikation eingesetzt, z.B. zwischen einem Abfüller und einem Kappenaufsetzer. PackTags können auch für den Datenaustausch zwischen Maschine und übergeordnetem Informationssystem wie Manufacturing Operations Management und Enterprise Information Systemen eingesetzt werden.

Diese Dokumentation beschreibt alle PackTags für die Navigation durch ein Zustandsmodell und für die Definition und Betätigung des Anlagensteuerungsmodus. Des Weiteren definiert diese Dokumentation auch eine Liste von PackTags, die eventuelle wichtige Informationen einer Maschine bereitstellen. Alle PackTags müssen genutzt werden, um den Prinzipien integrierter Konnektivität mit Systemen mit der gleichen Implementierungsmethode zu entsprechen.

Notwendig sind die Tags, die für die Funktion der automatisierten Maschine oder die Konnektivität zu Kontroll- oder Fernsystemen benötigt werden.

## 3.2 Tag-Arten

PackTags werden in drei Gruppen aufgegliedert: Command (Befehl), Status (Zustand) und Administration (Verwaltung). Befehl- und Zustand-Tags enthalten Daten für die Anbindung der Maschine an die Liniensteuerung zur Koordination oder zum Herunterladen von Rezepten/Parametern. Befehl-Tags werden als „Informationsempfänger“ an das Maschinenprogramm „übergeben“ und von ihm konsumiert, Zustand-Tags werden vom Maschinenprogramm erzeugt und ausgelesen. Verwaltung-Tags enthalten Daten, die übergeordnete Systeme zur Analyse der Maschinenleistung oder zur Information der Bediener sammeln.

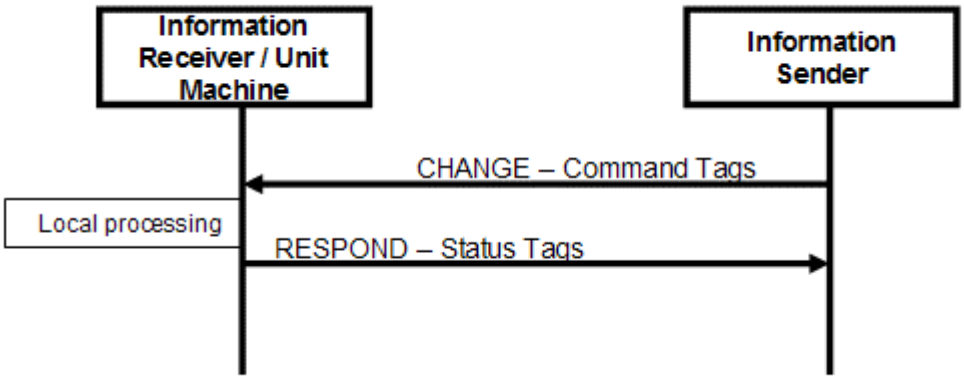
Die Gruppierung von Daten sollte in benachbarten Registern erfolgen, um die Kommunikation zu optimieren.

Normalerweise werden Informationsdaten via OPC in einem Ethernet-basierten Kommunikationsnetzwerk übertragen.

Präfix von Befehl-Tags ist „PMLc“.

Präfix von Zustand-Tags ist „PMLs“.

Präfix von Verwaltung-Tags ist „PMLa“.



## 3.3 Tag-Details

Der folgende Abschnitt gibt einen Überblick über die Tags. In den nachfolgenden Tabellen sind Befehl-, Zustand- und Verwaltung-PackTags aufgelistet.

### Befehlsstruktur PMLc

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

### Zustandsstruktur PMLs

| | | | | | Tag Name | Data Type |
|---|---|---|---|---|---|---|
| PMLs | | | | | PMLs | ST_PMLs |
| | UnitModeCurrent | | | | PMLs.UnitModeCurrent | DINT |
| | UnitModeRequested | | | | PMLs.UnitModerequested | DINT |
| | UnitModeChangeInProcess | | | | PMLs.UnitModeChangeInProcess | BOOL |

| | | | | | | |
|---|---|---|---|---|---|---|
| | StateCurrent | | | | PMLs.StateCurrent | DINT |
| | StateRequested | | | | PMLs.StateRequested | DINT |
| | StateChangeInProcess | | | | PMLs.StateChangeInProcess | BOOL |
| | MachineSpeed | | | | PMLs.MachineSpeed | REAL |
| | CurMachineSpeed | | | | PMLs.CurMachineSpeed | REAL |
| | MaterialInterlock | | | | PMLs.MaterialInterlock | DINT |
| | EquipmentInterlock | | | | PMLs.EquipmentInterlock | ST_Equipment |
| | | Blocked | | | PMLs.EquipmentInterlock.Blocked | BOOL |
| | | Starved | | | PMLs.EquipmentInterlock.Starved | BOOL |
| | RemoteInterface[#] | | | | PMLs.RemoteInterface[#] | ST_Interface |
| | | Number | | | PMLs.RemoteInterface[#].Number | DINT |
| | | ControlCmdNumber | | | PMLs.RemoteInterface[#].ControlCmdNumber | DINT |
| | | CmdValue | | | PMLs.RemoteInterface[#].CmdValue | DINT |
| | | Parameter[#] | | | PMLs.RemoteInterface[#].Parameter[#] | ST_Descriptor |
| | | | Id | | PMLs.RemoteInterface[#].Parameter[#].Id | DINT |
| | | | Name | | PMLs.RemoteInterface[#].Parameter[#].Name | STRING |
| | | | Unit | | PMLs.RemoteInterface[#].Parameter[#].Unit | STRING(5) |
| | | | Value | | PMLs.RemoteInterface[#].Parameter[#].Value | REAL |
| | Parameter[#] | | | | PMLs.Parameter[#] | ST_Descriptor |
| | | Id | | | PMLs..Parameter[#].Id | DINT |
| | | Name | | | PMLs..Parameter[#].Name | STRING |
| | | Unit | | | PMLs..Parameter[#].Unit | STRING(5) |
| | | Value | | | PMLs..Parameter[#].Value | REAL |
| | Product[#] | | | | PMLc.Product[#] | ST_Product |
| | | ProductId | | | PMLc.Product[#].ProductId | DINT |
| | | ProcessVariables[#] | | | PMLc.Product[#].ProcessVariables[#] | ST_Descriptor |
| | | | Id | | PMLc.Product[#].ProcessVariables[#].Id | DINT |
| | | | Name | | PMLc.Product[#].ProcessVariables[#].Name | STRING |
| | | | Unit | | PMLc.Product[#].ProcessVariables[#].Unit | STRING(5) |
| | | | Value | | PMLc.Product[#].ProcessVariables[#].Value | REAL |
| | | Ingredients[#] | | | PMLc.Product[#].Ingredients[#] | ST_Ingredient |
| | | | IngredientId | | PMLc.Product[#].Ingredients[#].IngredientId | DINT |
| | | | Parameter[#] | | PMLc.Product[#].Ingredients[#].Parameter[#] | ST_Descriptor |
| | | | | Id | PMLc.Product[#].Ingredients[#].Parameter[#].Id | DINT |
| | | | | Name | PMLc.Product[#].Ingredients[#].Parameter[#].Name | STRING |
| | | | | Unit | PMLc.Product[#].Ingredients[#].Parameter[#].Unit | STRING(5) |
| | | | | Value | PMLc.Product[#].Ingredients[#].Parameter[#].Value | REAL |

### Verwaltungsstruktur PMLa

| | | | | | **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 |
| | StateCummulativeTime[#,#] | | | | 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 Datentypen

## 3.4.1 Alarm

### 3.4.1.1 ST_Alarm

Sammlung der Tags für die Beschreibung von Alarmereignissen.

```
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
```

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 3.4.1.2 ST_DateAndTime

Diese Struktur dient zur Speicherung von Datum und Uhrzeit eines Ereignisses oder zum Quittieren eines Ereignisses.

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

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 3.4.2 Allgemein

### 3.4.2.1 ST_Count

Sammlung von Tags für die Beschreibung von Parametern in der Maschine.

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

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 3.4.2.2 ST_Descriptor

Sammlung von Tags für die Beschreibung von Parametern in der Maschine.

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

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 3.4.2.3 ST_Equipment

Sammlung von Tags für die Beschreibung von Parametern in der Maschine.

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

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 3.4.2.4 ST_Ingredient

Sammlung von Tags für die Beschreibung der benötigten Rohmaterialien für das Produkt.

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

```
END_STRUCT
END_TYPE
```

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 3.4.2.5 ST_Interface

Sammlung von Tags für die Beschreibung von Materialien in der Maschine.

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 3.4.2.6 ST_Product

Sammlung von Tags für die Beschreibung des auf der Maschine gefertigten Produkts.

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 3.4.3 ST_PMLa

Sammlung aller Verwaltung-Tags der PackTag-Struktur.

```
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
```

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 3.4.4 ST_PMLc

Sammlung aller Befehl-Tags der PackTag-Struktur.

```
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
```

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

### 3.4.5 ST_PMLs

Sammlung aller Zustand-Tags der PackTag-Struktur.

```
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
```

**Voraussetzungen**

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 3.5 Globale Parameter

Parameter für den Aufbau der Packaging Maschine Tag Strukturen. Diese können beim Einfügen der Bibliothek für das aktuelle Projekt angepasst werden.

```
(* 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
```

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

## 3.6 Globale Konstanten

Konstanten für den Aufbau der Packaging Maschine Tag-Strukturen. Diese können nicht verändert werden.

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

### Voraussetzungen

| Entwicklungsumgebung | Zielplattform | Einzubindende SPS Bibliotheken |
|---|---|---|
| Ab TwinCAT v3.1 Build 4018 | PC (i386) | Tc3_PackML_V2 |

# 4 Beispiel Tc3_PackML_V2

Das Beispiel zeigt anhand einer visualisierten Sortieranlage, wie die Tc3_PackML_V2-Bibliothek als Basis einer Maschinensteuerung verwendet werden kann:

https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc3_PackML_V2/Resources/3414434955.zip

# 5 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

### Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

### Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

### Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

| | |
|---|---|
| Hotline: | +49 5246 963-157 |
| E-Mail: | support@beckhoff.com |

### Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

| | |
|---|---|
| Hotline: | +49 5246 963-460 |
| E-Mail: | service@beckhoff.com |

### Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

| | |
|---|---|
| Telefon: | +49 5246 963-0 |
| E-Mail: | info@beckhoff.com |
| Internet: | www.beckhoff.com |

Mehr Informationen:
**www.beckhoff.com/te1000**

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