

PACKML UNIT/MACHINE IMPLEMENTATION GUIDE

Part 1: PackML Interface State Manager

Summary

This document “PackML Interface State Manager is version 2 of the PackML unit/machine Implementation Guide. The PackML unit/machine Implementation Guide is a Best Practice Recommendation based on ANSI/ISA TR88.00.02-2022. The purpose of this guideline is to help companies with different knowledge levels to implement PackML interfaces on Units/Machines. This document contains best practice recommendations based on the real implementation of PackML interfaces by a number of OMAC member companies as well as other users. Data will in future give big value in the Manufacturing Environment and the Supply Chain, and the PackML interface is one of the keys for accruing harmonized data to be able to digitize the Manufacturing environment and obtain Industry 4.0 application. One of the benefits of using the PackML Interface is the standardized data model and common understanding of state and mode of a Unit/Machine. It will give the possibility to harmonize data according to the above IT systems such as SCADA and MES. The usage of EDGE and Cloud on premiss systems will also increase the possibility to acquire PackML data from many units/machines and dispatch the data to the IT systems that benefit of using the data.

There are major changes between the ISA TR88.00.02-2015 and the new version ISA TR88.00.02-2022, is that the new version is more align with the core standard ISA-88.00.01 and Order (i.e. Batch) execution via recipes. The changes covers updated State Model, removal of the “Remote” Interface and the addition of PackTags that describe the Unit/Machine configuration via “Recipe” to a supervisory system and the Stack Light information.

This document is the first part in a series of document related to build PackML unit/machines. Further documents are forthcoming and will focus on issues such as PackML Machine software code structure, PackML network connections, PackML and Safety, PackML and Line Integration and PackML HMI.

Author: Ph.D. Carsten Nøkleby, SESAM-World

1. Contents

2. Executive Summary	4
2.1 Purpose	4
2.2 Goals	4
2.3 Background.....	5
2.4 Related PackML documents in process	6
3. Scope of the Implementation Guide.....	7
3.1 PackML Interface State Manager.....	9
4. Terminologies, Definitions and Abbreviations.....	10
4.1 ISA 88.01	10
4.2 The ISA 88 Physical Model.....	10
4.3 ISA 88 Recipe Management – unit/machine control recipe parameters	12
5. Identifying and defining a Unit	15
6. The PackML Interface State Model	17
6.1 The syntax of the PackML state model	17
6.2 The PackML Interface State Model	18
6.3 Colours used for the PackML State Model	19
6.4 The PackML State Model from a unit/machine perspective	20
6.5 The PackML State Model from an internal and external perspective	21
6.6 The PackML State Model from a unit/machine parameter perspective – production orders	22
6.7 The PackML State Model and Stacklight.....	25
7. PackML Event State Manager.....	27
7.1 Resetting	28
7.2 Starting	30
7.3 Execute	32
7.4 Holding.....	34
7.5 Unholding	39
7.6 Suspending	43
7.7 Unsuspending.....	46
7.8 Completing	49
7.9 Stopping	51
7.10 Aborting	53
7.11 Mapping table.....	56
7.12 Mapping Alarms to event triggers to PackML.....	58
8. PackML Control Command definitions	59

8.1	PackML Commands	59
8.2	Specification of commands on unit HMI and from External system	62
9.	PackML Interface State definition	63
10.	Unit control modes	67
11.	PackTags.....	70
11.1	Description and definition of PackTags.....	74
11.2	Migration from 2015 toward 2022 PackTags	74
12.	Example of Unit Controller Functionality	79
12.1	Syntax and Symbolic description	79
12.2	Prepare unit/machine	80
12.3	Start unit/machine	80
12.4	Stop unit/machine from Unit panel/HMI	82
12.5	Stop unit/machine from external system	83
12.6	Operator Break and pause on unit/machine	85
12.7	Resume unit/machine after break from External System	86
12.8	Stop Unit from unit panel/HMI.....	87
12.9	Abort unit/machine	89
12.10	Re-start unit from unit panel/HMI after stop or abort.....	90
12.11	Re-start unit from External system after stop or abort	91
12.12	Alarm and warning on unit	92
12.12.1	PackML alarm and event state management table.....	93
12.13	Background task: OEE	94
12.14	Background task: Mode & State	95
12.15	Suspend: Starvation or saturation.....	98
12.16	Unsuspend	99
13.	Reference information	100
13.1	Definitions and abbreviations	100
13.2	References.....	101
13.3	List of people involved in preparation of the guideline	102
13.4	Support	104
13.5	The author	104
14.	Version History	105

This page is initially left blank.

2. EXECUTIVE SUMMARY

2.1 PURPOSE

This PackML Interface State Manager document is the first part of the PackML unit/machine Implementation Guide, which is a Best Practice Recommendation based on ANSI/ISA TR88.00.02-2022. This guideline will help companies, with varying levels of knowledge and experience, to gain familiarity with the ANSI/ISA TR88.00.02-2022 technical report, and how to apply the procedures described to real-world machine applications. The PackML Interface State Manager is focused on the execution of a Production Order on a machine, and not how the machine performing its control functions. A unit/machine is defined as a collection of physical equipment and control functions that perform one or more major processing functions. A unit/machine can be a single machine or a subset of a whole packaging line.

The background information for the first version of the PackML unit/machine Implementation Guide has been collected from workgroup meetings held in 2015, 2016 and the ISA TR88.00.02-2015 version. The practical experience from the members of the workgroup has been used to construct ready-to-use, generic and vendor-independent implementation models.

The background information for the second version is the updated ISA TR88.00.02-2022.

The main changes from version 1 of the PackML unit/machine Implementation Guide, is the PackML state model is updated and extended with wider scope of the PackML commands. The biggest change is related to the PackTags where a Recipe structure for handling Production Order data have been added, as well as Tag for stacklight, and the remote interface is removed.

This PackML implementation guide is intended for Machine Suppliers, System Integrators and End Users. This guideline describes the requirements for a unit/machine to be compliant with the PackML standard as described in ANSI/ISA TR88.00.02-2022 and provides examples of implementations that can be used as guidance for specific machines.

2.2 GOALS

Data will be the driver for new Business Models and the way of working. Data will in near future give big value in the Manufacturing Environment and the Supply Chain, and the PackML interface is one of the keys for accruing harmonized data to be able to digitize the Manufacturing environment and obtain Industry 4.0 application. One of the benefits of using the PackML Interface is the standardized data model and common understanding of state and mode of a Unit/Machine. It will give the possibility to harmonize data according to the above IT systems such a SCADA and MES. The usage of EDGE and Cloud on premiss will also increase the possibility to acquire PackML data from many units/machines and dispatch the data to the IT systems that benefit of using the data.

This PackML implementation guide provides a definition of the requirements of a PackML interface. For example, there are practical examples on how different conditions can be handled when an error or event occurs on the equipment level that generates an alarm or warning. It also addresses how different interpretation from different users may be reconciled. The guideline provides examples of how to configure the PackML State model transitions according to specific End User requirements.

Furthermore, the guideline takes the maturity level of Machine Suppliers into consideration, as well as Machine Suppliers readiness or ability to implement PackML partly or fully.

Benefits for both End Users and Machine Suppliers include:

- Easy integration of unit/machines and test of interfaces
- Reduction of integration cost for a complete packaging line.
- Same operator Interface from a supervisory system
- Faster interface specification
- Less risk / Less uncertainty in commercial contract
- Less training effort
- Reliable data, e.g. OEE, energy data
- Replace one unit/machine at a line
- A standardized semantic model of machines and equipment on shop floor.
- The standard for being able to digitalize the manufacturing environment and supply chain.
- PackML combined by the Edge technology and Cloud on Premiss will speed up the journey toward Industry 4.0

2.3 BACKGROUND

The OMAC End Users want to achieve a unified way of interfacing with units/machines on the factory floor. Providing this unified interface will ensure easy integration with a supervisory control system. There are similar interfaces to all units, and the units have the same data structure interfaces available.

By applying a unified user interface based on the PackML State Model an operator can be guided in case of an unplanned stoppage. Furthermore, there can be common event handling on all units, providing the End User the possibility to view stop reasons in a common way for all units.

The unified data interface based on PackML and ISA TR88.00.02-2022 will help in the integration and structuring of data for the individual unit/machine and give the possibility to implement a common interface from the supervisory control system – E.g. Manufacturing Execution System (MES) or Supervisory Control and Data Acquisition (SCADA).

The interface described in this document is built on the international reference model PackML defined by The International Society of Automation (ISA). The interface follows the PackML State Model as described in the reference ANSI/ISA TR88.00.02-2022.

It is recommended to have knowledge on the batch standard ISA 88.00.01 as it is the basic for the related standards in the ISA 88 domain, as well as the Technical Report TR88.00.02-2022. The Technical Report is an interpretation of ISA 88.00.01 for discrete machines and manufacturing environment. Basic understanding and knowledge of the TR88.00.02 technical report is required before reading this Best Practice Recommendation. In fact, it is strongly recommended to have the report in hand when reading this document.

A unit/machine is defined as a collection of physical equipment and control functions that perform one or more major processing functions. A unit/machine can be a single machine or a subset of a whole packaging line. The term unit and machine are used interchangeably in this document.

Machine Suppliers are defined as companies or organizations that provide PackML compliant unit/machines. End users are defined as companies or organizations that use PackML compliant unit/machines in production facilities.

2.4 RELATED PACKML DOCUMENTS IN PROCESS

This document is the first part in a series of documents related to build PackML unit/machines. Further documents¹ will come and have focus on issues like:

- Part 1: PackML Interface State Manager (Available from OMAC)
- Part 2: PackML Standard Representation of PackTags in an OPC UA server (Available from OMAC)
- Part 3: PackML Network Connections
- Part 4: PackML and Safety
- Part 5: PackML and Line Integration
- Part 6: PackML User Interface – HMI (Available from OMAC)

¹ The list is not complete and there could be changes in the document titles. Part 1, 2 and 6 is available from the OMAC homepage.

3. SCOPE OF THE IMPLEMENTATION GUIDE

The PackML unit/machine Implementation Guide Part 1 focuses only on an individual unit/machine and its PackML Interface implementation, without any relationship to other units/machines or equipment.

The document does not cover how several Units/Machines are operated together in a line, this is to be covered by OMAC Part 5: PackML and Line Integration.

The PackML specification is an implementation of the ANSI/ISA-TR88.00.02-2022, Machine and unit/machine States. As an implementation example of ANSI/ISA-88.00.01, it covers the following two application examples:

1. PackML Interface State Manager²
2. PackML Machine State Manager (Not part of this document³)

The focus of the PackML unit/machine Implementation Guide Part 1 is the “PackML Interface State Manager”, equal to the top (blue) part of Figure 1.

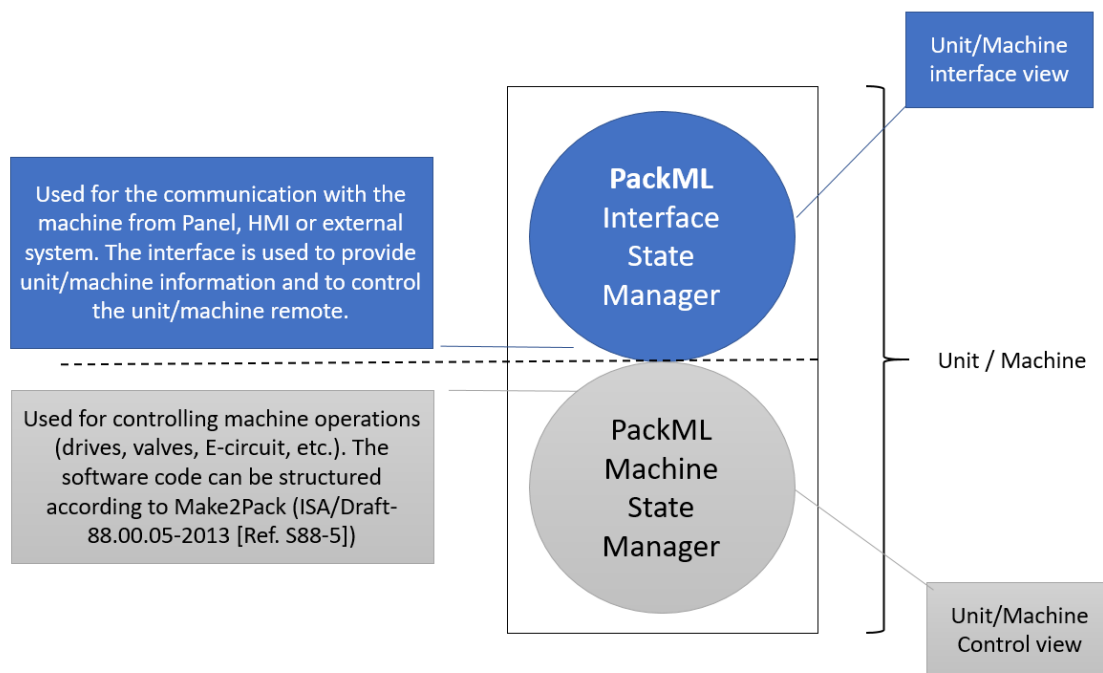


Figure 1: The definition of PackML Interface State Manager and PackML Machine State Manager

Machine Suppliers wanting to use the PackML state model for their internal Machine State Manager have two scenarios shown in Figure 2:

- 1) PackML gateway unit/machine
- 2) Full PackML compliant unit/machine

² ISA-TR88.00.02 describes a “base state model and the PackTags data area is used to drive action into the OEM’s “PackML Machine State Manager” using the PackTags Command, Status and Admin tags.

³ The full Make2Pack decomposition approach is part of ISA88.00.05-2013-Draft [Ref. No. S88-5].

The focus of the PackML unit/machine Implementation Guide Part 1 is the PackML Gateway solution and covers the PackML Interface State Manager.

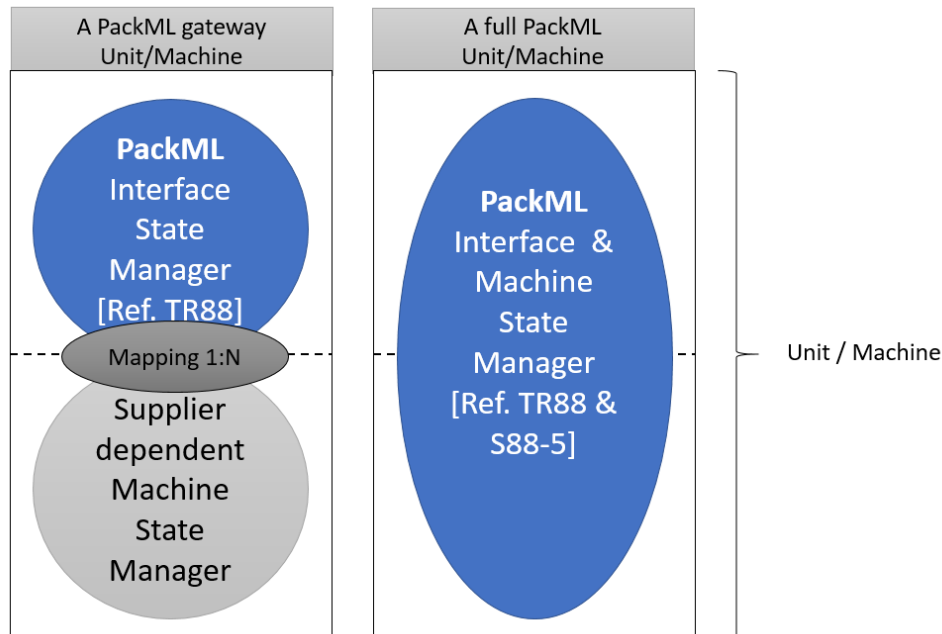


Figure 2: Mapping ⁴ of PackML Interface State Manager and Integrated Machine State Manager & PackML Interface

To implement a PackML gateway, Machine Suppliers must map the PackML Interface state model to their specific internal state logic³.

Machine Suppliers that use a PackML gateway unit/machine, must specify the mapping between the PackML Interface State Manager and their specific Machine State Manager. Normally, all Machine Suppliers should be able to fulfil the mapping to the PackML Interface State Manager.

When Machine Suppliers have a Machine State Manager based on PackML it is not necessary to specify any mapping between the PackML Interface State Manager and the Machine State Manager because they are the same⁵.

The PackML gateway approach makes sense for legacy equipment, and for Machine Suppliers that have fixed requirements for internal machine state logic.

To obtain a full PackML unit/machine it is often required that the unit/machine must be reprogrammed in accordance with ISA-88 [Ref. S88-1], PackML [Ref TR88] and Make2Pack [Ref. S88-5-Draft].

⁴ The mapping table is in section 7.11.

⁵ In the case where a machine has multiple state managers, each state manager should be presented to external systems, with appropriate mapping defined. For example a palletizer handling to pallets with different packaging patterns, might have a PackML State Manager for each part of the machine handling the individual pallet locations.

3.1 PACKML INTERFACE STATE MANAGER

The PackML Interface State Manager is used for communication with the unit/machine from Panels, HMIs or external systems. The Interface is used to get status of the unit/machine, start the unit/machine, stop the unit/machine, change recipe parameters, etc.

The PackML Interface State Manager provides a single communication interface between the HMI or other external control system (e.g. other unit/machine or supervisory control system) and the unit/machine.

The PackML Interface State Manager can either be implemented in the same control system (CPU) as the Machine State Manager or in a separate control system. A full PackML unit/machine will have one common Interface and one Machine State Manager that is totally integrated with the Machine State Manager. See Figure 2 *Figure 2: Mapping of PackML Interface State Manager and Integrated Machine State Manager & PackML Interface*.

From a communication point of view, a unit/machine has a single PackML Interface State Manager, which gives one interface for the operation of the unit/machine from an HMI and/or external control system.

For example, a unit/machine may have three internal drive controllers but it is seen as one unit/machine from a user and operator point of view. For example, a depalletizer that consists of some infeed conveyors, a robot and some outfeed conveyors is seen as a single unit/machine. The definition of the individual unit/machine is often related to the primary processing scope performed.

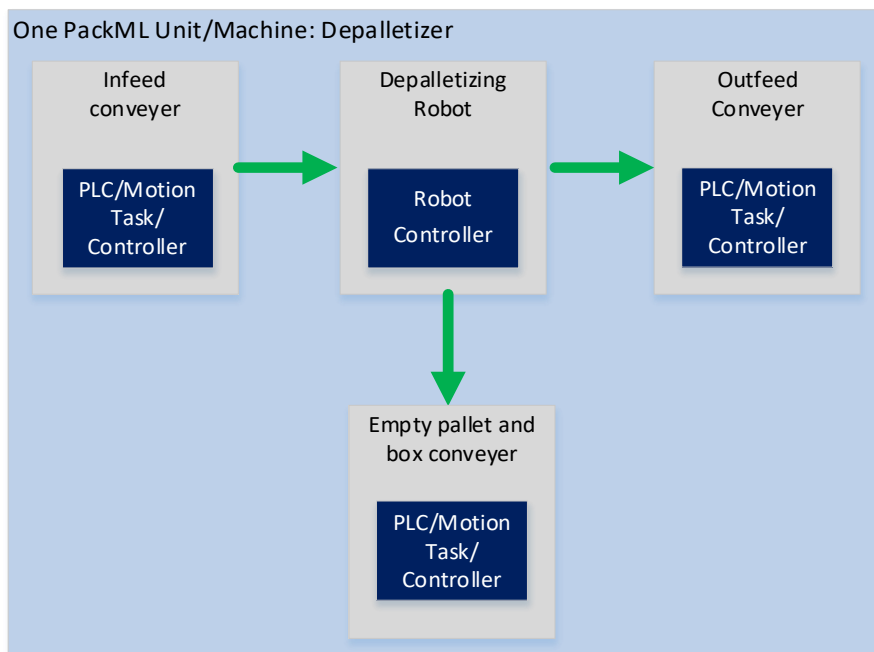


Figure 3 Example of a depalletizer composed of several controllers.

4. TERMINOLOGIES, DEFINITIONS AND ABBREVIATIONS

4.1 ISA 88.01

The standard ANSI/ISA-88.00.01-2010 [Ref. S88-1] is not only about just BATCH Industries! The standard applies to different types of production processes: discrete, continuous, and batch industries.

S88-1 is a model and methodology for designing & operating control system for flexible machines and manufacturing. Its architecture and models are independent of the underlying control system (PLC, DCS, PC, etc.) and independent of the underlying basic control algorithms.

The goal of the S88-1 is to allow recipe development without the service of a control systems engineer, and without control system reprogramming required. This is element is put into the ISA TR88.00.02-2022, which is the new interpretation of S88-1 in relation to discrete manufacturing machines and equipment.

4.2 THE ISA 88 PHYSICAL MODEL

The ANSI/ISA-88.00.01-2010 [Ref. S88-1] and the ANSI/ISA-TR88.00.02-2022 [TR88] define the levels of equipment and machines via a Physical Model, as shown in Figure 4. The PackML Machine Implementation Guide focuses only on the unit/machine in the physical model described below. The unit/machine in the ISA 88 Physical Model is equivalent to a Machine such as a Depalletizer, a Bagger, a Filler, a Capper, a Box Filler, a Palletizer, etc.

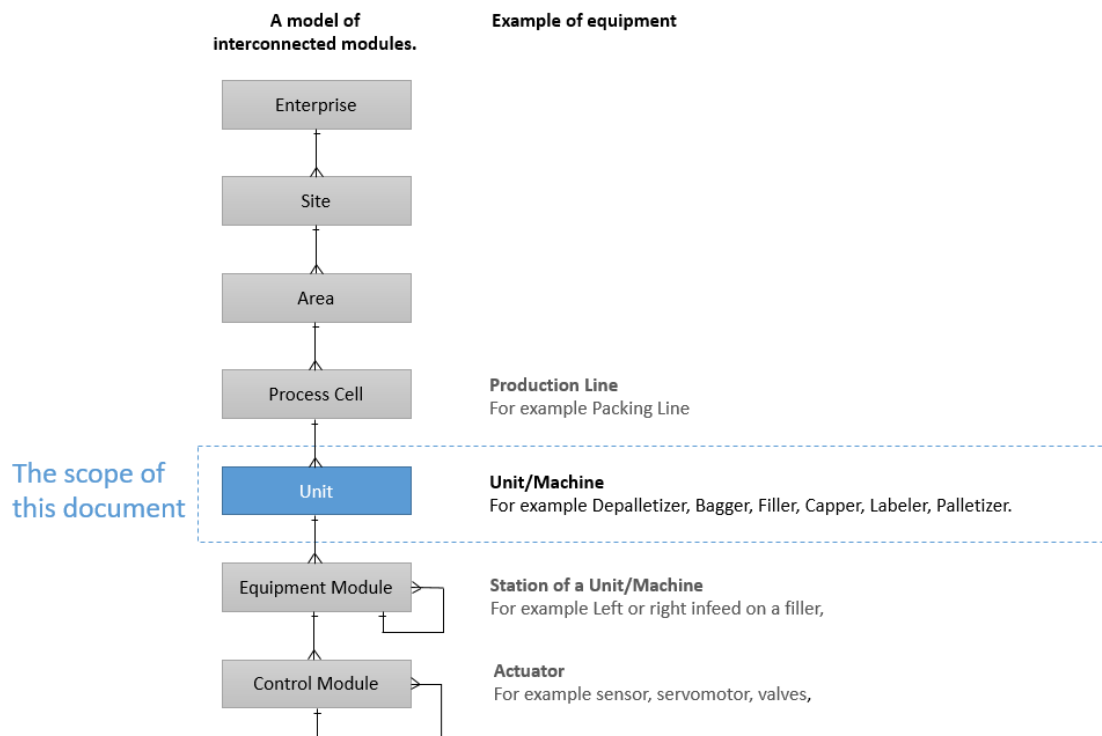


Figure 4: The ISA 88 Physical model mapped to Packing Equipment

The ISA Physical model has the following levels:

- Enterprise / Site / Area (Out of scope of this document).
- Process Cell
 - The entire collection of equipment needed to execute a single recipe from start to finish Production Order.
 - Consists of one or more units
- Unit/machine - Machine
 - A collection of equipment modules and control modules
 - Usually centered on a major piece of equipment
 - Frequently operates on or contains the complete Production Order.
 - May operate on or contain only a portion of the complete Production Order.
 - Cannot operate on or contain more than one Production Order at a time⁶
- Equipment Module
 - A collection of control modules or other equipment modules

⁶ In case it is necessary to run more than one production order at one time it is necessary to have several PackML Interface State Managers on the machine. An alternative is to have a buffer with several productions orders to be executed. It is then up to the machine to step through the production orders according to the PackML Interface State Manager. An example of a physical Unit/Machine as a Palletizer, that is able to run two Productions Orders at a time, and therefore requires a PackML Interface State Managers for each part of the equipment handling each Production Order.

- Can carry out a finite number of minor processing activities, i.e., phases
- Contains all the necessary processing equipment to carry out these processing activities
- For example, an equipment module might consist of, e.g. a circulation pump, a chilled water valve, a steam valve and a temperature controller. In this case the equipment module would represent a temperature control system
- Control Module
 - A collection of sensors, actuators, other control modules, and associated processing equipment
 - Acts as a single entity from a control standpoint
 - Is the direct connection to the process through its sensor and actuators
 - Examples of Control Modules: Valve, Pump, Motor, Pressure Controller, Limit Switch, ...

4.3 ISA 88 RECIPE MANAGEMENT – UNIT/MACHINE CONTROL RECIPE PARAMETERS

The ISA-TR88.00.02-2022 is changed and are more in line with the basic standard ISA-88.00.01, which defines a physical model, a recipe model and procedural model that makes it possible to make physical systems that are flexible in relation to what product to produced. The product is based on the recipe, which contains information on Process Variables (Temperatures, Speed, torque, etc.) and Ingrediencies (cardboard, glue, foil, pallet, etc.). This is one of the larges changes of the standard, as it is the key to achieving flexible production systems in the future.

Process related unit/machine parameters are used to specify which process specific task the unit/machine has to perform. The process related unit/machine parameters represent what the unit/machine should do for a specific production order, such as glue amounts or printer label format. In relation to ISA 88 [Ref. S88-1], such parameters are called control recipe parameters.

The focus within this guide is Control Recipe parameters, covering Process Variables and Ingredients. No other recipe management elements are in scope.

The Control Recipe parameters represent the specific process related parameters that can be configured on the Unit/Machine for preforming a specific task. For example, a Palletizer may have parameters that specify the packing pattern and number of layers on a pallet.

A PackML unit/machine should be able to handle process related machine parameters, represented as Control Recipe paTherameters, which are process relevant and process related parameters. Within ISA TR88.00.02-2022 the recipe parameters are represented by the PackTags Command.Recipe[#] and Status.Recipe[#].

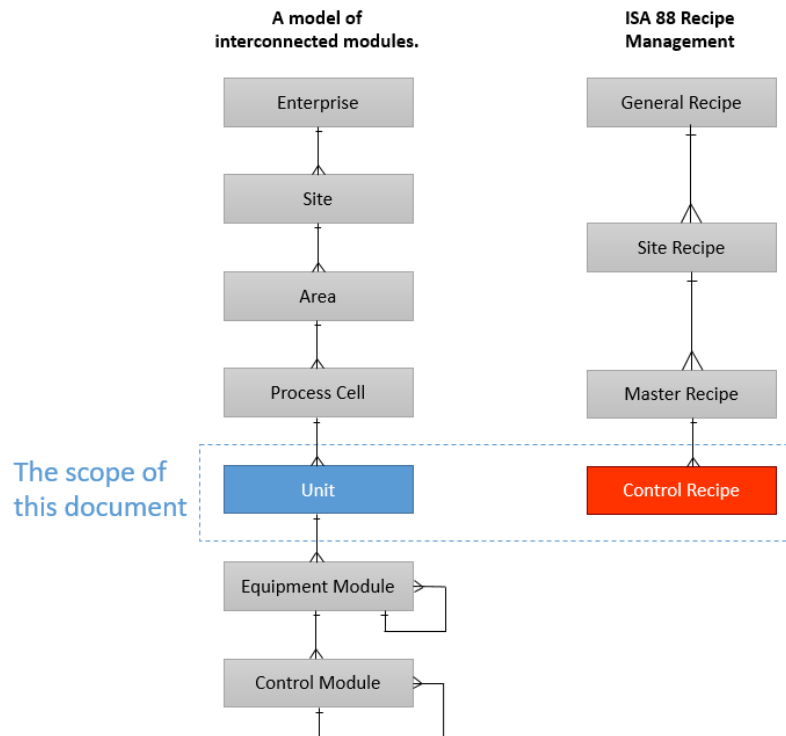


Figure 5: ISA physical model and ISA Recipe Management

The Recipe is the necessary set of information that uniquely defines the production requirements of a specific product. The relationship between recipes and equipment is shown in Figure 5.

- One **General Recipe** per produced material, maintained at the enterprise level (ERP system and defines the Bill of Materials – BOM).
- One **Site Recipe** per site and produced material, maintained at the site for local materials, language, or segment of production.
- One **Master Recipe** per production line (Process Cell) and produced material.
- One **Control Recipe** per production order that describes the parameters which need to be set in the unit/machine for the production of one specific product. An example could be a palletizer unit/machine that receives the three process related parameters for a production order: Packing pattern, number of layers and interleaf between layers.

The Control Recipe is added to the ISA-TR88.00.02-2022 version and contains Process Variables and Ingredients. Furthermore, recipe numbers have been added to the Command Tags, to be able to indicate which recipe to execute on a unit/machine.

***Hint:** The Control Recipe parameters for a unit/machine need to be defined collaboratively with the Machine Supplier and End User. The process related parameters do not have to reflect the internal unit/machine parameters. A unit/machine can have several hundred or thousand internal parameters that do not have any interest for the End User. It is only the parameters that defines the main configuration of the unit/machine for a specific product.*

Hint: A set of control recipe parameters has a 1:1 relation to a production order. In some industries a production order can be called a “production run” or even just a batch.

Hint: From the view of the unit/machine it makes no difference, if the control recipe parameters are transferred from other system or just entered directly on the HMI interface of the unit/machine itself.

Hint: The following are useful or critical in determining the scope of the Control Recipe parameters:

- Quality relevant parameters in regard of process⁷
- Risk based evaluation which parameters can have a bigger impact on production

As an example Heat Sealing can have the process related unit/machine control recipe parameters: speed, time temperature, pressure, contact area, temperature of material, variation of foil thickness, material layer.

The process related unit/machine parameters that are really relevant and critical for the unit/machine can be identified through a Failure Mode and Effects Analysis (FMEA) working together with the End User experts.

Hint: The purpose of the Control Recipe is to be able to reprogramme or configure the unit/machine to make it flexible for the End User to introduce new products or setups.

⁷ For example, quality related parameters could be recipe name/numbers combined with other product specifying parameters (The unit/machine would have a local recipe management system that automatic loads all the required internal machine parameter set into the machine for the downloaded recipe name/number.)

5. IDENTIFYING AND DEFINING A UNIT

A unit/machine is defined as a collection of physical equipment and control functions that perform one or more major processing functions. A unit/machine can be a single machine or a subset of a whole packaging line.

A unit/machine is functionally or physically defined through a common unit/machine Interface. The PackML Interface State Manager provides a single communication interface between the HMI or other external control system and the unit/machine, as shown in Figure 7.

An error on a unit/machine may stop all the subsystems within the unit/machine and generate an alarm or warning. Often the safety circuit defines the boarder of a unit/machine in a packaging line.

If an error on a unit/machine only stops part of the subsystems and generates an alarm or warning, the individual subsystem could be defined as an individual unit, but this is not required.

Hint: A PackML Interface is defined for each part of a unit that handles a specific product. A unit/machine can only handle one production order at a time.

For example, a unit/machine can have 3 internal drive controllers but is perceived as one unit/machine from a user situation and operator access.

For example a depalletizer, may consist of infeed conveyers, a robot, outfeed conveyers and is seen as one unit/machine, as shown in Figure 6.

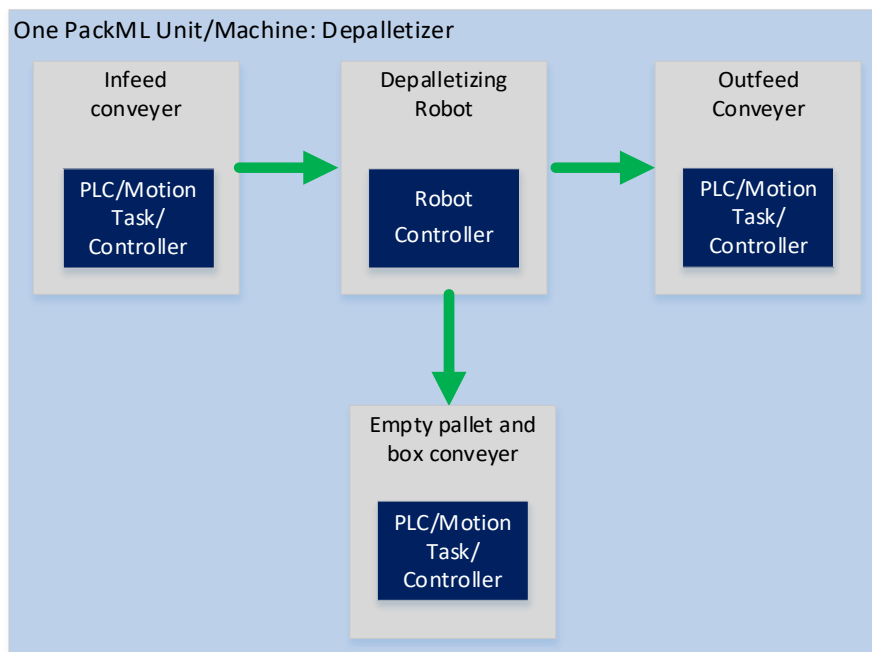


Figure 6: A PackML unit/machine made up of different system elements.

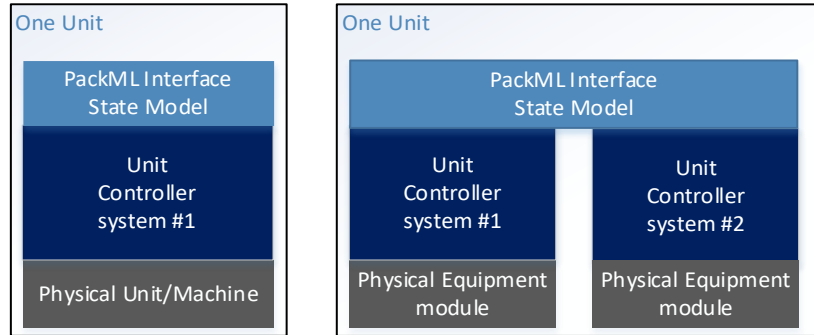


Figure 7: Example of units/machines mapping to one PackML interface, even when the unit/machine contains several controls systems.

For a unit/machine which executes more than one independent process, a PackML interface is required for each of the independent process. For example a palletizer with two independent packing cells in the same line, needs to have two PackML interfaces.

A unit/machine can implement the PackML State Manager Gateway (PackML GW) in two different ways; one where the software code for the PackML Gateway is part of the unit/machine, and the second where the PackML Gateway code is located and executed within an external system such as a PLC or micro-controller.

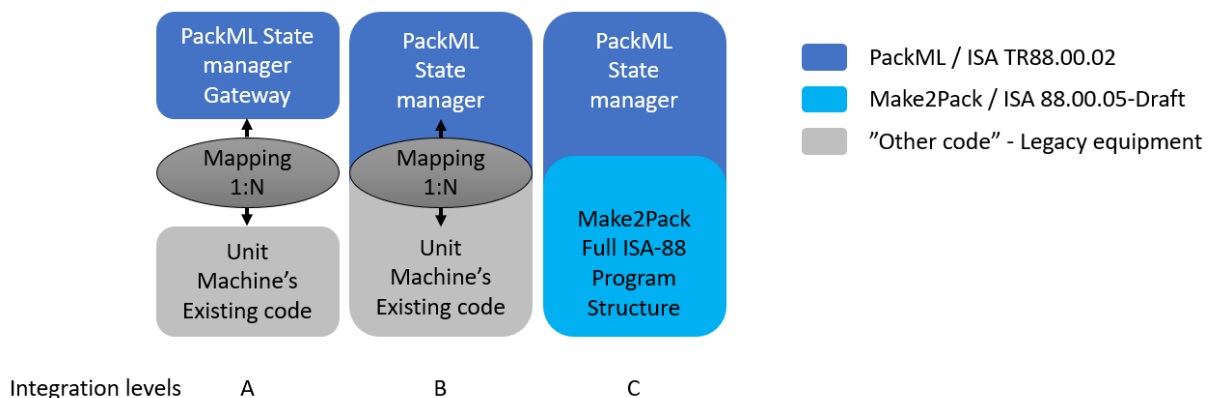


Figure 8 PackML implementation levels

There are three main types of PackML integrations levels as shown in Figure 8:

- Unit/machine integration level type A:**
 The unit/machine uses a PackML Gateway which runs either locally or externally – it will mainly expose the PackTags⁸. The unit/machine is controlled by an existing Machine State Model and the internal machine States are mapped to the PackML Interface State Model via the PackML Gateway. This means that the Machine Supplier needs to define the mapping between the machine existing code and the PackML Gateway. The PackML Gateway will communicate to other systems using the PackML interface and PackTags.
- Unit/machine integration level type B:**
 The unit/machine implements the PackML Gateway code and PackML Interface State Manager in the software code of the unit/machine's control system. The main difference

⁸ See section 0 of this document for an introduction to PackTags.

between integration level A and B is the location of the PackML gateway code. This implementation will expose the PackTags and the PackML state model. The PackML interface is to be mapped to the existing control code of the machine.

- **Unit/machine integration level type C⁹:**
The unit/machine program is structured according to PackML and ISA 88.00.05-Draft programming rules. See more details in Make2Pack [Ref. No. S88-5]. The unit/machine will communicate to other systems using the PackML interface PackTags. Full implementation of the PackML interface is integrated with the unit/machine program code.

6. THE PACKML INTERFACE STATE MODEL

6.1 THE SYNTAX OF THE PACKML STATE MODEL

The PackML Interface State model is based on two main elements:

- Commands
 - o A trigger that moves the unit/machine from one state to another (for example a pushbutton on the unit/machine or an external command sent via the network)
- States
 - o Acting states (a state where the unit/machine performs an action)
 - o Waiting states (a stable situation for the unit/machine). A wait state needs a command to enter the next state.

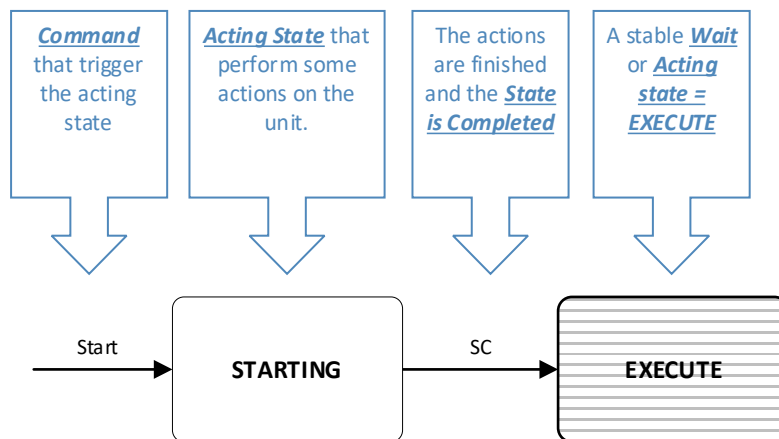


Figure 9 The syntax of the PackML State Model

⁹ It should be noted that Integration Level “C” may or may not support the full use of Make2Pack (M2P) based upon the specific template provided by the Technology Provider. Additionally, the Technology Providers could provide 2 templates for their OMAC PackML / TR88 template – one with Make2Pack and without Make2Pack!

6.2 THE PACKML INTERFACE STATE MODEL

The PackML Interface State Model in the ISA-TR88.00.02-2022 is more aligned with the handling of a physical unit/machine in relation to the state changes.

The PackML Interface State Model is a state model that represents the unit/machine states in a standardized manner. The interface description is based on a state model, a state description and related control commands. Figure 10 is the interface state diagram (ref. ISA TR88.00.02-2022).

Hint: It is recommended to implement all 17 of the PackML states in production mode. However, Machine Supplier and End User may agree to a deviation from the 17 states.

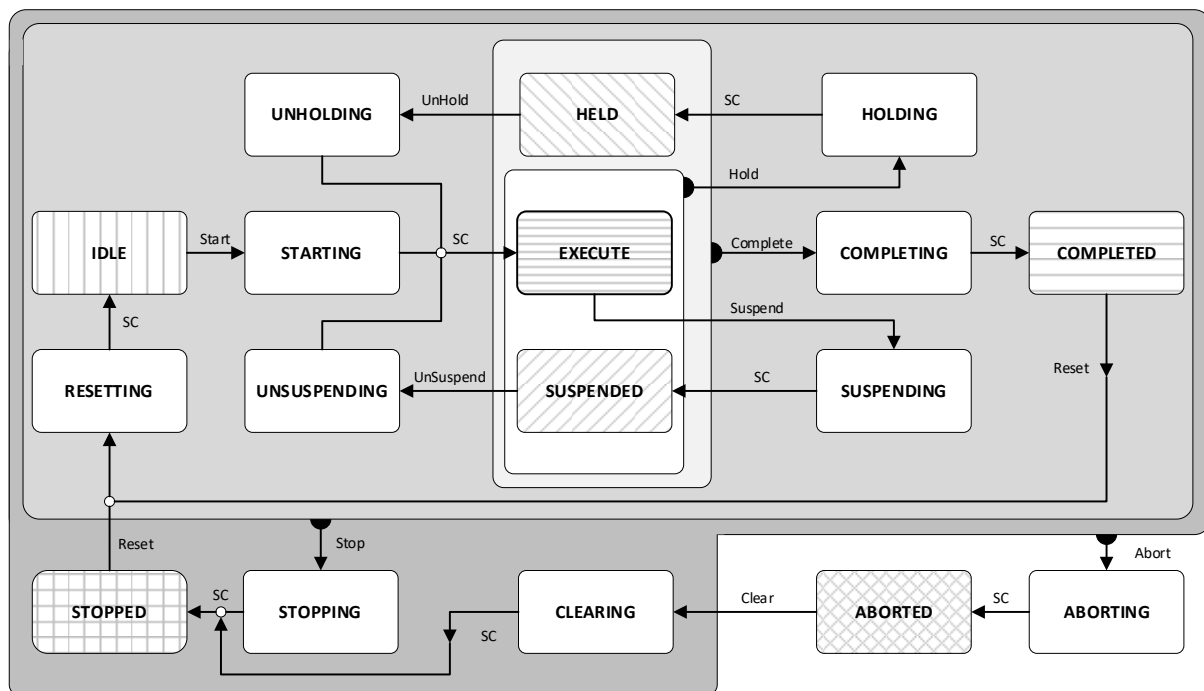






Figure 10: The PackML Interface State Model

Table 1: The Interface State Model with different states represented by different patterns

Pattern	State	Description
	EXECUTE	Acting State - The unit/machine is in a stable acting state - unit/machine is producing.
 	STOPPED IDLE COMPLETE	Wait State – A stable state used to identify that a unit/machine has achieved a defined set of conditions. In such a state the unit/machine is holding or maintaining a status until a command causes a transition to an Acting state. The unit/machine is powered and stationary.
	ABORTED	

Pattern	State	Description
	RESETTING STARTING SUSPENDING UNSUSPENDING COMPLETING HOLDING UNHOLDING ABORTING CLEARING STOPPING	Acting State – A state which represents some processing activity, for example ramping up speed. It implies the single or repeated execution of processing steps in a logical order, for a finite time or until a specific condition has been reached, for example within the Starting state the quality and validity of the received data is checked, before ramping up speed for execution.
 	HELD ABORTED	Wait state – A state which represents an error state on the Unit which will generate an alarm or warning. In this state the unit/machine is not producing, until the operator made a transition to the EXECUTING state. The state holds the unit/machine operations while material blockage are cleared, or safe correction of an equipment fault before the production may be resumed.
	SUSPENDED	Wait State – In this state the unit/machine is not producing any products. It will either stop running or continue to cycle without producing until external process conditions return to normal, at which time the SUSPENDED state will transition to the UNSUSPENDING state, typically without any operator intervention.

6.3 COLOURS USED FOR THE PACKML STATE MODEL

Colours can be used to represent different states in the PackML Interface Model, however it was decided to put all the colour guidance in a separate document: PackML User Interface – HMI.

An example of stacklight from the ISA TR88.00.02-2022 is given in section 6.7 *The PackML State Model and Stacklight*. Else further details on User Interfaces & HMI is given in the PackML guideline: *Part 6: PackML User Interface – HMI*.

6.4 THE PACKML STATE MODEL FROM A UNIT/MACHINE PERSPECTIVE

The figure below shows “Operator” intervention using HMI pushbuttons¹⁰:



The four pushbuttons have been added to the state model, as shown in Figure 11, to assist in the clarification of the unit/machine HMI and the PackML State Model relationship.

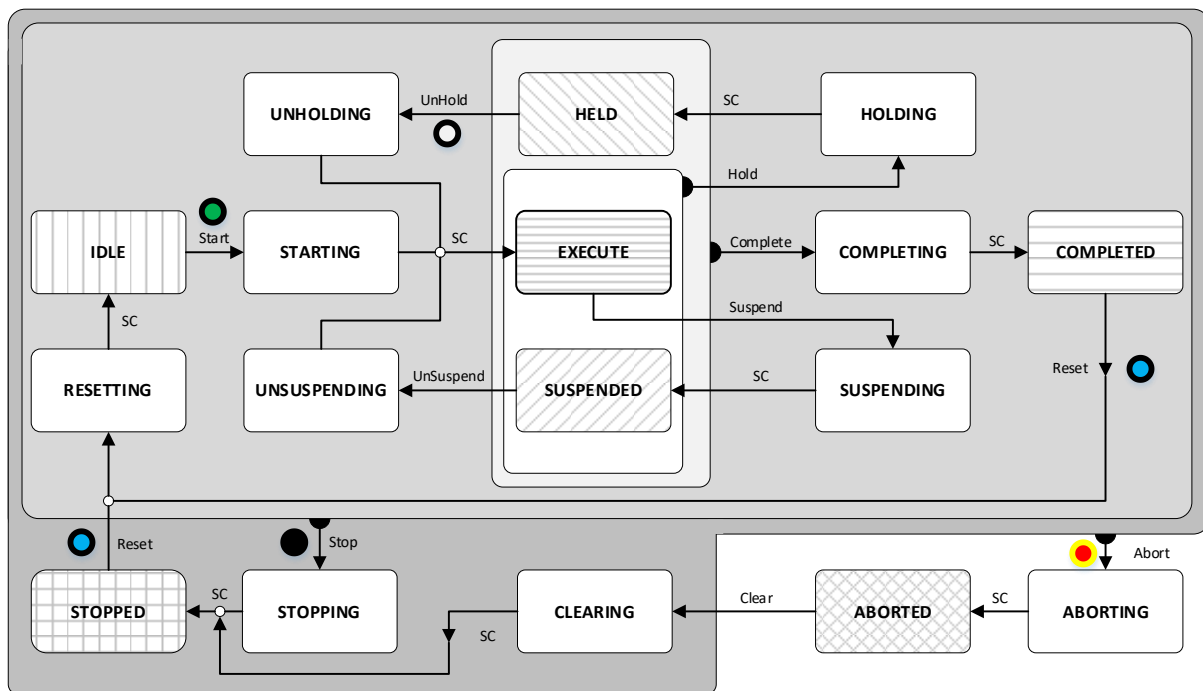


Figure 11: Pushbuttons for operating¹¹ the machine from HMI.

Further details on Pushbuttons is given in the PackML guideline: *Part 6: PackML User Interface – HMI*.

¹⁰ The colour coding for pushbutton control devices used on machines can be defined according to standard IEC 60204-1: 1997, 'Safety of Machinery':

- The colours for START should be WHITE, GREY or BLACK with a preference for WHITE. GREEN is also permitted. RED shall not be used.
- The colour YELLOW/RED shall be used for emergency stop.
- The colours for STOP should be BLACK, GREY or WHITE with a preference for BLACK. RED is also permitted, but it is recommended that RED is not used near an emergency operation device. GREEN shall not be used.
- WHITE, GREY or BLACK are the preferred colours for push-buttons actuators that alternately act as START and STOP push-buttons. The colours RED, YELLOW or GREEN shall not be used.
- WHITE, GREY or BLACK are the preferred colours for push-buttons that cause operation while they are actuated and cease the operation when they are released (e.g. hold-to-run). The colours RED, YELLOW or GREEN shall not be used.
- Reset push-buttons shall be BLUE, WHITE, GREY or BLACK. Where they also act as a STOP button, the colours WHITE, GREY or BLACK are the preferred with the main preference being BLACK. GREEN shall not be used.

¹¹ Operator intervention is required to make a transition from Aborted state to Clearing State. It requires a Clear command.

6.5 THE PACKML STATE MODEL FROM AN INTERNAL AND EXTERNAL PERSPECTIVE

Figure 12 shows the two different stop situations when the unit/machine is producing, a HELD situation and a SUSPEND situation:

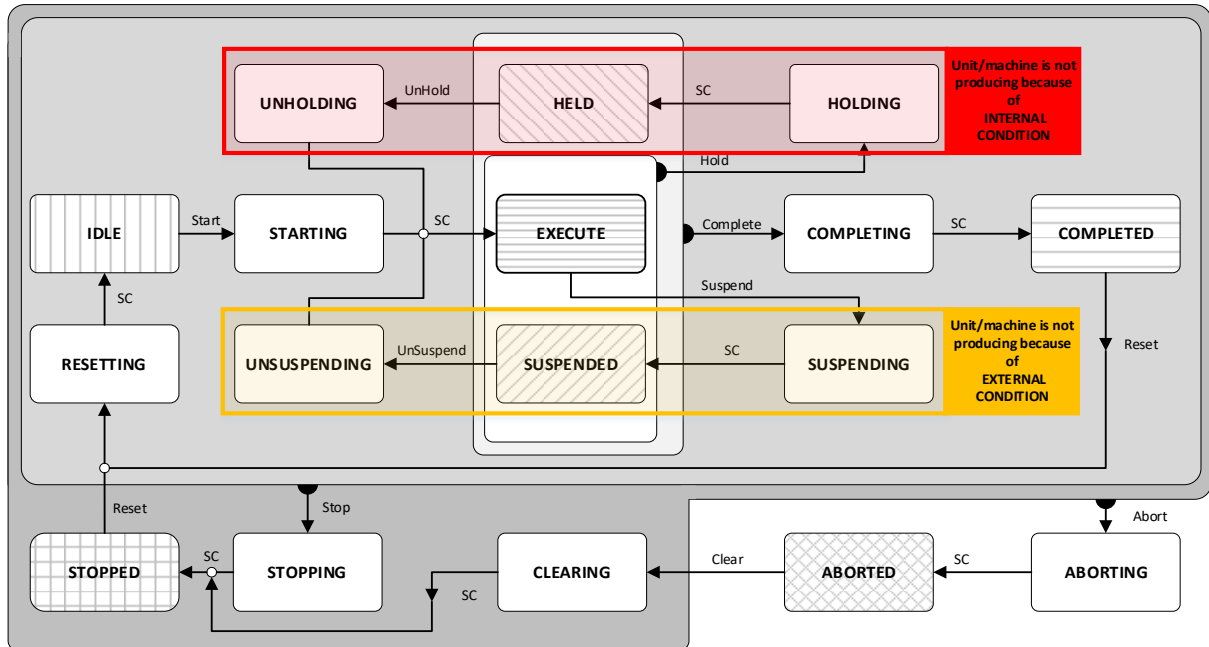


Figure 12: The difference between HELD and SUSPEND in mode Production.

HOLDING
HELD
UNHOLDING

These states shall be used when INTERNAL unit/machine conditions do not allow the unit/machine to continue producing. This happens when the unit/machine leaves EXECUTE or SUSPENDED due to internal conditions. Note that internal machine conditions are conditions inside the unit/machine and not from other machines on the production line. The HELD state is typically used for routine machine conditions that requires minor operator servicing in order to continue production. The HOLDING state can be initiated automatically or by an operator and can easily be recovered from.

An example of this would be a unit/machine that requires an operator to periodically refill a glue dispenser or carton magazine and due to the machine design, these operations cannot be performed while the unit/machine is running. Since these types of tasks are part of production operations, it is not desirable to go through aborting or stopping sequences and because these functions are integral to the unit/machine they are not considered to be “external”. While in the HOLDING state, the machine is typically brought to a controlled stop and then transitions to HELD upon state complete.

Hint: To be able to restart production correctly after the HELD state, all relevant process set-points and status information at the time of receiving the HOLD command must be saved in the unit/machine controller when executing the HOLDING procedure.

Hint: An event that caused transition from *EXECUTE* to *HOLDING* may need to be reset in the *UNHOLDING* state.

SUSPENDING
SUSPENDED
UNSUSPENDING

These states shall be used when *EXTERNAL* unit/machine conditions do not allow the unit/machine to continue processing. External process conditions are conditions outside the unit/machine but usually within the same integrated production line.

This happens, when the unit/machine leaves *EXECUTE* due to upstream or downstream conditions on the line. This is typically due to a blocked or starved event. This condition may be detected by a local machine sensor or based on a supervisory system external command. While in the *SUSPENDING* state, the unit/machine is typically brought to a controlled stop and then transitions to *SUSPENDED* upon state complete.

Hint: To be able to restart production correctly after the *SUSPENDED* state, all relevant process set-points at the time of receiving the *SUSPEND* command must be saved in the unit/machine controller when executing the *SUSPENDING* procedure.

Hint: An event that caused transition from *EXECUTE* to *SUSPENDING* may need to be reset in the *UNSUSPENDING* state.

6.6 THE PACKML STATE MODEL FROM A UNIT/MACHINE PARAMETER PERSPECTIVE – PRODUCTION ORDERS

Figure 13 illustrates the information exchange in relation to the individual states and PackTags types. The black boxes at the top of the figure indicate the information to be exchanged. The blue arrows represents data related to PackML Status Tags and the red arrows represent data related to PackML Command Tags. The green arrows indicate the information related to the PackML Admin Tags.

The communication of data to a unit/machine consists of the following types of data and the data is identified by PackTags types:

- Command Tags – Used for control of the unit/machine.
- Status Tags – Used to obtain status information from the unit/machine.
- Admin Tags – Used to obtain performance information from the unit/machine.

Table 2 Data types and PackTag types

Data type	PackTag type
Mode & State	Status Tags
Warning & Alarms	Admin Tags
Performance data - OEE data, Counters for processed, defective, consumed products	Admin Tags
Command - Reset, Start, Stop, Complete, etc.	Command Tags
Parameters (Machine parameters)	Command Tags Status Tags
Recipe (Control Recipe parameters)	Command Tags Status Tags

The control logic and synchronization of units is managed through Command Tags. Commands are used to request a change of the state of a unit. The commands can be triggered by either an internal unit/machine condition, the unit/machine HMI (pushbuttons), or they can be received from an external system.

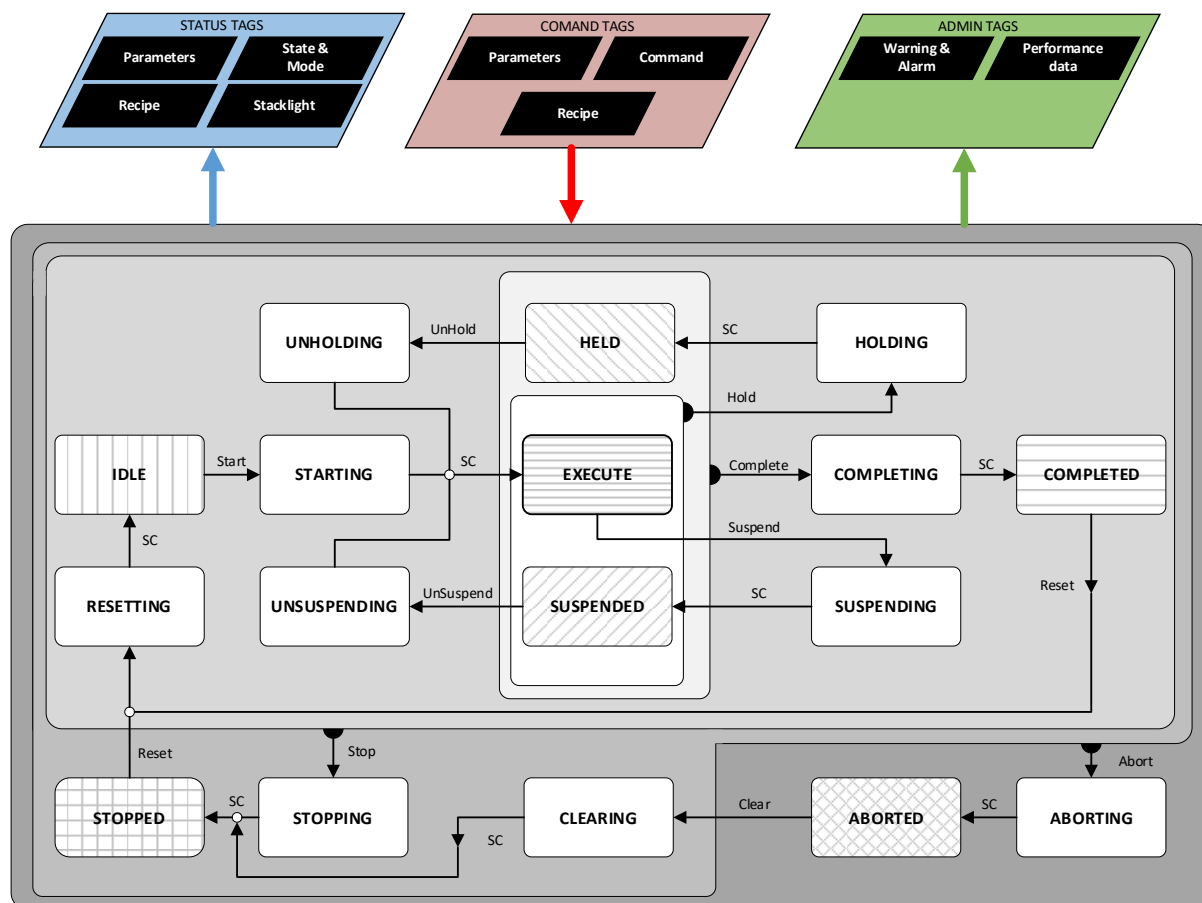


Figure 13 The PackML State Model from a unit/machine parameter perspective

Hints: In the design of a PackML Interface State Manager it is recommended to specify which information and when the information is to be exchanged between the PackML Interface State Manager and the Supplier dependent Machine State Manager.

To make a simplification of the communication and mapping, it is recommended to specify when information can be sent and received between the two State Managers and in addition, to specify what information can be exchanged in the different states. For example, Command Tags Parameters are only exchanged between the PackML Interface State Manager and the Supplier dependent Machine State Manager in IDLE or STOPPED state. Normally, relevant unit/machine parameters can also be changed by operator.

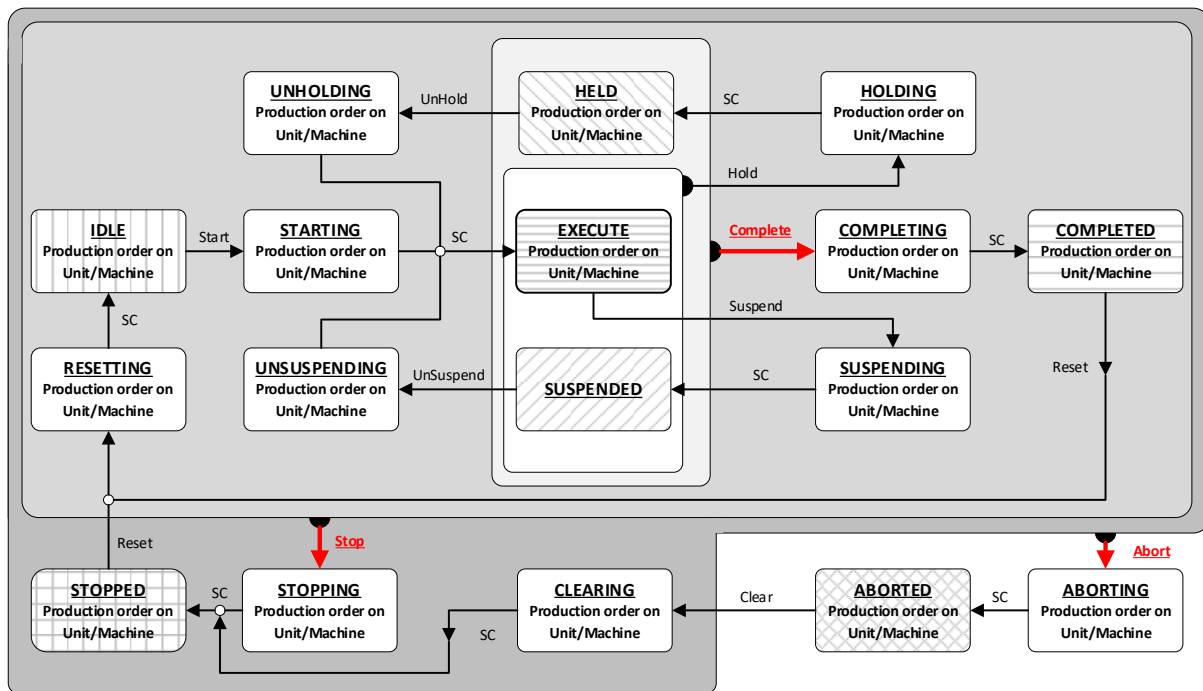


Figure 14 PackML State Model from a Production Order¹² perspective

The PackML base state model has its origins in the batch industry (See Ref. No. S88-1), and the state model indicates the change in the execution of a Production Order (Batch).

There are three ways to bring the Production Order to an end:

- Through a Stop command: this means that the currently produced products can be used. The production order is stopped.
- Through a Complete command: this means that the current Production Order have reached its end by a defined criterion (e.g., number of produced products). The production order is completed.
- Through an Abort command: this means that currently produced products may be destroyed. The production order is terminated.

¹² The Production Order is used, as it represents the handling of products within the unit/machine as defined within reference ISA-88.00.01.

6.7 THE PACKML STATE MODEL AND STACKLIGHT

A Status Tag Status.Stacklight[#] is used to indicate the Stacklight status related to the individual PackML State. The Status.Stacklight[#] Tag is a DINT, and the individual bit location are related to a specific stacklight colour and function (i.e. Solid or Flashing)

Bit No.	Stacklight function
0	Red Solid
1	Red Flashing
2	Amber Solid
3	Amber Flashing
4	Blue Solid
5	Blue Flashing
6	Green Solid
7	Green Flashing
8	Horn Solid
9	Horn Flashing
10..31	User-Defined

Within the ISA TR88.00.02-2022 is an example of how the stacklight can be made to correspond to certain machine conditions and PackML States. The example is in accordance with the standard IEC 60073 and fits also to OMAC guideline Part 6: PackML User Interface – HMI.

	Aborting	Clearing	Aborted	Complete	Stopped	Resetting	Idle	Starting	Execute	Completing	Holding	Held (No Production)	Unholding	Suspending	Suspended	Unsuspending	Stacklight	Status.Stacklight[#]
Abnormal Stop	F	F	F														Red Lamp Flashing	1
Controlled Stop				S	S	S	S										Red Lamp Solid	0
Starved Upstream														F	F		Amber Lamp Flashing	3
Blocked Downstream														S	S		Amber Lamp Solid	2
Low Material	F	F	F	F	F	F	F	F	F	F			F	F	F	F	Blue Lamp Flashing	5
Material Exhausted										S	S						Blue Lamp Solid	4
Ready to Start							F										Green Lamp Flashing	7
Running								S	S	S			S			S	Green Lamp Solid	6
Starting / Restarting								F					F			F	Horn Flashing	9
Not used																	Horn Solid	8
Not used																	User-Defined	10..31

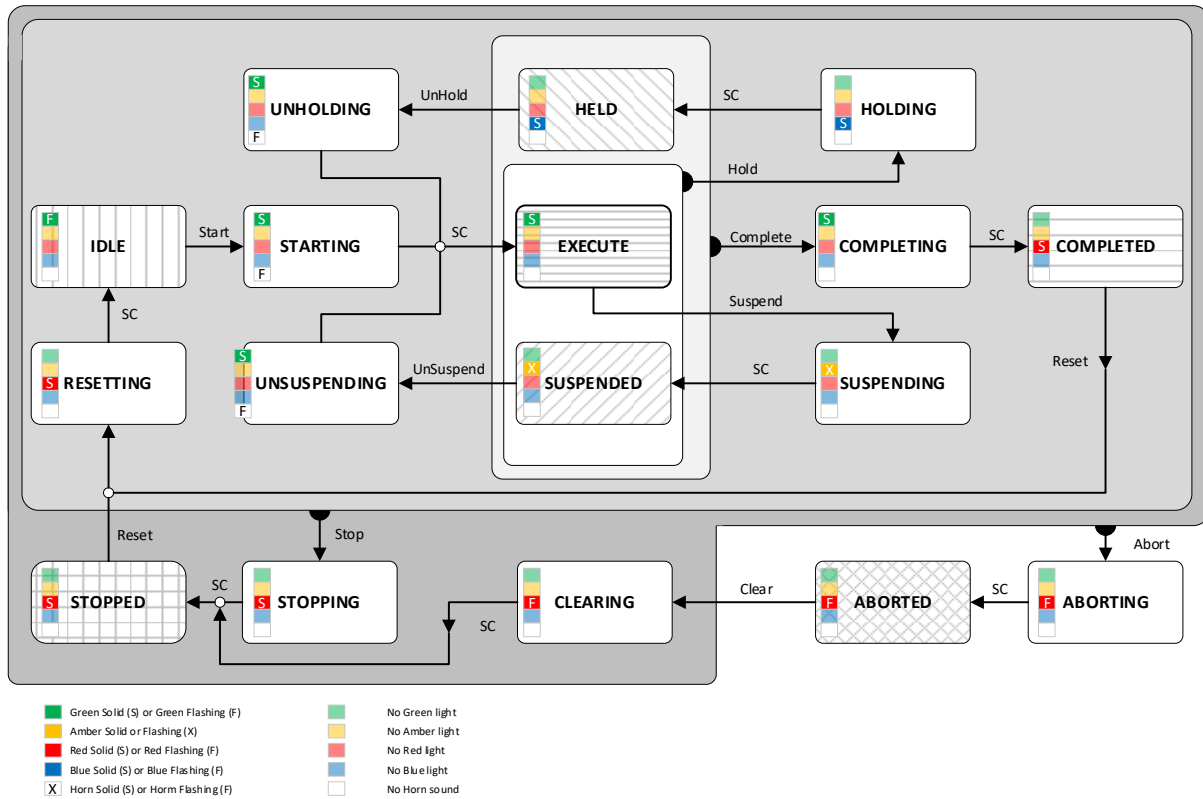


Figure 15 PackML State Model and Stacklight example.

7. PACKML EVENT STATE MANAGER

It is possible to create a matrix that allows Machine Suppliers to configure what types of unit/machine events will lead to a PackML Interface State Change and vice versa. This can be determined when a unit/machine should switch to a specific state in the PackML Interface State Model.

For example, when the unit/machine makes an internal state change and maps to the PackML interface states: HELD, STOPPED, SUSPENDED and ABORTED, the End Users may have different interpretations and requirements for the mapping. The tables in the following parts of Section 7 illustrate the mapping between the PackML Interface State Manager and an example Supplier dependent Machine State Manager. The blue colour and the left side of the tables represent the PackML Interface State and the grey colour and right side of the tables represent the Supplier dependent Machine State, matching the colour scheme in Figure 16. Below the tables a set of figures illustrates a possible mapping of all Acting PackML Interface States into example Supplier dependent Machine States. The Supplier dependent Machine State is illustrated with a standard PackML state model, however, be aware that the state model of a Machine Supplier might be rather different.

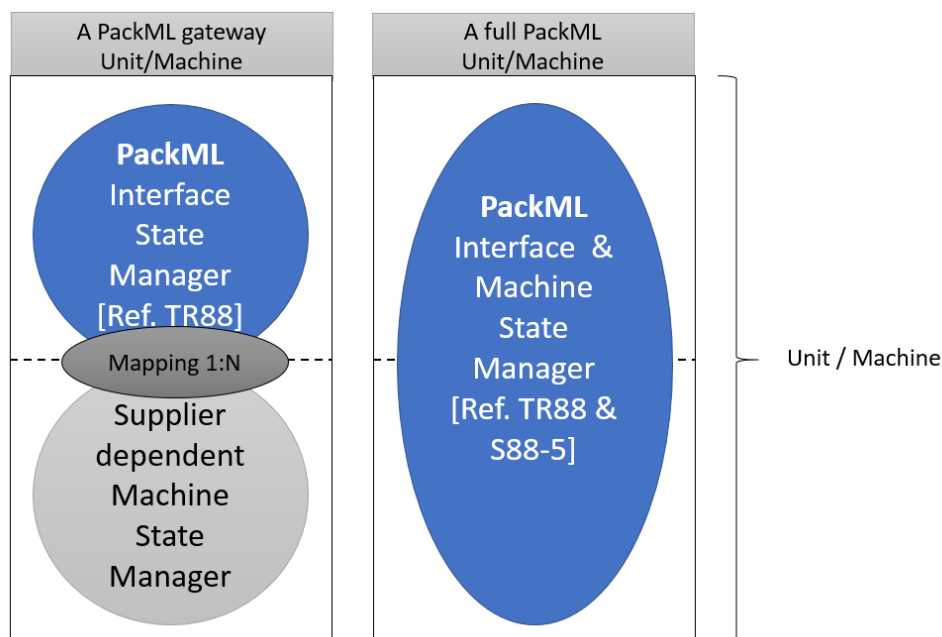



Figure 16: The mapping of PackML Interface State Manager and PackML different Machine State Manager (Supplier dependent or PackML)

Hint: The buyer and the seller of a machine have to agree on when the machine has to be in each PackML interfaces state. It is recommended to use a matrix with the PackML Interface States as one dimension and then the current machine dependent machine states as the second dimension of the table or as a second dimension uses the unit/machine tags need to be mapped to the current PackML state. Mark all possible situations for being in the different PackML Interface states. See more details of a sample matrix in section 7.11

7.1 RESETTING

Table 3 Interpretation of the Resetting state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
RESETTING	<p>The PackML interface RESETTING state will start the resetting process, which typically causes the unit/machine to clear data and to place the unit/machine in an IDLE state, where unit/machine components are energized waiting for a Start command. By resetting the Unit, the machine parameters will be cleared, and the unit/machine is ready for new machine parameters. For example, the operator has changed tooling and by activating the Resetting process the operator have indicated that the machine is ready for production.</p> <p>Actions to be taken during RESETTING are to be specified by the Machine Supplier and End User.</p>	Resetting	<p>This state is the result of a RESET command from the STOPPED or COMPLETE state. Faults and stop causes are reset. RESETTING will typically cause safety devices to be energized and place the machine in the IDLE state where it will wait for a START command.</p> <p>RESETTING will typically reset alarms caused by actuated safety devices after the unit/machine has been reset.</p> <p>No hazardous motion should happen in this state.</p> <p>This acting state is triggered by a reset request  Machine movement such as “re-homing”¹³ after a stop may occur in this state.</p>

¹³ It is essential that the clearing of data is carried out at the right time for the Supplier dependent Machine State Manager to ensure that the unit/machine does not physical start with the tooling and components for the wrong production order.

PACKML INTERFACE (RESETTING)

The Production order/Batch is being Reset, and a new Production order/Batch can be started.

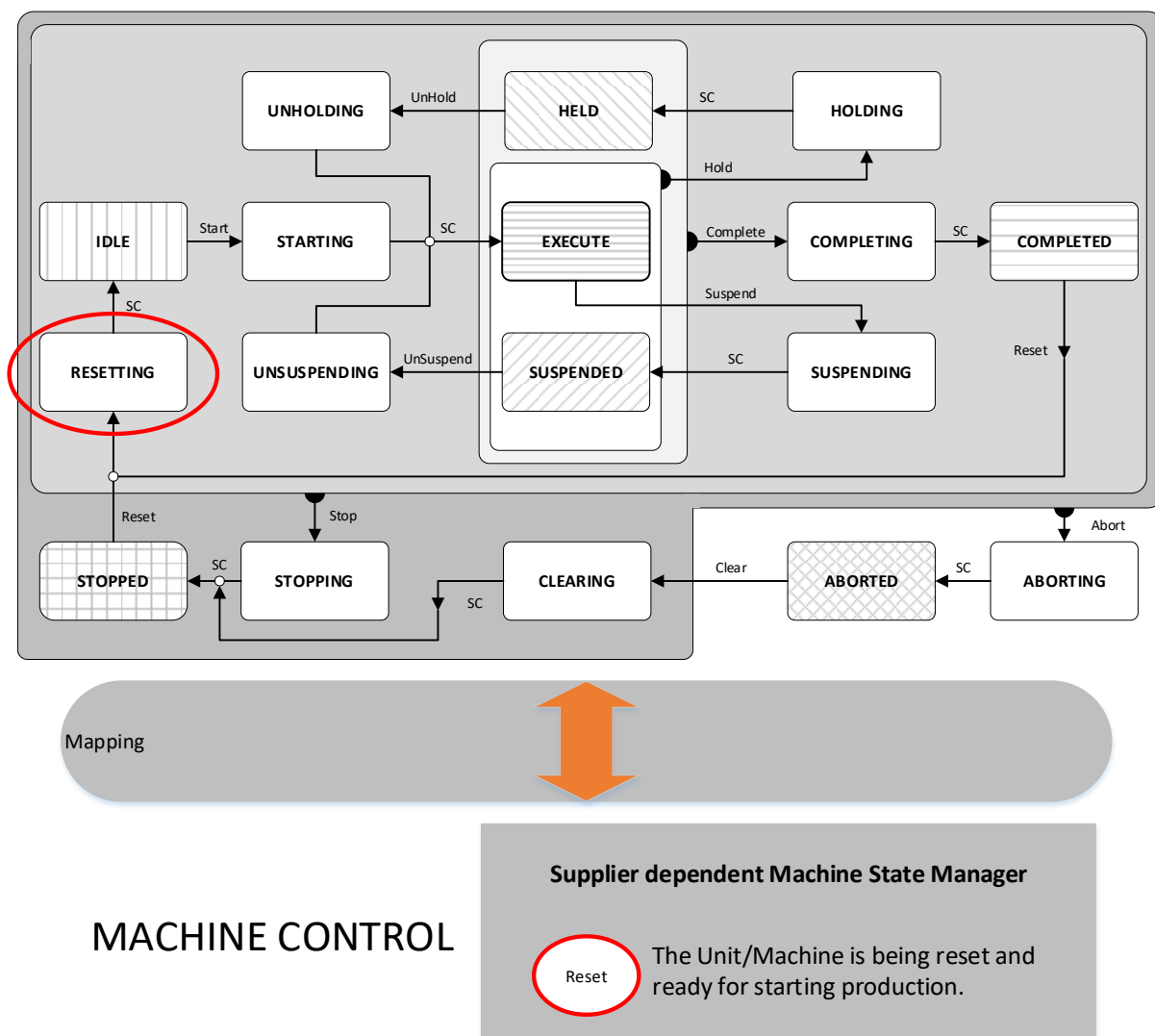



Figure 17 Reset mapping¹⁴ between the PackML Interface State Model and the Machine control State model.

¹⁴ A reset on machine/unit is not an “end of a Production Order” on the Supervisory Control Level and the other units in a line. Downstream units may still be executing the Supervisory Control Level production order.

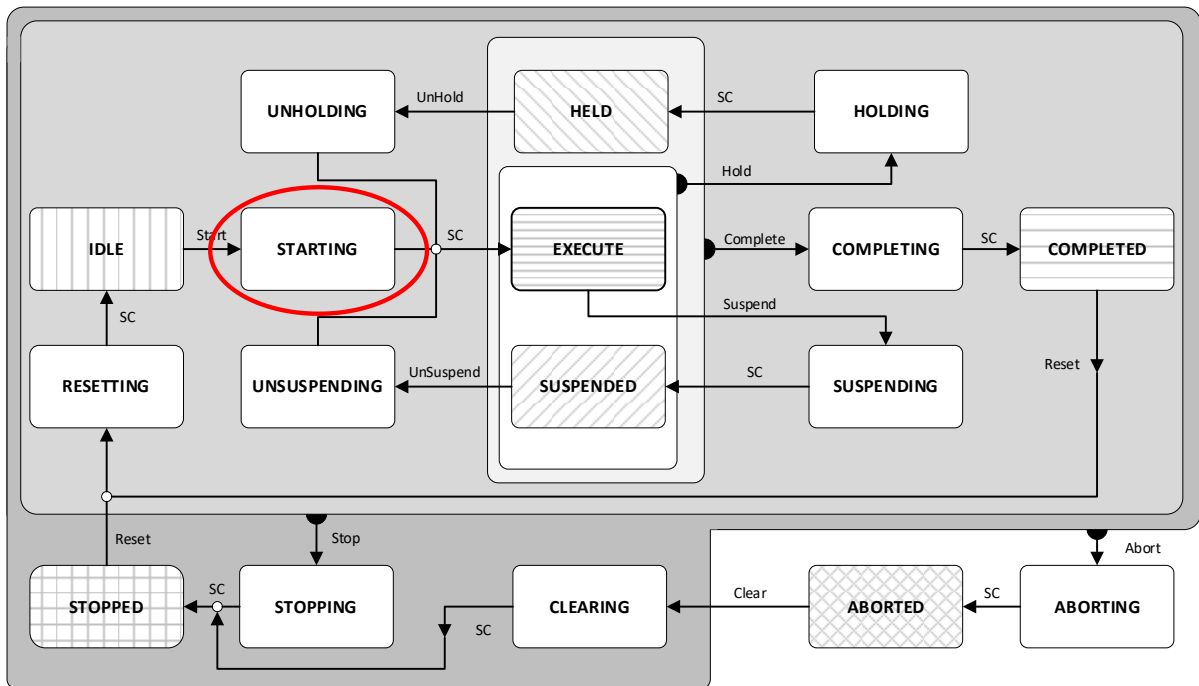
7.2 STARTING

Table 4 Interpretation of the Starting state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
STARTING	<p>This state provides the steps needed to be able to start the unit/machine and is triggered by a Start command. The starting logic will check the unit/machine parameters and move the unit/machine into the state EXECUTE if parameters has been accepted. The starting logic will ramp up speed, pressure, etc. to be ready for production. However, if the parameters are not valid, the starting logic will move the unit/machine into the STOPPING state and the unit/machine will end in STOPPED state.</p> <p>If the machine is not able to complete the Starting state process (internal error), the unit/machine will enter the STOPPING state and end in STOPPED state.</p> <p>Actions to be taken during STARTING are to be specified by the Machine Supplier and End User.</p>	Starting	<p>The machine completes the steps needed to start. This state is entered as a result of a Start command (local or remote). Following this command, the machine will begin to “execute”.</p> <p>As a result of requesting a starting request  to the EXECUTE state.</p>

PACKML INTERFACE (STARTING)

The Production order/Batch is being started.



Mapping



MACHINE CONTROL

Supplier dependent Machine State Manager

Starting

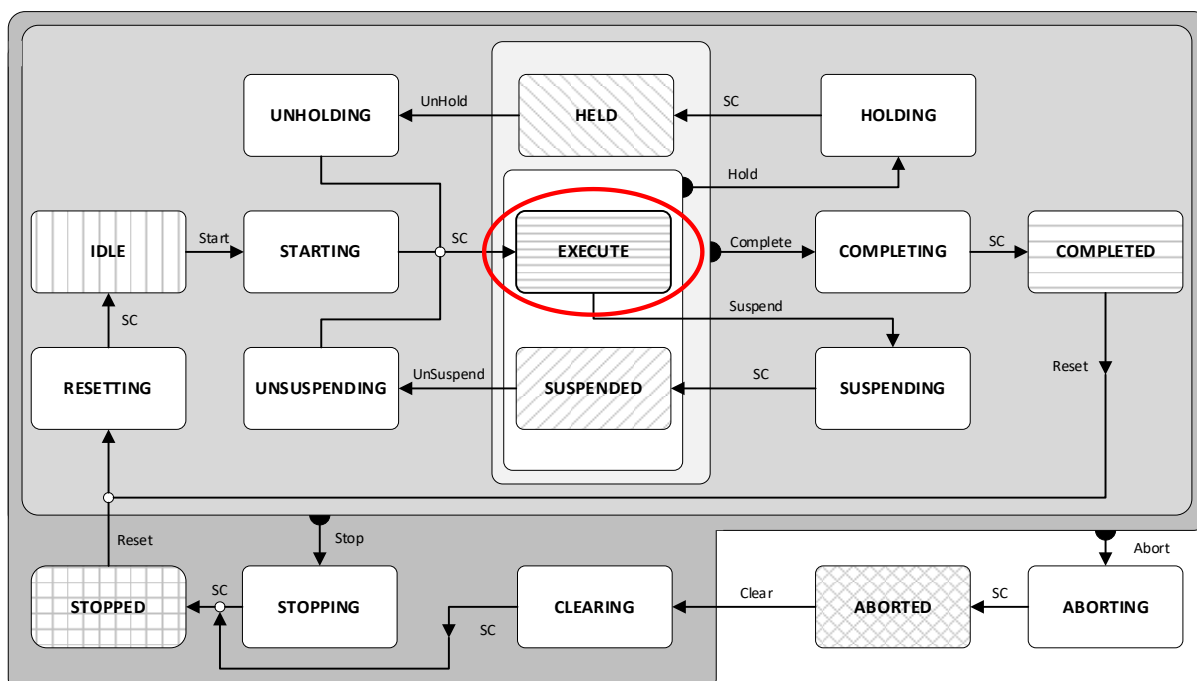
The Unit/Machine is being started and ready execute the production order/batch.

Figure 18 Starting mapping between the PackML Interface State Model and the Machine control State model

7.3 EXECUTE

PACKML INTERFACE (EXECUTE)

The Production order/Batch is being executed.



Mapping



MACHINE CONTROL

Supplier dependent Machine State Manager

Execute

The Unit/Machine is running and products are being produced.

Table 5 Interpretation of the Execute state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
EXECUTE	The unit/machine is running with all conditions met, defined by the selected MODE and / or recipe selected.	Execute	When the machine is processing materials, it is in the EXECUTE state. Different machine modes will result in specific types of EXECUTE activities. For example, if the machine is in the "Production" mode, the EXECUTE will result in products being

			<p>produced. The machine is producing product at the desired set speed.</p> <p><i>HINT: It is desirable to store set speed as a control recipe parameter within a product “recipe” to avoid speed losses due to unnecessary blocking and starvation.</i></p>
--	--	--	--

c

Figure 19 Execute mapping between the PackML Interface State Model and the Machine control State model

7.4 HOLDING

Below are three machine control scenarios that can bring the PackML Interface State Model into the HOLDING state. The unit/machine can be in the HOLDING state, because of one of the following reasons:

- 1) Minor issue caused by a hold situation on the Unit/Machine
- 2) A stop of the Unit/Machine which does not prevent continuing the current production order
- 3) Or a major issue (ABORTING) that requires a Unit/Machine safety stop but does not prevent continuing the current production order

Normally, a unit/machine only supports one of the three scenarios.

Table 6 Interpretation of the Holding state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
HOLDING	<p>The HOLDING logic brings the unit/machine to a stop or to a state which represents HELD for the particular Unit.</p> <p>This is an internal control logic that is executed when an error¹⁵ occurs on the unit/machine or an operator initiates a Hold command from unit/machine HMI. The Holding control logics change the unit/machine from the Holding state to the HELD state.</p>	Holding ¹⁶	<p>This state is to be used when INTERNAL (inside this unit/machine and not from another machine on the production line) machine conditions do not allow the machine to continue producing, and the machine leaves EXECUTE or SUSPENDED due to internal conditions, but does not prevent continuing the current production order.</p> <p>This is typically used for routine machine conditions that requires minor operator servicing to continue production. This state can be initiated automatically or by an operator and can easily be recovered from. An example of this would be a machine that requires an operator to periodically refill a glue dispenser or carton magazine and due to the machine design, these operations cannot be performed while the machine is running. Since these types of tasks are normal production operations, it is not desirable to go through aborting or stopping sequences, and because these functions are integral to the machine they are not considered to be “external”. While in the HOLDING state, the machine is typically brought to a controlled stop and then transitions to HELD upon state complete. To be able to restart production correctly after the HELD state, all relevant process set-points and return status of the procedures at the time of receiving the <i>HOLD</i> command must be saved in the machine controller when executing the HOLDING procedure.</p>

¹⁵ Not all machine errors will cause the machine to transition to Holding state. For example if an error does not influence the production on the unit/machine, the unit/machine should not be put into the PackML interface state HOLDING. The OEMs can determine in a table, which unit/machine errors and events should put the unit/machine into a certain PackML Interface State. The errors and events should be mapped to alarm and warning information.

¹⁶ The mapping between the PackML Interface States and an example Supplier dependent Machine State Manager is defined within section 7.11.

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
		Stopping ¹⁶	The machine is powered and stationary after completing the stopping state but can continue the current production order after the stop condition is corrected. All communications with other systems are functioning, if applicable. As a result of requesting a Controlled stop to a stopped state.
		Aborting ¹⁶	The aborting state can be entered at any time in response to an abort command or on the occurrence of a machine fault. The aborting logic will bring the machine to a rapid safe stop, but can continue the current production order after the abort condition is corrected.

PACKML INTERFACE (HOLDING from internal hold)

The reason for moving a unit/machine into the HOLDING state can be:

- 1) Minor issue caused by a hold situation on the Unit/Machine

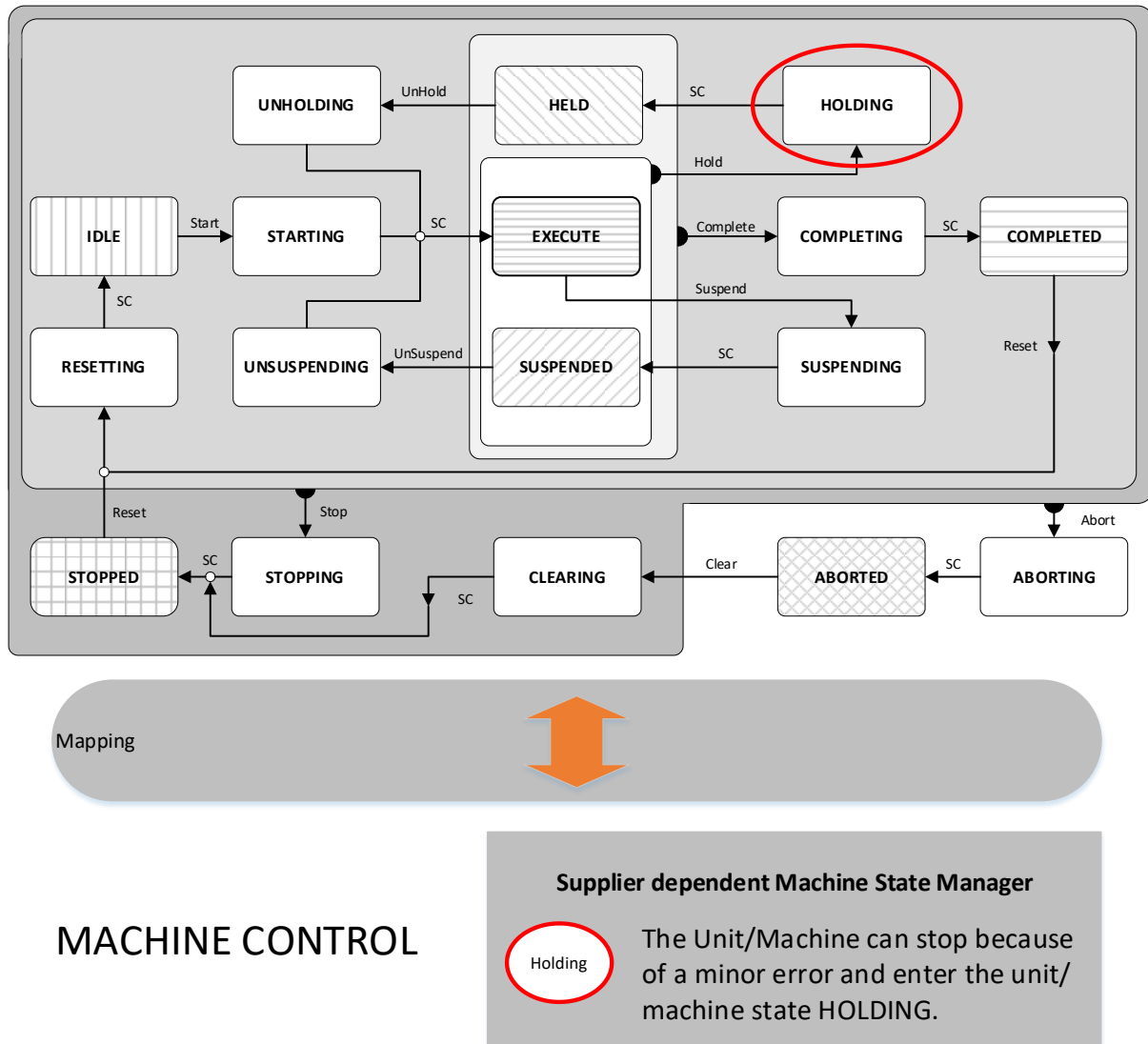
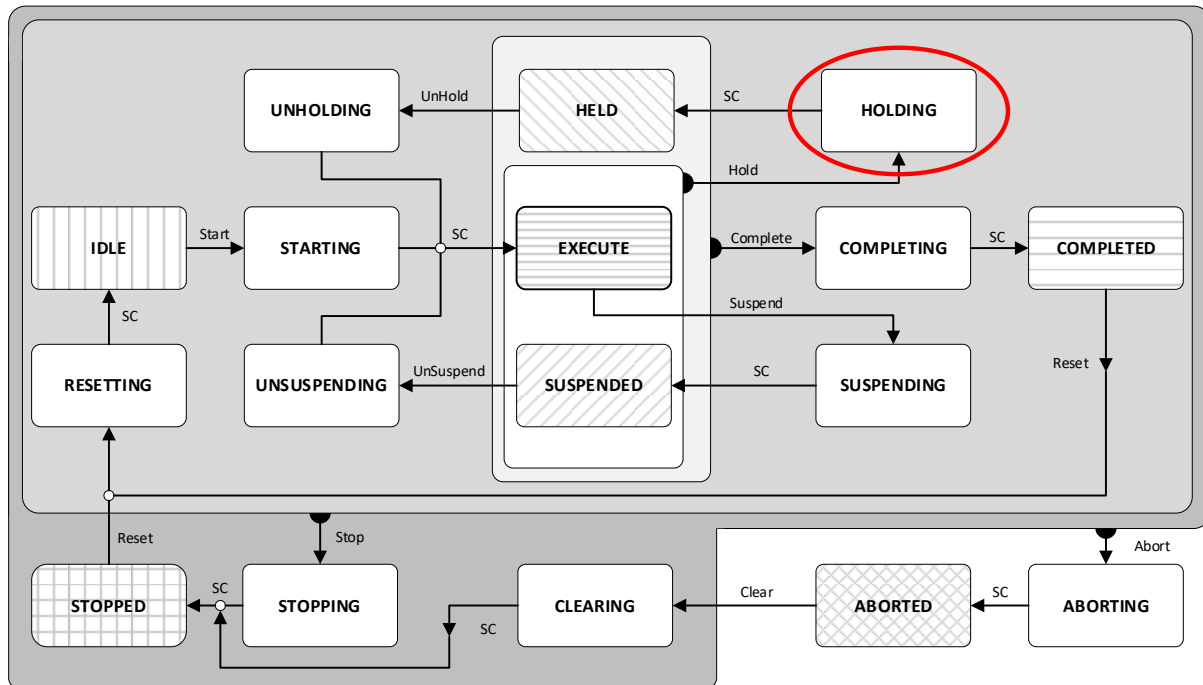


Figure 20 Hold mapping between the PackML Interface State Model and the Machine control State model

PACKML INTERFACE (HOLDING from internal stop)

The reason for moving a unit/machine into the HOLDING state can be:

2) A stop of the Unit/Machine



Mapping



MACHINE CONTROL

Supplier dependent Machine State Manager

Stopping

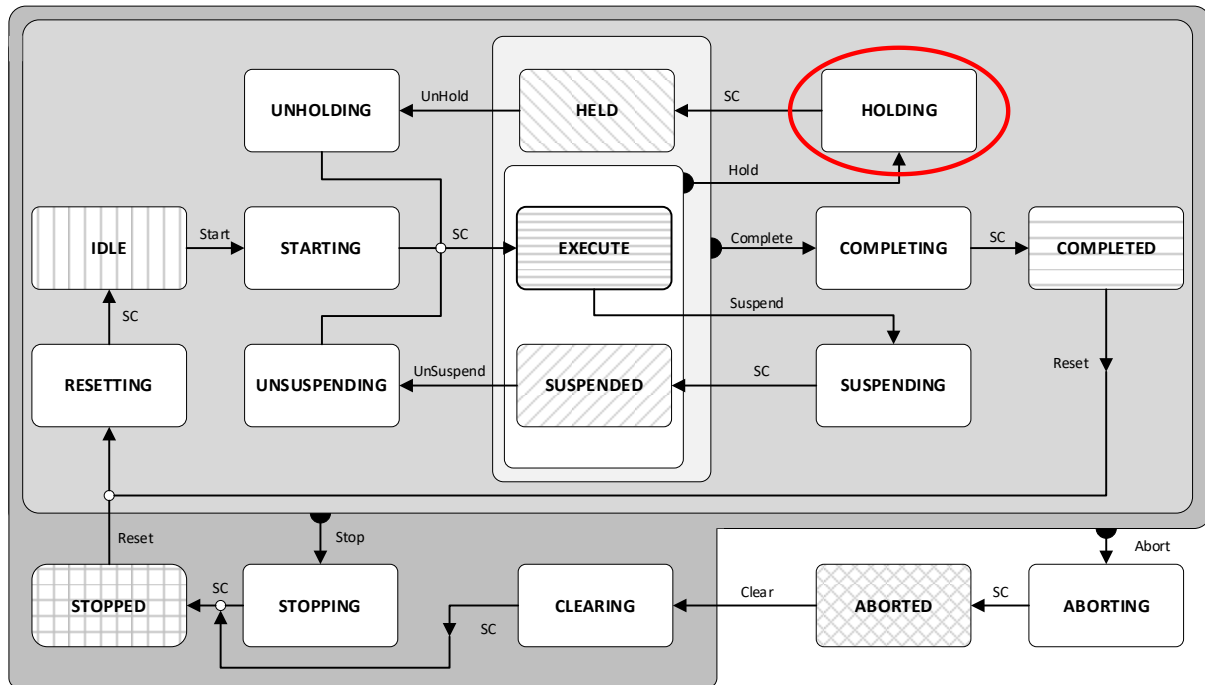
The Unit/Machine can stop because the operator stops the unit and the unit/machine enter the state STOPPING.

Figure 21 Stopping mapping between the PackML Interface State Model and the Machine control State model

PACKML INTERFACE (HOLDING from an internal abort)

The reason for moving a unit/machine into the HOLDING state can be:

3) Or a major issue (ABORTING) that requires a safety stop



Mapping



MACHINE CONTROL

Supplier dependent Machine State Manager

Aborting

The Unit/Machine can stop because the operator pressed an E-stop and the unit/machine enter the state ABORTING.

Figure 22 Aborting mapping between the PackML Interface State Model and the Machine control State model

7.5 UNHOLDING

The UNHOLDING scenarios are related to the way that the unit/machine was entering the HOLDING state. The HOLDING state has three different scenarios and therefore, there are also three scenarios to the UNHOLDING process. The UNHOLDING process will move the unit/machine into EXECUTE state.

- 1) UNHOLDING can be carried out from the Machine Control just by restarting (Unhold) the unit/machine.
- 2) The Machine Control system was original stopped, and needs to be restarted on the same Production Order.
- 3) The Machine Control system was original E-stopped and needs to be cleared and restarted on the same Production Order.

Table 7 Interpretation of the UNHOLDING state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine States	Machine State description (Supplier dependent Machine State Manager)
UNHOLDING	The UNHOLDING state is a response to an Operator command to resume production. The control logics in the UNHOLDING state prepares the unit/machine to re-enter the normal EXECUTE state. The UNHOLDING logic will ramp up speed, pressure, etc. to be ready for production.	Unholding	Refers to HOLDING for when this state is used. A machine will typically enter into UNHOLDING automatically when INTERNAL conditions, material levels, for example, return to an acceptable level. If an operator has been required to perform minor servicing to replenish materials or make adjustments, then the <i>UNHOLD</i> command may be initiated by the operator.
		Reset Idle Starting	The operator needs to handle the actions on the Machine Control panel (HMI) to return to the EXECUTE state. The operator has made a controlled stop of the unit/machine, and has to restart on the same Production Order after this situation.
		Clearing Stopped Reset Idle Starting	The operator has originally activated the safety circuit and the unit/machine is making a fast stop. The operator needs to clear the safety circuit and handle the actions on the Machine Control panel (HMI) to return to the EXECUTE state. The operator has made an E-stop of the unit/machine, and has to clear E-stop circuits and restart after this situation on the same Production Order

PACKML INTERFACE (UNHOLDING from unholding)

The execution of a Production order/Batch is UNHOLDING because of problem on the unit/machine is fixed. Normally a minor problem have been solved. The Machine Controller have not been stopped.

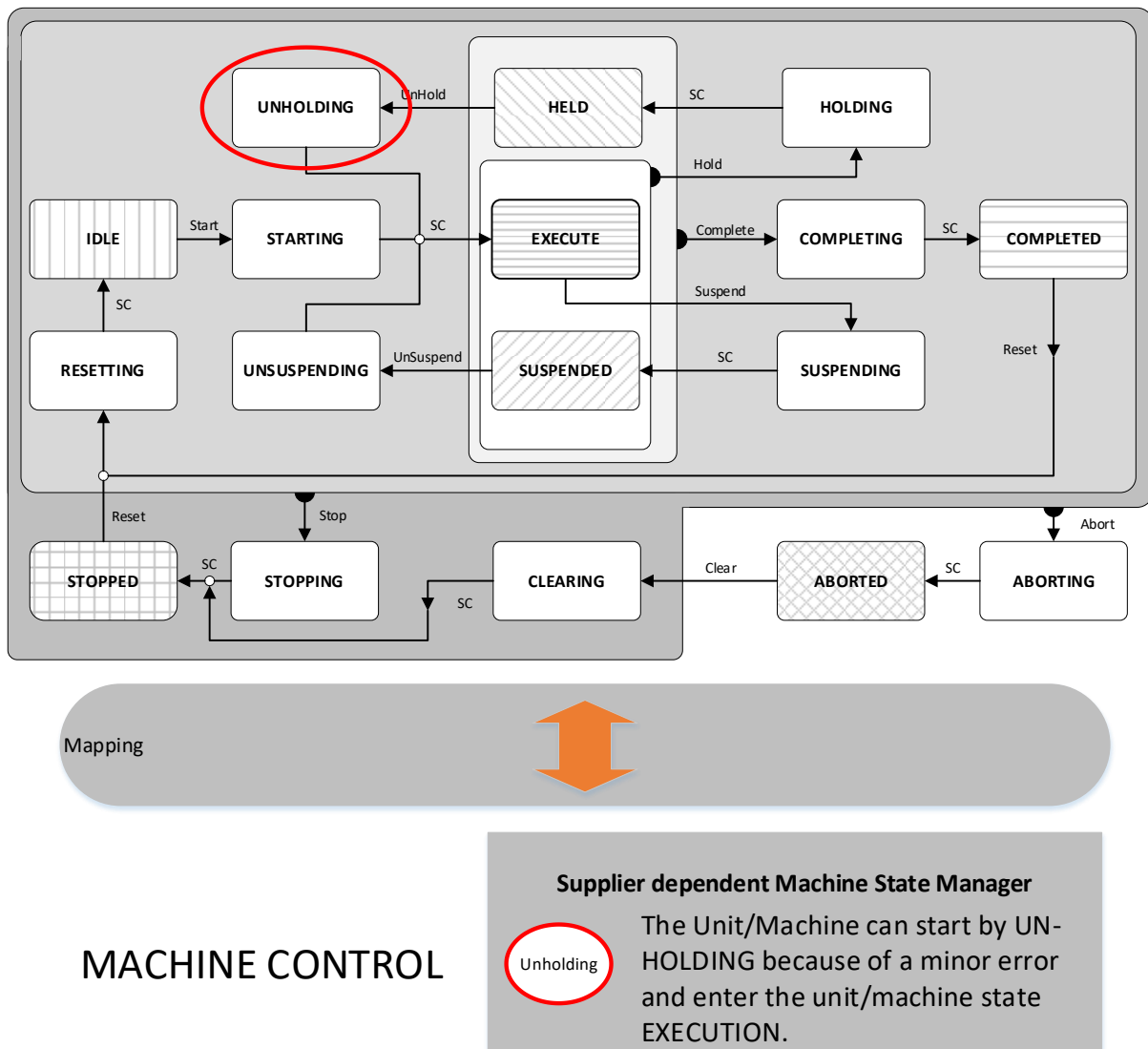


Figure 23 Unhold mapping between the PackML Interface State Model and the Machine control State model

PACKML INTERFACE (UNHOLDING from resetting, idle, and starting)

The execution of a Production order/Batch is UNHOLDING because of problem on the unit/ machine is fixed. The Machine Controller have been stopped and needs to be restarted.

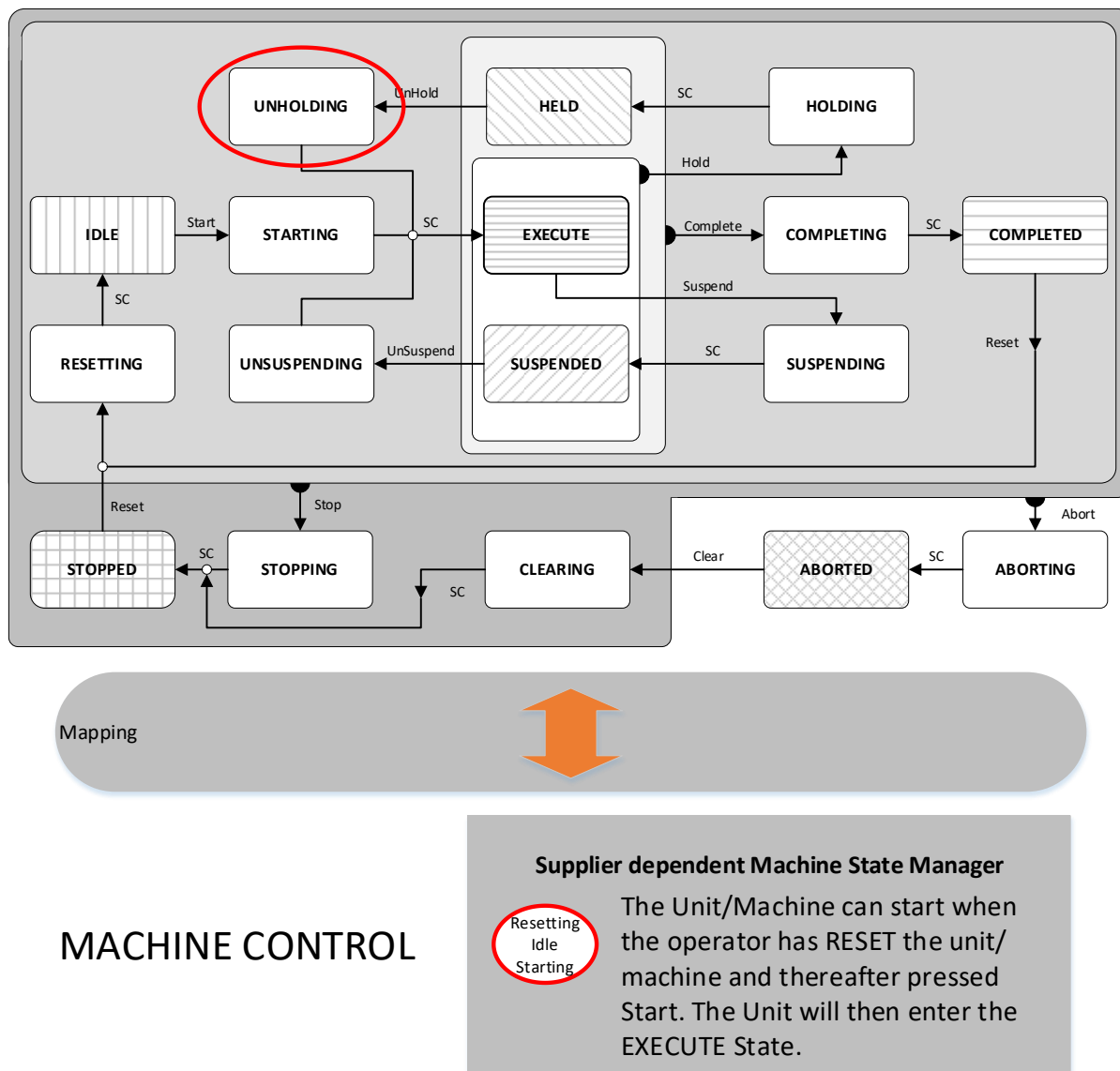
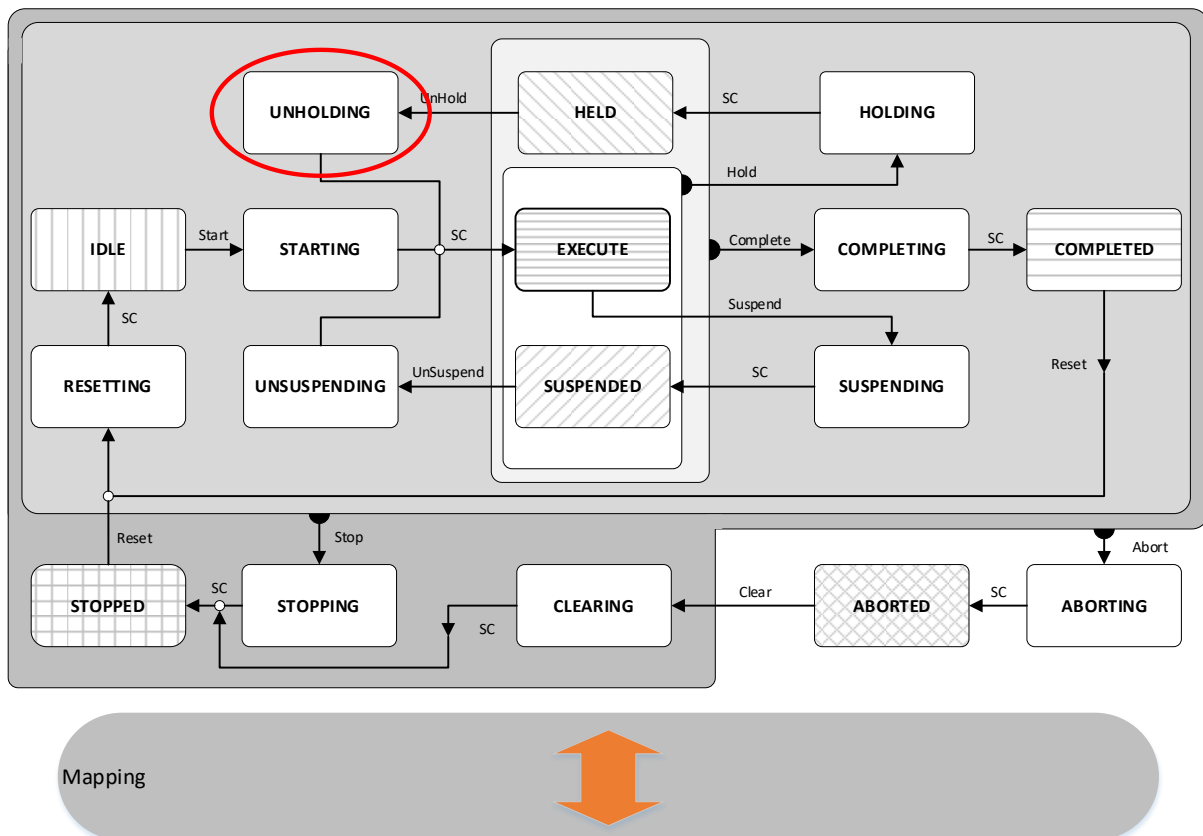


Figure 24 Resetting mapping between the PackML Interface State Model and the Machine control State model

PACKML INTERFACE (UNHOLDING from clear, stopped, ...)

The execution of a Production order/Batch is UNHOLDING because of problem on the unit/machine is fixed. The Machine Controller have been E-stopped and needs to be cleared and restarted.



MACHINE CONTROL

Supplier dependent Machine State Manager

Clear
Stopped
Resetting
Idle
Starting

The Unit/Machine can start when the operator has CLEARED the E-stop, RESET the unit/machine and thereafter pressed Start. The Unit will then enter the EXECUTE State.

Figure 25 Clear mapping between the PackML Interface State Model and the Machine control State model

7.6 SUSPENDING

Below are two machine control scenarios that can bring the PackML Interface State Model into the SUSPENDING state. The unit/machine can be in the SUSPENDING state, because of one of the following reasons:

- 1) Starvation or saturation (Blockage) of material upstream or downstream.
- 2) Operator has stopped the machine control because of an external condition.

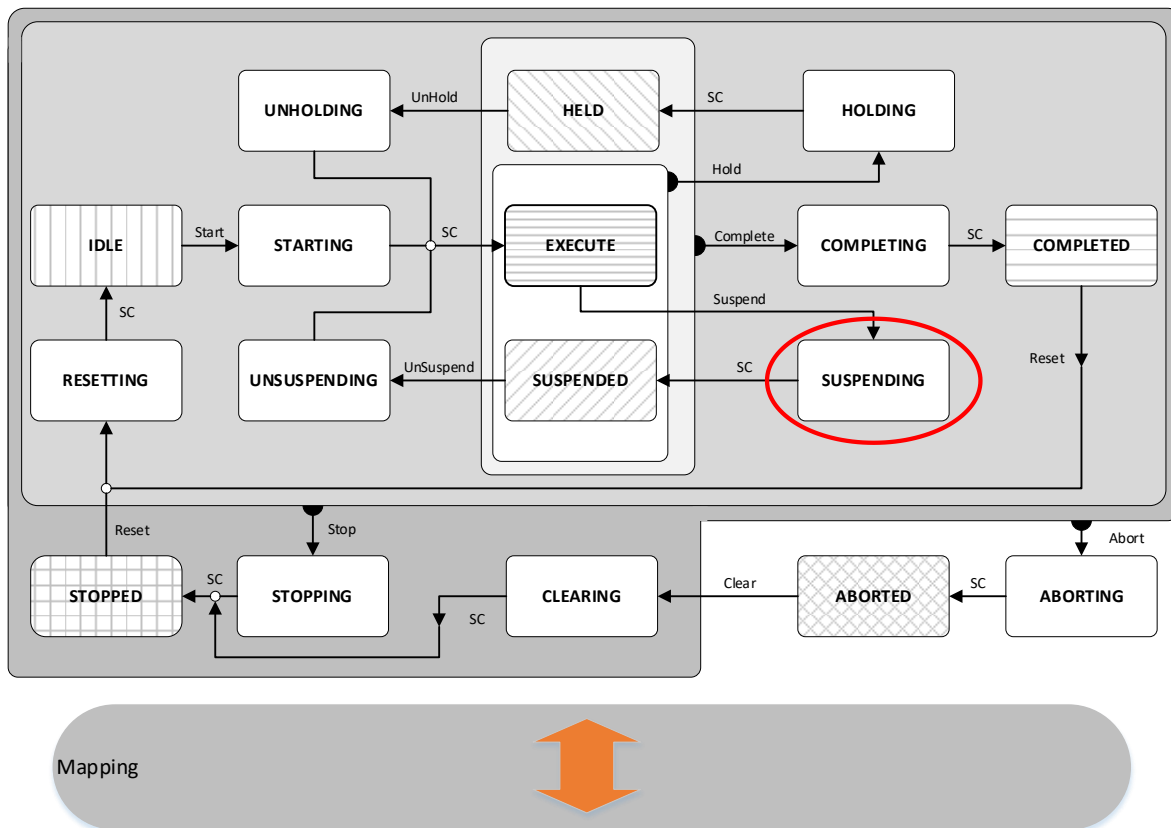
Table 8 Interpretation of the SUSPENDING state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
SUSPENDING	<p>This SUSPENDING state is required prior to the SUSPENDED wait state, and prepares the unit/machine for e.g. stop ironing, stop cutting, stop debagging, etc.</p> <p>The unit/machine may be running at a relevant set point speed, but there is no product being send to the next unit/machine downstream.¹⁷</p> <p>The Machine Supplier and End User must specify the special conditions that have to take place when the units enter the Suspending and SUSPENDED state. Does the unit/machine fill up its internal product buffers or does the unit/machine stop immediately.</p> <p>The actions to be taken when the unit/machine enters the Suspending state are to be defined.</p> <p>Suspending is the result of starvation of material within the in-feeds or a result of saturation related out-feed blockage that prevents the unit/machine from Executing continued steady production. During the controlled sequence of Suspending the unit/machine will transition to a SUSPENDED state.</p>	Suspending	<p>This state shall be used when external process conditions (outside this unit/machine but usually on the same integrated production line) do not allow the machine to continue producing. Usually when the machine leaves EXECUTE due to upstream or downstream conditions on the line. For example, the unit/machine moves from EXECUTE if either upstream equipment cannot supply material or downstream equipment cannot receive material.</p> <p>This condition may be detected by a local machine sensor or based on a supervisory system external command.</p> <p>While in the SUSPENDING state, the machine is typically brought to a controlled stop and then transitions to SUSPENDED upon state complete. To be able to restart production correctly after the SUSPENDED state, all relevant process set-points and return status of the procedures at the time of receiving the <i>SUSPEND</i> command must be saved in the machine controller when executing the SUSPENDING procedure.</p>
SUSPENDING	<p>The SUSPENDING state might be forced by the operator via the external interface, such as when the operator needs to have a break.</p>	Stopping	<p>The machine is powered and stationary after completing the Stopping state. All communication with other systems is functioning, if applicable.</p> <p>As a result of requesting a Controlled Stop to the Stopped state.</p>

¹⁷ What is happening at specific unit/machine have to be defined, because this is unit/machine specific. Some unit/machines may not stop with open products.

PACKML INTERFACE (SUSPENDING from suspending)

The execution of a Production order/Batch is SUSPENDING because of problem on units outside the current unit/machine, that gives a starvation or saturation situation for the current unit/machine.



MACHINE CONTROL

Supplier dependent Machine State Manager

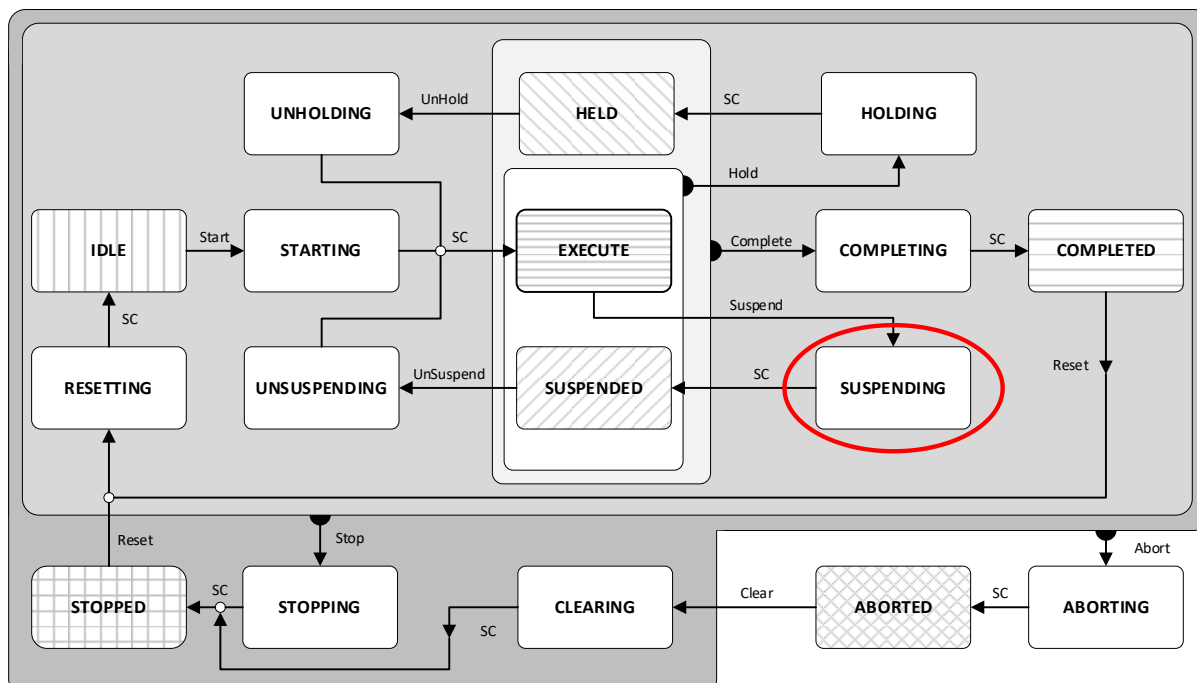
Suspending

The Unit/Machine can stop because of external issues, such as starvation or blockage of products.

Figure 26 Suspending mapping between the PackML Interface State Model and the Machine control State model

PACKML INTERFACE (SUSPENDING from stopping)

The execution of a Production order/Batch is SUSPENDING because the operator have ordered a stop. For example a short break.



Mapping



MACHINE CONTROL

Supplier dependent Machine State Manager

Stopping

The Unit/Machine can stop because of operator actions.

Figure 27 Stopping mapping between the PackML Interface State Model and the Machine control State model

7.7 UNSUSPENDING

The UNSUSPENDING scenarios are related to the way that the unit/machine has entered the SUSPENDING state. The SUSPENDING state has two different scenarios and therefore, there are also two scenarios in the UNSUSPENDING process.

- 1) Automatic change on Machine Control level, because the sensors indicating starvation or saturation (blockage) of material upstream or downstream are no longer active.
- 2) The operator wants to restart the unit/machine.

Table 9 Interpretation of the UNSUSPENDING state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
UNSUSPENDING	<p>This state is a result of a unit/machine generated request from SUSPENDED state to go back to the EXECUTE state or an Unsuspend command via the external interface.</p> <p>This state is done prior to the EXECUTE state and prepares the unit/machine for the EXECUTE state.</p> <p>The actions of this state may include ramping up speeds, turning on vacuums, and the re-engagement of clutches.</p> <p>The unit/machine is not allowed to change state to EXECUTE unless that the external command has been set equal to Unsuspend.</p> <p>The actions to be taken when the unit/machine enters the UNSUSPENDING state are to be defined by the Machine Supplier and End User.</p>	Unsuspending	<p>Refer to SUSPENDING for when this state is used. This state is a result of process conditions returning to normal. The UNSUSPENDING state initiates any required actions or sequences necessary to transition the machine from SUSPENDED back to EXECUTE. To be able to restart production correctly after the SUSPENDED state, all relevant process set points and return status of the procedures at the time of receiving the SUSPEND command must be saved in the machine controller when executing the SUSPENDING procedure.</p> <p>The machine moves from SUSPENDED if either upstream equipment or downstream equipment can receive product again. For example, a signal may come from an external Line Controller or sensors when any line buffering becomes clear and the machine can start producing again.</p>
UNSUSPENDING		Reset Idle Starting	The operator has handled the actions on the Machine Control panel (HMI) necessary to return to the EXECUTE state. The operator has made a controlled stop of the unit/machine, and has restarted.

PACKML INTERFACE (UNSUSPENDING from unsuspending)

The execution of a Production order/Batch is UN-SUSPENDING because of problem on units outside the current unit/machine are fixed.

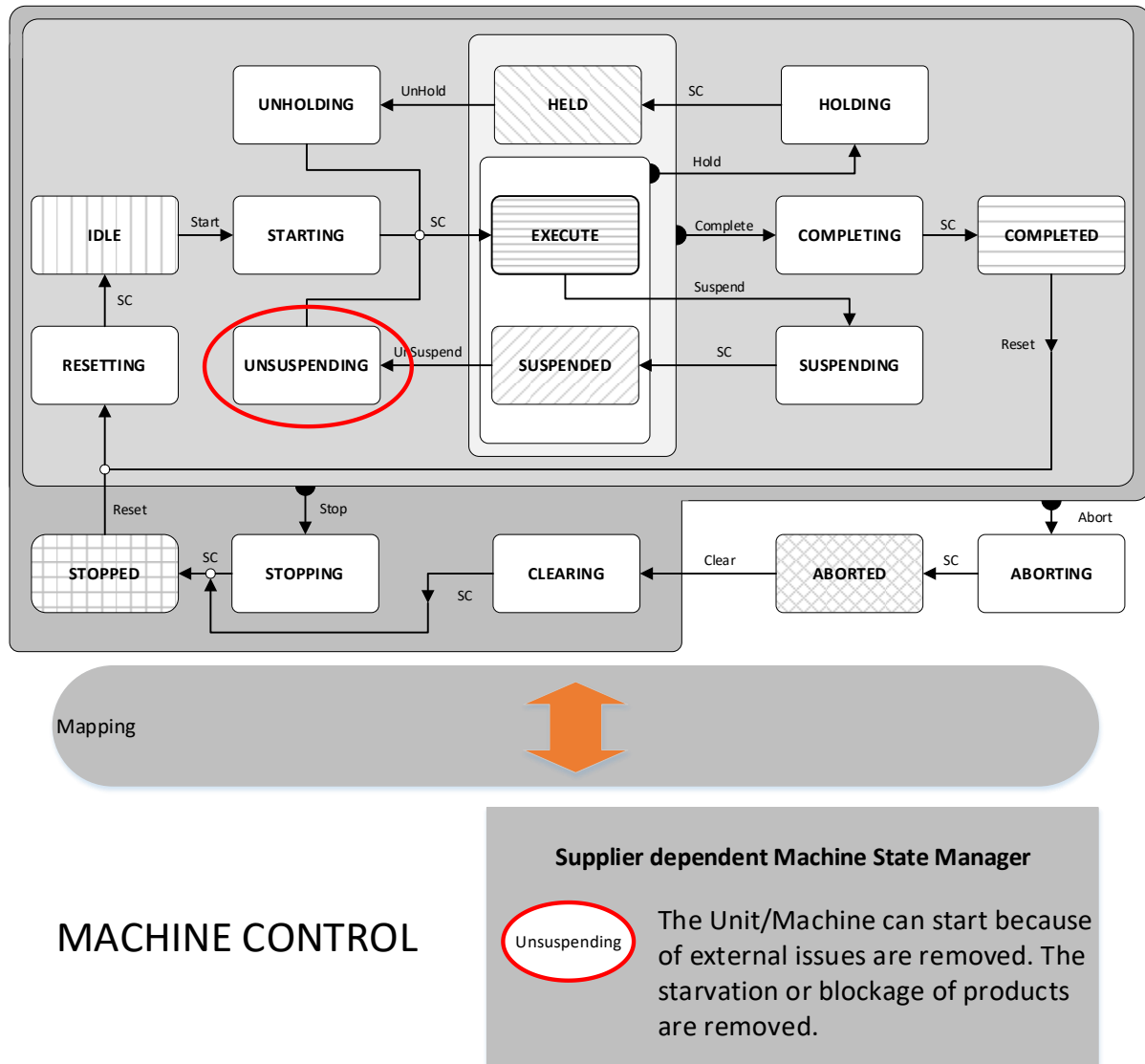


Figure 28 Unsuspend mapping between the PackML Interface State Model and the Machine control State model

PACKML INTERFACE (UNSUSPENDING from resetting & idle)

The execution of a Production order/Batch is UNSUSPENDED because operator want to restart production. The Machine Controller have been stopped and needs to be restarted.

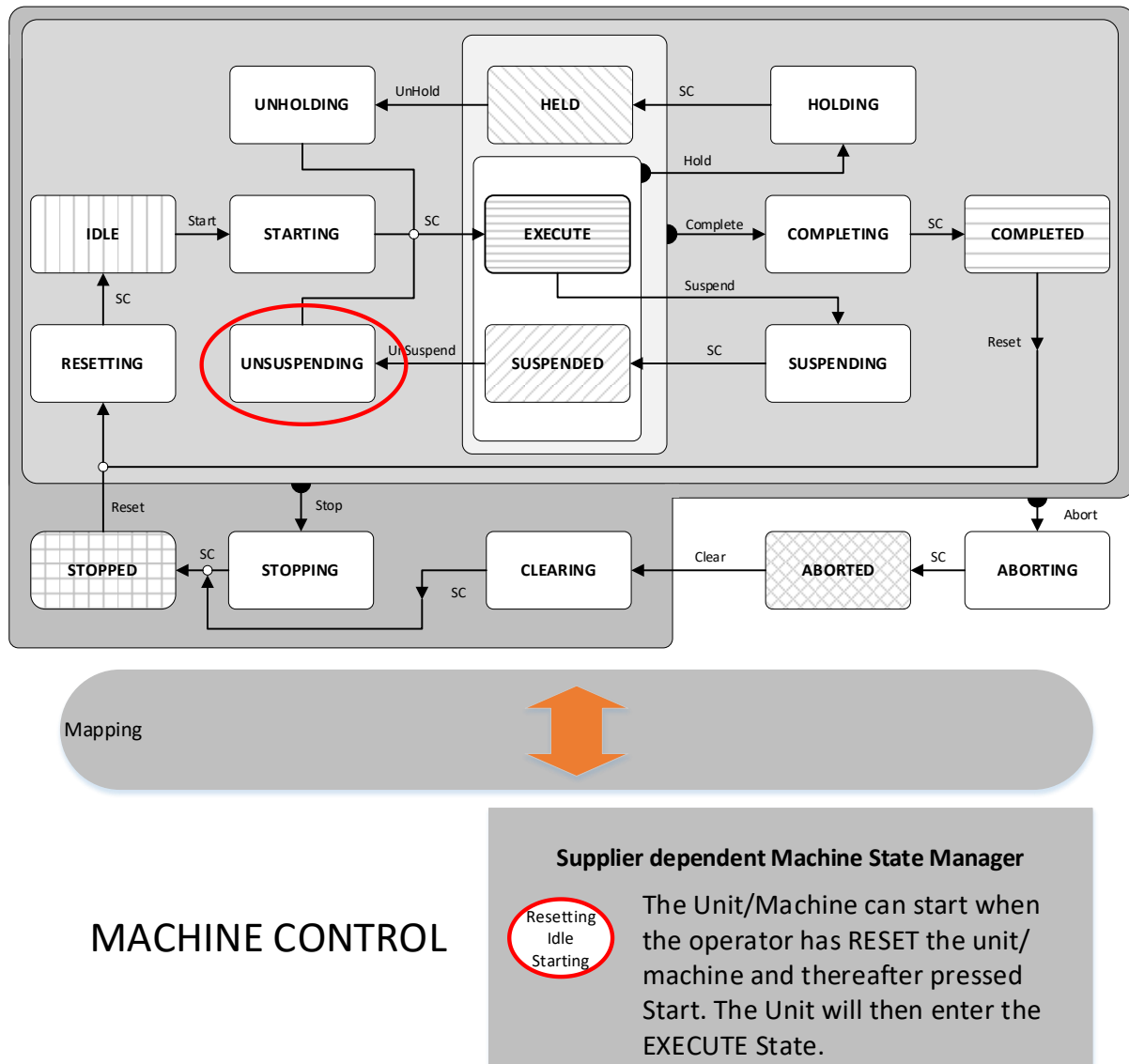


Figure 29 Resetting mapping between the PackML Interface State Model and the Machine control State model

7.8 COMPLETING

The Completing is related to the way in which the unit/machine is able to stop the production of a Production order.

Table 10 Interpretation of the COMPLETING state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
COMPLETING ¹⁸	<p>This state provides the steps needed to complete the job usually as a result of product counter reaching a limit in the unit. The unit/machine will make available machine parameters back to the external system. The machine parameters can be obtained by the external system to get information about manual changed or corrected data on the unit/machine during executing.</p> <p>The Machine Supplier and End User have to specify the special conditions that have to take place when the unit/machine enters the COMPLETING and COMPLETE state. For example, does the unit/machine empty its internal product buffers, or does the unit/machine stop immediately with products within the unit.</p> <p>Actions to be taken during COMPLETING are to be specified by the Machine Supplier and End User.</p>	Completing	<p>This state is an automatic response from the EXECUTE state when the specified number of products have been processed. Normal operations have run to completion, i.e., processing of material at the infeed will stop.</p> <p>For example, this state could also be issued by the operator on the Machine Control by pressing an HMI push button.</p>

¹⁸ In the case that the operator wants to force "Finalize" or "Complete" the production. It is not possible to enter the COMPLETE state from an external system. The operator has to issue a Stop command to the unit/machine. This gives the operator the flexibility to stop the production at a certain amount of products through an external system.

PACKML INTERFACE (COMPLETING from completing)

The execution of a Production Order/Batch is COMPLETING because the number of products to produce are reached.

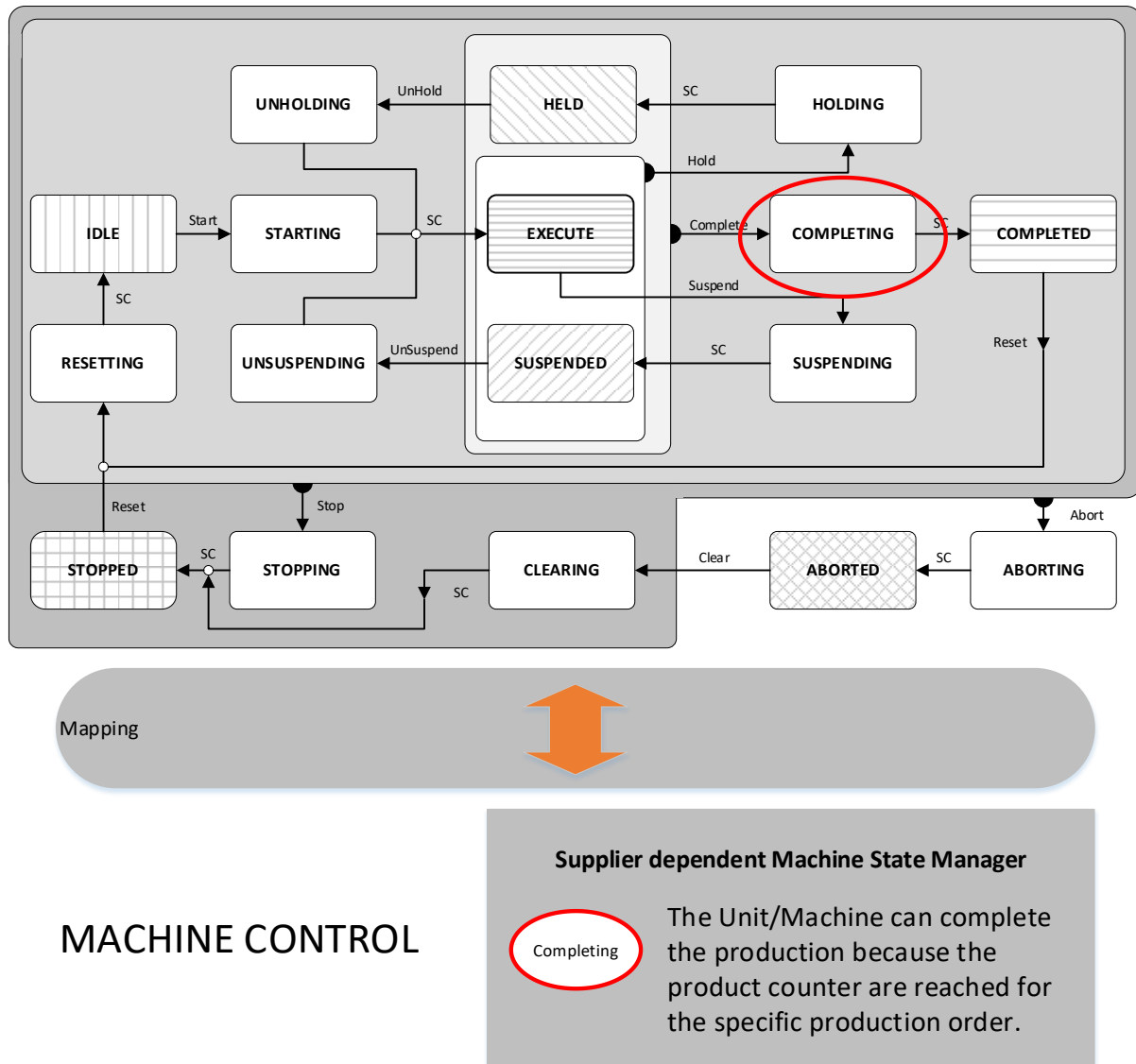


Figure 30 Completing mapping between the PackML Interface State Model and the Machine control State model

7.9 STOPPING

The purpose of the STOPPING state is to stop the production of a production order from a Supervisory control system or when the operator wants to stop the production order at a unit/machine.

Table 11 Interpretation of the STOPPING state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
STOPPING	<p>This state executes control logic which brings the unit/machine to a controlled stop as reflected by the STOPPED state.</p> <p>The unit/machine can make available machine parameters back via the External interface. The machine parameters can be used to get information about manual changes and corrections on the unit/machine during executing.</p> <p>The Machine Supplier and End User must specify the special conditions that have to take place when the unit enters the STOPPING and STOPPED state. For example, does the unit/machine empty its internal product buffers, or does the unit/machine stop immediately.</p>	Stopping	<p>This state is entered in response to a <i>STOP</i> command. In this state the machine executes the logic which brings it to a controlled stop as reflected by the STOPPED state.</p> <p>Normal STARTING of the machine cannot be initiated unless RESETING had taken place.</p> <p>As a result of completing a controlled stop the unit/machine will transition to the STOPPED state.</p>

PACKML INTERFACE (STOPPING from stopping)

The execution of a Production order/Batch is STOPPING because the operator has stopped the production order/Batch.

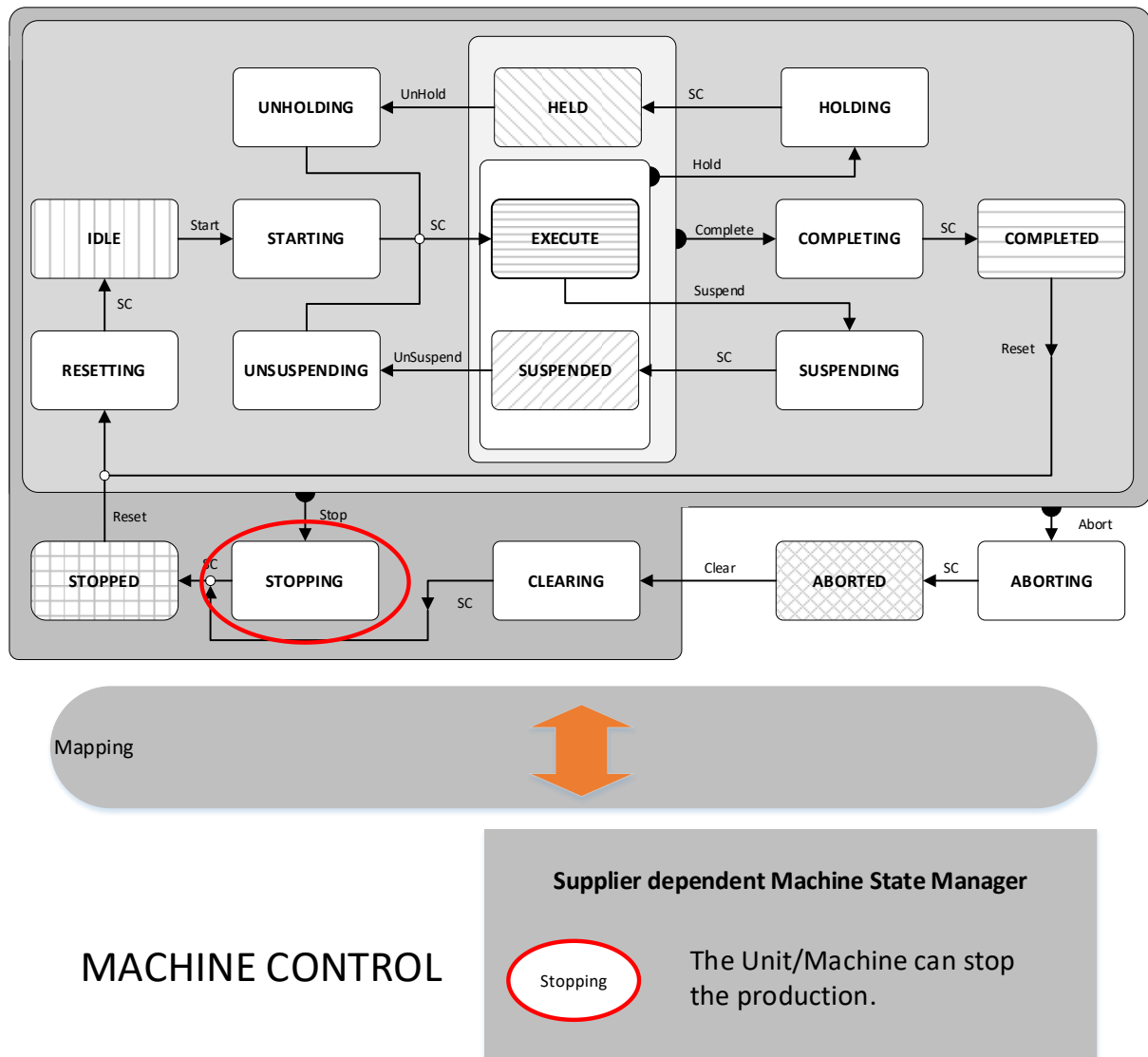


Figure 31 Stopping mapping between the PackML Interface State Model and the Machine control State model

7.10 ABORTING

Table 12 Interpretation of the ABORTING state

PackML Interface State	PackML Interface description (PackML Interface State Manager)	Machine State	Machine State description (Supplier dependent Machine State Manager)
ABORTING	The ABORTING state can be entered at any time as a response to an Abort command. The aborting logic will bring the unit/machine to a rapid safe stop. The Aborting logic will provide the necessary actions to move the unit/machine into the ABORTED State. The actions to be taken when the unit/machine enters the Aborting state are to be defined by the Machine Supplier and End User.	Aborting ¹⁹	The ABORTING state can be entered at any time in response to the <i>ABORT</i> command or on the occurrence of a machine fault. The aborting logic will bring the machine to a rapid safe stop. Aborting is a result of e-stop button pushed or a machine fault that needs a fast stop of the unit/machine.
		Stopping	This state is entered in response to a <i>STOP</i> command. While in this state the machine executes the logic which brings it to a controlled stop as reflected by the unit/machine stopped state. Normal starting of the machine cannot be initiated unless PackML RESETING had taken place.

¹⁹ The products produced before the Aborting, are in the packaging world, normally not destroyed. The aborted state indicate that the production order is finished and cannot be started again.

PACKML INTERFACE (ABORTING from stopping)

The execution of a Production order is **ABORTED** because the operator has terminated the production order. There are two different ways to map to the abort state:

- 1) The Unit/Machine maps a stopped state to the PackML Aborted State or
- 2) The Unit/Machine maps the aborted state to the PackML Aborted State.

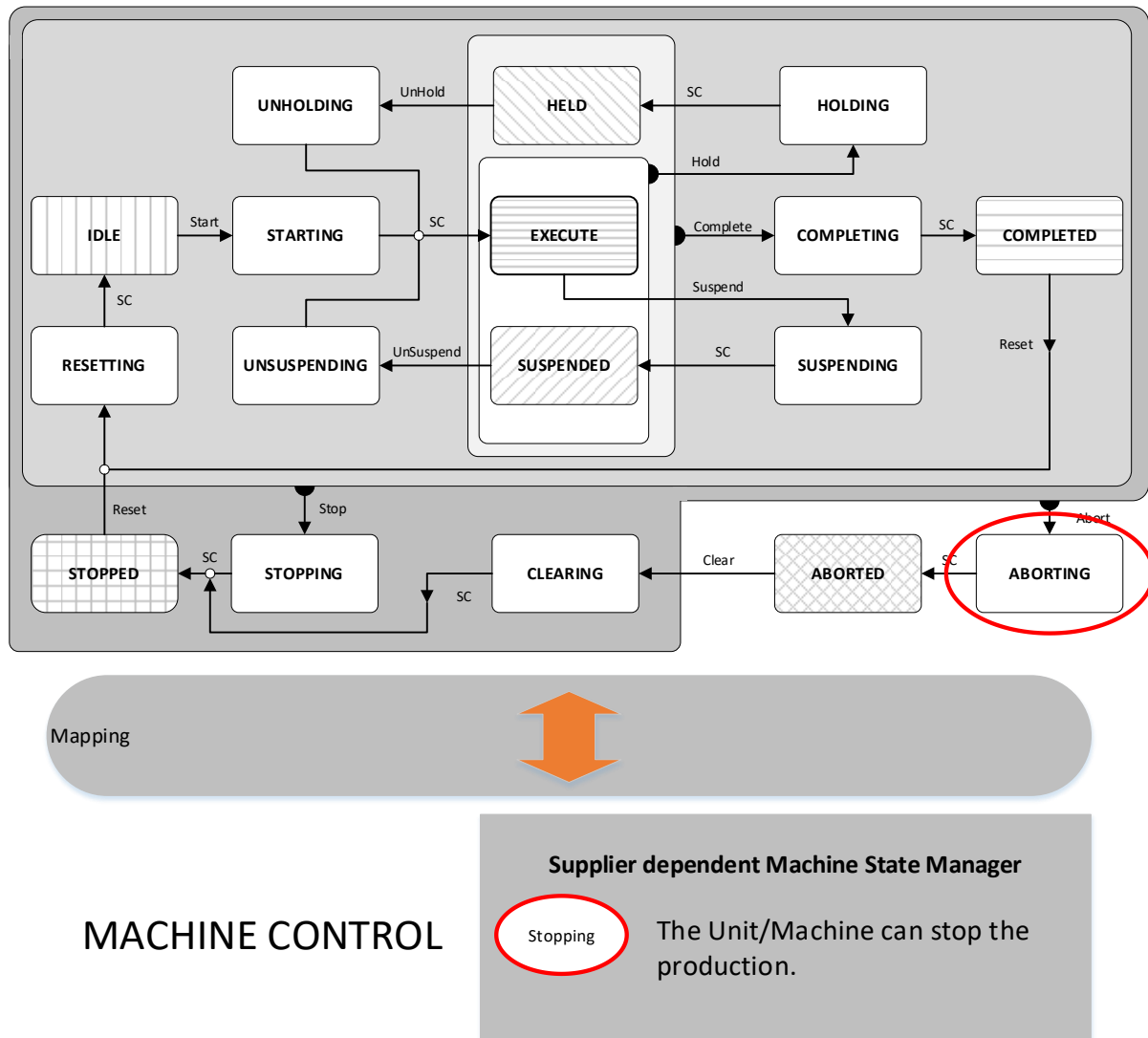


Figure 32 Stopping mapping between the PackML Interface State Model and the Machine control State model

PACKML INTERFACE (ABORTING from aborting)

The execution of a Production order is ABORTED because the operator has terminated the production order. There are two different ways to map to the abort state:

- 1) The Unit/Machine maps a stopped state to the PackML Aborted State or
- 2) The Unit/Machine maps the aborted state to the PackML Aborted State.

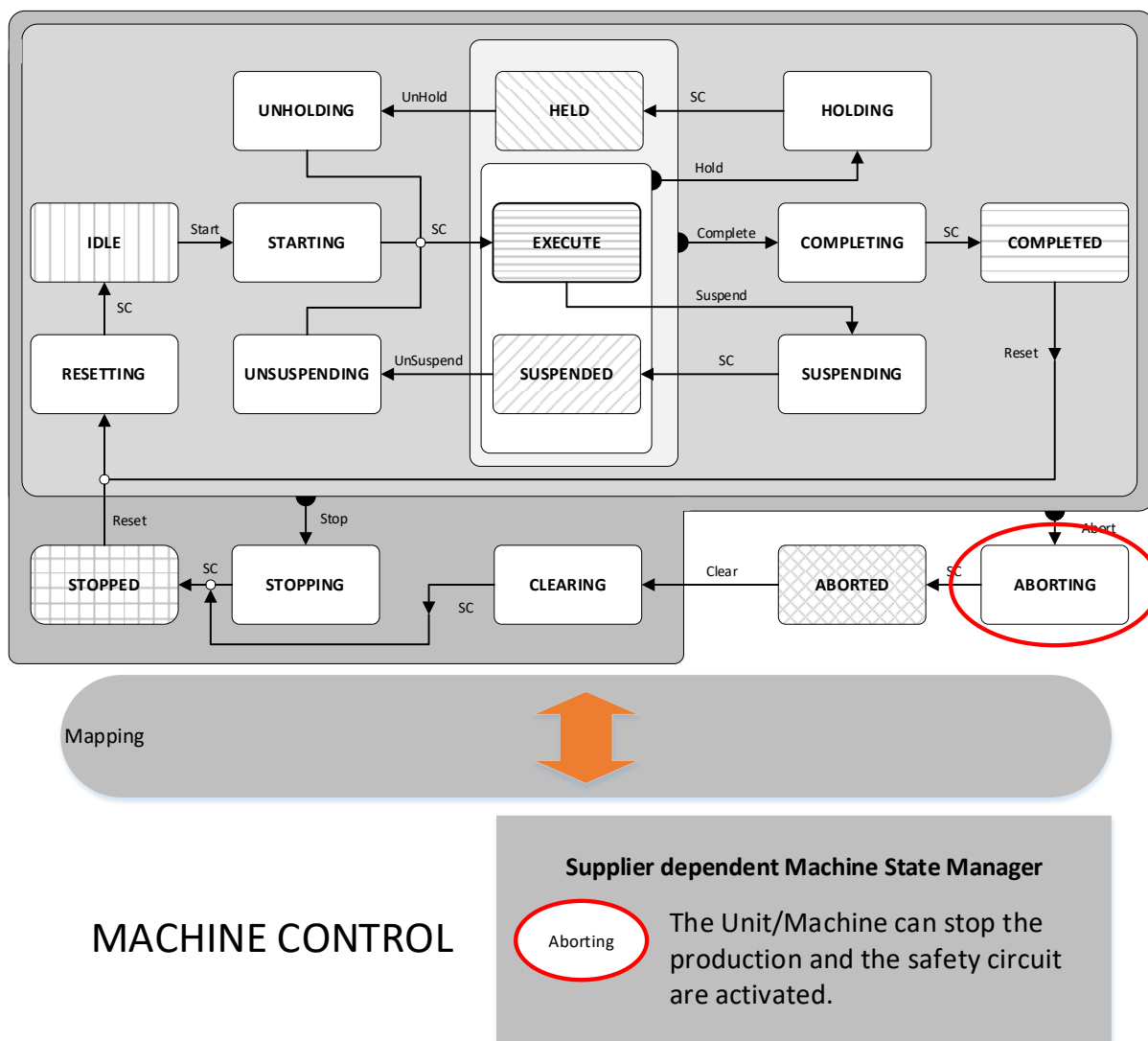


Figure 33 Aborting mapping between the PackML Interface State Model and the Machine control State model

7.11 MAPPING TABLE

As illustrated below, Table 13 gives an overview of the mapping possibilities between the PackML Interface States and a set of example Supplier dependent Machine States. The example Supplier dependent Machine State model is depicted in Figure 34.

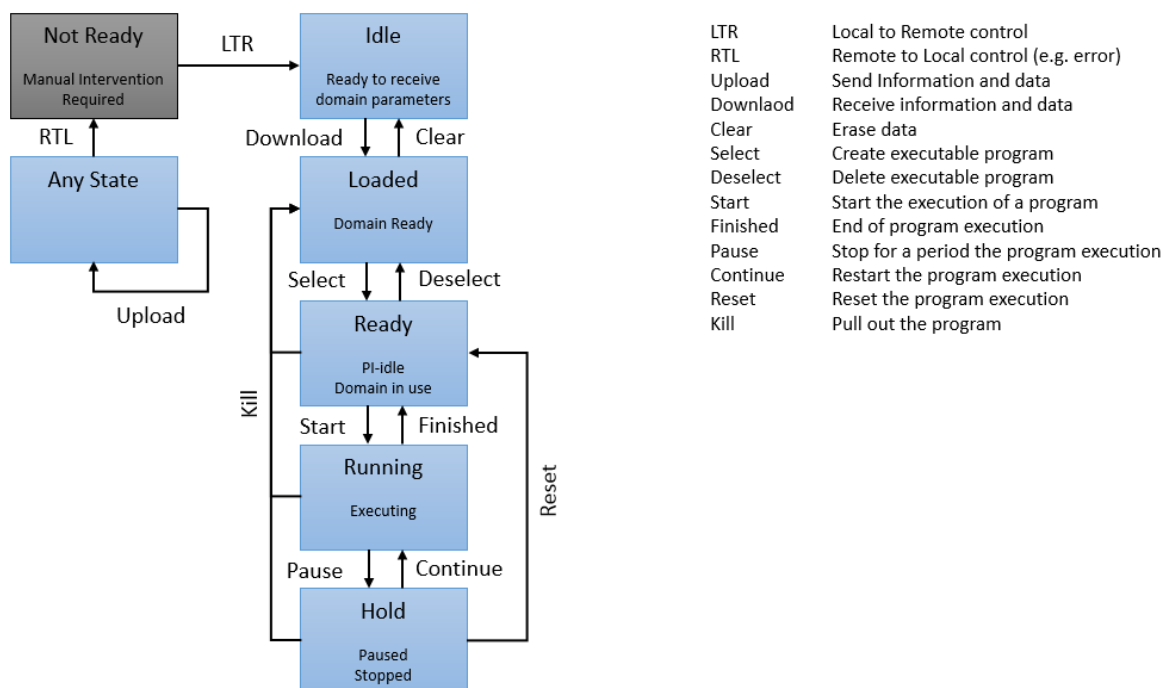


Figure 34 Example supplier dependent Machine State model

The mapping process determines which Supplier dependent Machine states and commands correspond to PackML States and commands. The state model above shows several states and a lot of state transactions that can be mapped against the PackML acting states (White coloured boxes on the PackML state model diagram), as shown in Table 13.

Table 13 An example mapping table between PackML Interface States and Machine States

		Supplier dependent Machine State & transitions																		
		Not ready (Manual intervention required)	LTR – Local to Remote control	RTL – Remote to local control (E.g. error)	Idle	Download (Not supported by PackML)	Clear	Loaded	Select (Not supported by PackML)	Deselect (Not supported by PackML)	Ready	Start	Finished	Running	Pause	Continue	Hold	Reset	Kill	Upload (Not supported by PackML)
PackML Interface State	STOPPED	X																		
	Resetting		C				C													
	IDLE				X			X												
	Starting									X	C									
	EXECUTE												X							
	Holding			C																
	HELD	X																		
	Unholding		C																	
	Suspending														C					
	SUSPENDED																X			
	Unsuspending															C				
	Completing												C							
	COMPLETE													X						
	Aborting			C															C	
	ABORTED	X																		
	Clearing		C																	
Stopping			C														C			

X = State, C = Command (Acting state)

Hint: The Machine Supplier and End User have to agree on when the machine is to be in the different PackML interfaces states. It is recommended to make a matrix such as illustrated in Table 13 with the PackML Interface States as one dimension of the table and the current machine dependent machine states as the second dimension of the table. Mark all the possible situations for the different PackML Interface states.

Hint: During the specification of a specific unit/machine, use a version of Table 13, and fill in the specific Supplier dependent Machine State & transitions.

7.12 MAPPING ALARMS TO EVENT TRIGGERS TO PACKML

It is possible to create a matrix that allows a Machine Supplier to configure which types of alarms and events that will lead to a PackML state change. It can be used to specify when a machine should switch to the states in the PackML state model. See Table 14 as an example mapping table.

It is especially useful to document when the unit/machine has to move to HELD, STOPPED, SUSPENDED and ABORTED states, because the End Users sometimes have different interpretations and expectations of when the different states should be entered.

Table 14: Error and event triggers relation to PackML state management table

Events / Alarms	State change ID = Commands	Reaction for Machine Control	Reaction for PackML Interface state manager
1	8	Abort	ABORTING
2	8	Abort	ABORTING
3	4	Hold – Error on material	HOLDING
4	4	Hold – Error on glue	HOLDING
5	3	Stop – Drive failure	ABORTING
6	3	Stop – Sensor failure	HOLDING
7	6	Suspend - Blocked	SUSPENDING
8	6	Suspend – Queue	SUSPENDING
9	20	Pop-up information on HMI – Executing state	No state change
10	20	Pop-up information on HMI – Executing state	No state change
.....

Hint: During the specification of a specific unit/machine, is it recommended to use an empty version of Table 14, and to fill in the specific Supplier dependent Machine alarms and warnings and the PackML Interface State Manager action.

8. PACKML CONTROL COMMAND DEFINITIONS


The Control Command provides the state commands to drive state changes in the PackML Interface State Model (see Figure 10: The PackML Interface State Model). The processing of the command in the unit/machine Control System may be combined with remote or local unit/machine conditions to drive the interface state model from a Wait state to an Acting state. The unit/machine shall always respond to a command²⁰, but not all commands will cause a PackML state change.

8.1 PACKML COMMANDS

The Control Command can be set by the unit/machine HMI or remote set via the PackML interface. The column “Issued” in Table 15, describes who is able to issue the command.


The Control Commands shall be exposed as an Integer, the specific command values are given in 8.2.




Table 15: Control internal, external and e-stop commands

Command	Issued			Comments
	On or by unit/machine ²¹	By External interface	E-stop	
Start	YES (*) 	YES		The start command will move the unit/machine into the Starting state. The start command will finally put the unit/machine into the EXECUTE state. If an error occurs the unit/machine will be put into STOPPED or ABORTED. An external system is allowed to give the Start command when the unit/machine is in Production Mode and IDLE State. Starting the unit/machine from an external system, when the parameters were downloaded in STOPPED or IDLE state, will automatic start the unit, because the unit/machine is already locally approved for production by operator (See reset command). (*) Start on the unit/machine itself can be used if there is no external system.
Hold	YES			The Hold command can be executed by the unit/machine when an internal unit/machine fault is detected or by an operator command on the unit/machine during Production. The Hold command changes the unit/machine from the EXECUTE state to the HOLDING state. The Hold command offers the operator a safe way to intervene manually in the process, for example removing

²⁰ If a command is not valid in relation to the current state, the unit/machine should not react, but should give a warning. For example a start command is send to a unit in state STOPPED, and then the unit/machine has to respond by rejecting the command.

²¹ The column “Issued On or by unit/ machine” indicates that the operator can, from the HMI or via a hardware push button at the machine, initiate the command. The column also indicates that the unit itself can initiate the command. For example, an Unsuspend command does not have to come from the operator.

Command	Issued			Comments
	On or by unit/ machine ²¹	By External interface	E-stop	
				broken material from the in-feed, or opening a safe door and removing cardboard. The configuration of safe stops are not defined within this guideline.
Unhold	YES 			<p>The Unhold command releases the HOLD state and moves the unit/machine to the UNHOLDING state and then back to EXECUTE state.</p> <p>The Unhold command is a response to an Operator command to resume the production after an error/fault on the unit indicated by an alarm or warning. For example, when the operator has cleared material blockage.</p> <p>An Unhold command is always required from the operator and can never be initiated automatically by the unit.</p>
Suspend	YES	YES		<p>The Suspend command from the unit/machine is the result of starvation of material within the in-feeds, or a result of saturation related out-feed blockage that prevents the unit/machine from Executing continued steady production, or on operator command.</p> <p>A unit/machine that is already in SUSPENDED state and receives an external Suspend command, will only change the stop reason.</p> <p>A Suspend command may come as the result of upstream or downstream problems outside the unit/machine itself.</p>
Unsuspend	YES	YES		The Unsuspend command releases the unit/machine from SUSPENDED state and moves the unit/machine to the UNSUSPENDING state and finally to the EXECUTE state. The Unsuspend command is initiated by the unit when the Downstream and Upstream units are ready to execute, or when the an external system issues an Unsuspend.
State Completed (SC) from any acting states	YES			The State Completed (SC) is an internal unit/machine action/command, which moves the unit/machine itself to the next state. The SC is activated when the appropriate unit/machine conditions are reached.
Complete	YES	YES		The complete command may be internally generated based on unit/machine conditions, such as reaching the end of a pre-defined production count, or externally generated. This means the production order is complete. The Complete command moves the unit/machine from the EXECUTE, HELD or SUSPENDED state to the COMPLETING state and finally reach the COMPLETE state.

Command	Issued			Comments
	On or by unit/ machine ²¹	By External interface	E-stop	
				<i>HINT: The unit/machine can be forced via the external interface with a Stop command to complete the order, and if required empty its output buffer for products.</i>
Stop	YES 	YES		<p>The Stop command will start the execution of logic which brings the unit/machine to the STOPPING state and finally to the STOPPED state.</p> <p>The operator can stop the unit/machine via the external interface or on the unit/machine HMI. The unit/machine will stop and make available parameter data via the PackML interface.</p> <p>The Stop command can be initiated from any state except ABORTED, ABORTING, CLAERING, STOPPING and STOPPED. This means, that the Stop command can be initiated within the grey area in the PackML interface state model (see Figure 10: The PackML Interface State Model).</p> <p>To finish production, an external Stop command can be sent to the unit, when the unit/machine is in EXECUTE, HELD, SUSPENDED, COMPLETE, or IDLE state and the unit/machine is in Production Mode.</p>
Reset ²²	YES 			<p>The Reset command is activated by the operator on the unit/machine HMI.</p> <p>The operator has to initiate on the unit/machine that is ready for production and move the unit/machine into the IDLE state via the RESETING state.</p> <p>The operator has to check that all safety issues are satisfied, and the unit/machine conditions are ready for production.</p> <p>The unit/machine can either be in the STOPPED or COMPLETE state when the Reset command is processed.</p>
Abort			YES 	<p>The Abort command is to be handled automatically when the E-stop is activated on the Unit.</p> <p>The abort command can be executed at any state (see Figure 10 – the light grey and dark grey), and will change the unit/machine to the Aborting state and to the transitional state ABORTING and the secure state ABORTED.</p> <p>Operation of the emergency stop will cause the unit/machine to be tripped by its safety system and send an Abort command. It is to be determined, which technical solutions</p>

²² The Reset command is required to ensure that the unit/machine is safe to start from remote.

Command	Issued			Comments
	On or by unit/ machine ²¹	By External interface	E-stop	
				that ensures the unit/machine to move and remain in ABORTED state.
Clear	YES			The clear command is used when the unit/machine has been stopped by an Abort command. The Clear command is activated on the unit/machine HMI to clear faults that may have occurred when ABORTING and were presented in the ABORTED state before proceeding to a CLEARING state and then the STOPPED state.

8.2 SPECIFICATION OF COMMANDS ON UNIT HMI AND FROM EXTERNAL SYSTEM

The Machine Supplier should fill out the table below to identify what commands are possible to activate from unit HMI or External system via interface.

Table 16 Defined Commands

Command	PackML Tag Value	Implemented on Unit			
		External interface	Unit push buttons and/or HMI action	E-stop	Name on unit HMI
No command	0	Optional	N/A		N/A
Reset	1	Optional	YES		Reset
Start	2	Optional	YES		Start
Stop	3	Optional	YES		Stop
Hold	4	Optional	YES		N/A
Unhold	5	Optional	YES		Restart
Suspend	6	YES	YES		N/A
Unsuspend	7	YES	YES		N/A
Abort	8	Optional		YES	E-Stop
Clear	9	Optional	YES		Reset E-Stop
Complete	10	Optional	YES		End or Stop Order

***Hint:** During the specification of a specific unit/machine, is it recommend use an empty version of Table 16, and fill in the specific Supplier dependent Machine commands and equivalent HMI names.*



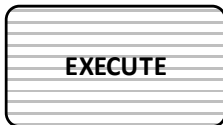
9. PACKML INTERFACE STATE DEFINITION






The Interface unit/machine State defines the current condition of a Unit. An Interface unit/machine State is expressed as the overall state for the unit/machine as seen through the HMI or external control system. The unit/machine could contain several internal controllers (PLC's or Drive controllers) but from the PackML point of view, the unit/machine will be seen as a single Unit.

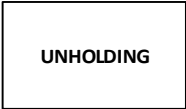



The State is to be exposed as an Integer value as shown in Table 17.

When a unit/machine is powered on it will start in the STOPPED state.

Table 17: Interface unit/machine state definitions

State name	PackML Tag Value	Description
<p>IDLE</p> 	1	<p>This is a state which indicates that Resetting is complete. The unit/machine is ready for new production and machine parameters can be received. If an error occurs on the unit, the unit/machine will enter the STOPPING state and end in STOPPED state.</p> <p>In the IDLE state the unit/machine is approved for safety issues and ready for production. The unit/machine can allow, in the IDLE state, a remote Start command from external system. It is to end User responsibility to ensure that company safety is followed for external Start commands.</p>
<p>STARTING</p> 	3	<p>This state provides the steps needed to start the unit/machine and is a result of a Start command. The starting logic will load the unit/machine parameters into the Machine Control and move the unit/machine into the EXECUTE state if parameters were loaded correctly. The starting logic will ramp up speed, pressure, etc. to be ready for production. Otherwise, if there were errors in the loaded data, the starting logic will move the unit/machine into the STOPPING state and the unit/machine will end in STOPPED state. The unit/machine is not allowed to change state to EXECUTE unless the Start command is received.</p> <p>If an error occurs on the unit, the unit/machine will enter the STOPPING state and end in the STOPPED state.</p> <p>Actions to be taken during the STARTING state are to be specified by the Machine Supplier and End User.</p>
<p>EXECUTE</p> 	6	<p>Once the unit/machine is producing materials it is in the EXECUTE state.</p> <p>The machine / unit/machine is running with all conditions met, as defined by the selected MODE and / or recipe selected.</p>

State name	PackML Tag Value	Description
<p>COMPLETING</p> 	16	<p>This state provides the steps needed to complete the job, such as the result of a product counter value has been reached in the unit. Corrected parameters (Recipe Control Parameters) on the unit/machine by the operator can be made available to the external system. The machine parameters can be used by the external system to get information about manual changed or corrected data on the unit/machine during execution.</p> <p>The Machine Supplier and End User have to specify the special conditions that have to take place, when the unit/machine enters the COMPLETING and COMPLETE state. For example, does the unit/machine empty its internal product buffers, or does the unit/machine stop immediately with products within the unit.</p> <p>Actions to be taken during the COMPLETING state will be specified by the Machine Supplier and End User.</p>
<p>COMPLETE</p> 	17	<p>The unit/machine has finished the COMPLETING state and is now waiting for a Reset command before transitioning to the Resetting state. The unit/machine is powered and all data is still represented within the unit. The unit/machine is ready to be reset.</p>
<p>Resetting</p> 	15	<p>The RESETTING state will start the resetting process, which typically cause the unit/machine to clear data and to place the unit/machine in an IDLE state, where the unit/machine components are energized waiting for a Start command. By resetting the Unit, the data (recipe & parameters) is cleared and the unit/machine is ready for new machine parameters. For example, the operator may have changed tooling and by activating the Resetting process the operator has indicated that the machine is ready for production.</p> <p>Actions to be taken during Resetting will be specified by the Machine Supplier and End User.</p>
<p>HOLDING</p> 	10	<p>The HOLDING logic brings the unit/machine to a controlled stop or to a state which represents HELD for the particular Unit.</p> <p>This is internal control logic that is executed when an error/fault occurs on the unit/machine or an operator initiates a Hold command from unit/machine HMI. The Holding control logics changes the unit/machine from the HOLDING state to the HELD state.</p> <p>The E-stop is an exception, which are handled separately and not part of the Holding logic. See state Aborting and ABORTED.</p>
<p>HELD</p> 	11	<p>The HELD state holds the Unit's operation while material blockages are cleared or enables the safe correction of an equipment fault before the production may be resumed.</p> <p>To be able to restart production correctly after the HELD state, all relevant process set-points and return status of the procedures at</p>

State name	PackML Tag Value	Description
		the time of receiving the Hold Command must be saved in the unit/machine controller when executing the Holding procedure.
UNHOLDING 	12	<p>The UNHOLDING state is a response to an operator command to resume production. The control logic in the UNHOLDING state prepares the unit/machine to re-enter the normal EXECUTE state. The Unholding logic will ramp up speed, pressure, etc. to be ready for production.</p>
SUSPENDING 	13	<p>This SUSPENDING state is required prior to the SUSPENDED wait state, and prepares the unit/machine to suspend, for example it may stop ironing, stop cutter, stop debagger, etc. The unit/machine may be running at a relevant set point speed, but there is no product being produced for the next unit/machine downstream. During the controlled sequence of Suspending the unit/machine will transition to a SUSPENDED state.</p> <p>The Machine Supplier and End User must specify the special conditions that have to take place when the units enter the SUSPENDING and SUSPENDED state. For example, does the unit/machine fill up its internal product buffers, or does the unit/machine stop immediately.</p> <p>Suspending is the result of starvation of material within the in-feeds or a result of saturation related out-feed blockage that prevents the unit/machine from Executing continued steady production. The SUSPENDING state might be forced by the operator via the external interface, when the operator needs to have a break or suspends due to external events.</p>
SUSPENDED 	5	<p>The SUSPENDED state can be reached as a result of external process conditions.</p> <p>In the SUSPENDED state the unit/machine is waiting for external process conditions to return to normal and then transition to UNSUSPENDING State and then the normal EXECUTE state.</p>
UNSUSPENDING 	14	<p>This state is a result of a unit/machine generated request from the SUSPENDED state to go back to the EXECUTE state or an Unsuspend command via the external interface.</p> <p>This state is done prior to EXECUTE state, and prepares the unit/machine for the EXECUTE state.</p> <p>The actions of this state may include ramping up speeds, turning on vacuums, and the re-engagement of clutches.</p> <p>The actions to be taken when the unit/machine enters the UNSUSPENDING state are to be defined by the Machine Supplier and End User.</p>
Stopping	7	This state executes control logic which brings the unit/machine to a controlled stop as reflected by the STOPPED state.

State name	PackML Tag Value	Description
<div>STOPPING</div>		The Machine Supplier and End User must specify the special conditions that have to take place when the unit/machine enter the Stopping and STOPPED state. For example, does the unit/machine empty its internal product buffers, or does the unit/machine stop immediately.
STOPPED <div>STOPPED</div>	2	<p>The unit/machine is powered and stationary after completing the STOPPING state. All communication with other systems is functioning. The unit/machine is powered and all parameters is still represented within the unit. The unit/machine is ready to be reset.</p> <p>Normal Starting of the unit/machine cannot be initiated unless Resetting had taken place.</p>
Aborting <div>ABORTING</div>	8	<p>The ABORTING state can be entered at any time as a response to an Abort command. The Aborting logic will bring the unit/machine to a rapid safe stop. The Aborting logic will provide the necessary actions to move the unit/machine into the ABORTED State.</p> <p>The actions to be taken when the unit/machine enters the ABORTING state are to be defined by the Machine Supplier and End User.</p>
ABORTED <div>ABORTED</div>	9	<p>The unit/machine in this state has to maintain unit/machine status information.</p> <p>The unit/machine can only exit the ABORTED state after an explicit Clear command, subsequently to manual intervention to correct and reset the detected safety circuit, for example an E-stop.</p>
Clearing <div>CLEARING</div>	1	<p>Within the CLEARING state the operator clears the safety system and moves the unit/machine into the STOPPED state. The unit/machine clears faults that may have occurred when Aborting and are presented in the ABORTED state before proceeding to a STOPPED state.</p> <p>The unit/machine will make available machine parameters to the external system. The machine parameters are available in order to get information about eventual manual changes and corrections of data on the Unit.</p> <p>The actions to be taken when the unit/machine enters the CLEARING state are to be defined by Machine Supplier and End User.</p>

10. UNIT CONTROL MODES

A unit/machine can be in different modes, for example Production, Maintenance, Manual, Clean, Jog Mode, etc.

A Unit control mode is an ordered subset of states and commands that determines the strategy to be carried out by the unit/machine process.

This means that the PackML interface State Model above may be represented differently in different modes. The PackML Interface State Model must be implemented fully for Production mode, and can be partly implemented for manual and maintenance mode. In the Manual mode all communication with external systems that can manipulate the unit/machine are disabled. Within Manual mode the operator can take over control of the unit via the unit HMI without a connection to an external system.

The Machine Supplier must specify which modes besides Production, Manual and Maintenance are available.

The Unit Control modes shall be exposed as an Integer, as defined in Table 18.

It is up to the Machine Supplier and End User to define the starting mode after power on, for example STOPPED or IDLE.

Table 18 Mandatory and optional Unit/machine modes

PackML Tag Value	Mode	PackML State Model	Restricted Access	Use
1	PRODUCTION	Mandatory	NO	Primary Mode used for all production activities, All PackML model states will be utilized. PackTags are active and reporting. The unit/machine executes relevant logic in response to commands which are mainly coming from External Systems, or entered directly by the operator.
2	MAINTENANCE	Optional	YES	Used for routine preventive Maintenance or planned maintenance. Shall be utilized to document planned maintenance occurrence and duration. This mode allows authorized personnel to run the unit/machine independent of other systems. This mode would typically be used for fault finding, machine trials, or testing operational improvements. For example: The cleaning of a print head is maintenance.
3	MANUAL	Optional	YES	Used for fault diagnosis of unplanned technical intervention. <i>HINT: In Manual Control mode, it is not necessary to implement all PackML states.</i>

Examples of user defined PackML Modes Further Unit Control Modes might be available on the unit/machine, but the unit need not follow the PackML Interface State Model in these other Unit Control Modes .				
PackML Tag Value	Mode	PackML State Model	Restricted Access	Use
4	CHANGE OVER	Optional	NO	Can be used specifically for format or recipe change over, includes “cleaning” operations.
5	CLEAN	Optional	NO	Can be used for Routine Cleaning requirements example: as specified by factory maintenance procedures.
6	SET UP	Optional	Optional restriction	Can be used for set up or adjustments example: mechanical adjustments and testing
7	EMPTY OUT ²³	Optional	NO	Would be used to empty out machine example: end of a block of shifts prior to factory weekend shut down, empty out resident product within the machine that could be sent to finished product and minimize part finished “rework”.
To be defined	OEM specific	Optional	YES	Would be used for OEM ONLY, all settings that are required for the machine, and likely NEVER be needed by any customer.
To be defined	?	Optional	?	?

Hint: The mode change²⁴ only takes place, when the unit/machine is in **STOPPED** state. This means that exit and enter modes are in **STOPPED** state. The mode change will not make any change in PackML state. Therefore, a unit/machine in production mode and stopped state will enter for example Maintenance mode in **STOPPED** state. The mode change can also happen in **ABORTED** state, when entering Manual mode. The exit to Manual mode, is with restrictions because of safety issues.

²³ The mode change to mode EMPTY OUT might require a change in the EXECUTE state. This mode change can be done manual by operator or automatically by the unit/machine when it reaches the end of production order.

²⁴ According to ISA-TR88.00.02-2022 is it allowed to change mode in all PackML wait states.

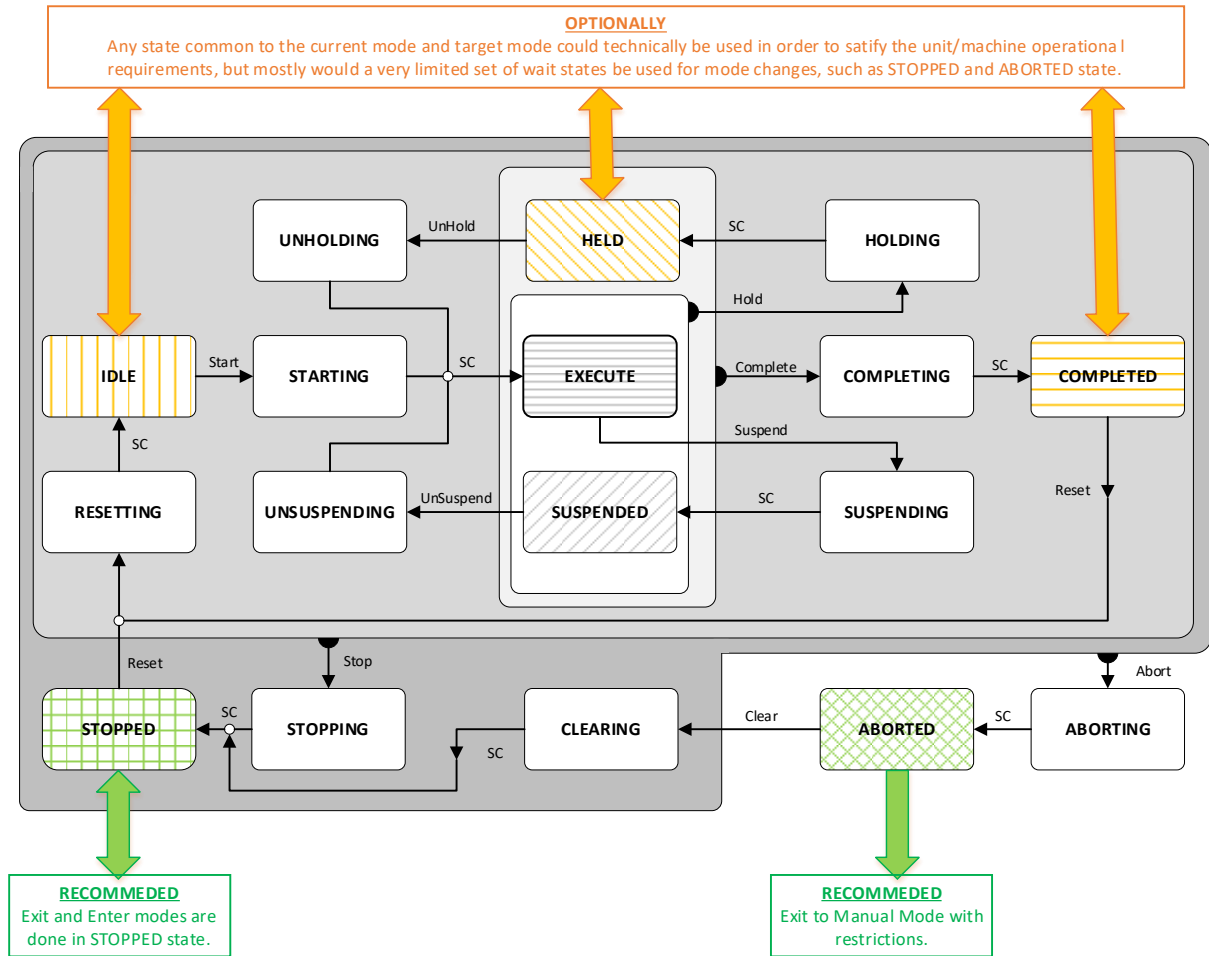


Figure 35 Recommended PackML states in which mode changes can be made

11. PACKTAGS

PackTags provide a uniform set of naming conventions for data elements used in the base state model. PackTags are named data elements used for open architecture, interoperable data exchange in automated machinery. This document includes the End Users minimum set of the data elements and their data types, defined in Table 19. PackTags are useful for communication between machines, as well as for exchange of data between machines and higher-level information systems.

Major changes have been made to the PackTags, by removal of the “Remote” Interface and the addition of PackTags that describe the Unit/Machine configuration via “Recipe” to a supervisory system.

The changes added from first version of the PackML Guideline is “Remote” Interface and the addition of PackTags that describe the Unit/Machine configuration via “Recipe” to a supervisory system. The Parameters have been separated into type of variable: REAL, STRING, LREAL and DINT.

Hint: Use the Pascal Case where the first letter of each word in an identifier is capitalized for readability. In order to ensure inter-operability with all systems, IEC 61131 is not case sensitive and therefore, it is recommended that the Pascal Case format is followed. Optionally, underscores (“_”) may also be used in place of the “dot” notation for legacy systems that do not support structured tagnames.

The table below indicates the minimum set of PackTags that the End Users recommend Machine Suppliers to implement. Of course, it is allowed for Machine Suppliers to implement additional PackTags. The minimum set of PackTags required are indicated in the column “End user Minimum tags”.

Table 19: Minimum STATUS PackTags²⁵

PackTag type	PackTag	Example of End user term	Datatype IEC 61131-3	TR 88.00.02 Minimum tags	End user Minimum tags
Status	StateCurrent	State	DINT	X	X
Status	UnitModeCurrent	Mode	DINT	X	X
Status	MachSpeed	Nominal Speed	REAL	X	X
Status	CurMachSpeed	Current Speed	REAL	X	X
Status	EquipmentInterlock.Blocked	Blockage	BOOL	X	X
Status	EquipmentInterlock.Starved	Starvation	BOOL	X	X
Status	Parameter_REAL [#]	Machine data/parameter	Array Structure		X
Status	Parameter_REAL [#].ID	Parameter ID	DINT		X
Status	Parameter_REAL [#].Name	Name of parameter	STRING		X
Status	Parameter_REAL [#].Unit	Unit of measure	STRING[6]		X
Status	Parameter_REAL [#].Value	Value of parameter	REAL		X
Status	Parameter_STRING [#]	Machine data/parameter	Array Structure		X
Status	Parameter_STRING [#].ID	Parameter ID	DINT		X
Status	Parameter_STRING [#].Name	Name of parameter	STRING		X
Status	Parameter_STRING [#].Unit	Unit of measure	STRING[6]		X
Status	Parameter_STRING [#].Value	Value of parameter	STRING		X
Status	Parameter_LREAL [#]	Machine data/parameter	Array Structure		X
Status	Parameter_LREAL [#].ID	Parameter ID	DINT		X

²⁵ ANSI/ISA TR88.00.02-2022 is the reference for all PackTags.

PackTag type	PackTag	Example of End user term	Datatype IEC 61131-3	TR 88.00.02 Minimum tags	End user Minimum tags
Status	Parameter_LREAL [#].Name	Name of parameter	STRING		X
Status	Parameter_LREAL [#].Unit	Unit of measure	STRING[6]		X
Status	Parameter_LREAL [#].Value	Value of parameter	LREAL		X
Status	Parameter_DINT [#]	Machine data/parameter	Array Structure		X
Status	Parameter_DINT [#].ID	Parameter ID	DINT		X
Status	Parameter_DINT [#].Name	Name of parameter	STRING		X
Status	Parameter_DINT [#].Unit	Unit of measure	STRING[6]		X
Status	Parameter_DINT [#].Value	Value of parameter	DINT		X
Status	RecipeCurrent	Number of Recipe	DINT		X
Status	RecipeRequested	Requested Recipe No	DINT		X
Status	RecipeChangeInProgress	Recipe is changing	BOOL		
Status	Recipe ²⁶ [#]	Recipe structure	Array Structure		X
Status	Recipe [#].ID	Recipe ID	DINT		X
Status	Recipe [#].Name	Recipe Name	STRING		X
Status	Recipe [#].Unit	Unit of Measure (E.g., Pc, Crtn, Case, ...)	STRING[6]		X
Status	Recipe [#].PrimaryQty	Primary	REAL		X
Status	Recipe [#].ProcessVariables ²⁷ .	Process variables			X
Status	Recipe [#].ProcessVariables.Parameter_REAL [#]	Process variables Machine data/parameter	Array Structure		X
Status	Recipe [#].ProcessVariables.Parameter_REAL [#].ID	Process variables Parameter ID	DINT		X
Status	Recipe [#].ProcessVariables.Parameter_REAL [#].Name	Process variables Name of parameter	STRING		X
Status	Recipe [#].ProcessVariables.Parameter_REAL [#].Unit	Process variables Unit of measure	STRING[6]		X
Status	Recipe [#].ProcessVariables.Parameter_REAL [#].Value	Process variables Value of parameter	REAL		X
Status	Recipe [#].ProcessVariables.Parameter_STRING [#]	Process variables Machine data/parameter	Array Structure		X
Status	Recipe [#].ProcessVariables.Parameter_STRING [#].ID	Process variables Parameter ID	DINT		X
Status	Recipe [#].ProcessVariables.Parameter_STRING [#].Name	Process variables Name of parameter	STRING		X
Status	Recipe [#].ProcessVariables.Parameter_STRING [#].Unit	Process variables Unit of measure	STRING[6]		X
Status	Recipe [#].ProcessVariables.Parameter_STRING [#].Value	Process variables Value of parameter	STRING		X
Status	Recipe [#].ProcessVariables.Parameter_LREAL [#]	Process variables Machine data/parameter	Array Structure		X
Status	Recipe [#].ProcessVariables.Parameter_LREAL [#].ID	Process variables Parameter ID	DINT		X
Status	Recipe [#].ProcessVariables.Parameter_LREAL [#].Name	Process variables Name of parameter	STRING		X
Status	Recipe [#].ProcessVariables.Parameter_LREAL [#].Unit	Process variables Unit of measure	STRING[6]		X
Status	Recipe [#].ProcessVariables.Parameter_LREAL [#].Value	Process variables Value of parameter	LREAL		X
Status	Recipe [#].ProcessVariables.Parameter_DINT [#]	Process variables Machine data/parameter	Array Structure		X
Status	Recipe [#].ProcessVariables.Parameter_DINT [#].ID	Process variables Parameter ID	DINT		X
Status	Recipe [#].ProcessVariables.Parameter_DINT [#].Name	Process variables Name of parameter	STRING		X
Status	Recipe [#].ProcessVariables.Parameter_DINT [#].Unit	Process variables Unit of measure	STRING[6]		X
Status	Recipe [#].ProcessVariables.Parameter_DINT [#].Value	Process variables Value of parameter	DINT		X
Status	Recipe [#].Ingredients ²⁸ .	Ingredients variables			X
Status	Recipe [#].Ingredients.Parameter_REAL [#]	Ingredients/parameter	Array Structure		X
Status	Recipe [#].Ingredients.Parameter_REAL [#].ID	Ingredients Parameter ID	DINT		X
Status	Recipe [#].Ingredients.Parameter_REAL [#].Name	Ingredients Name of parameter	STRING		X
Status	Recipe [#].Ingredients.Parameter_REAL [#].Unit	Ingredients Unit of measure	STRING[6]		X
Status	Recipe [#].Ingredients.Parameter_REAL [#].Value	Ingredients Value of parameter	REAL		X
Status	Recipe [#].Ingredients.Parameter_STRING [#]	Ingredients Machine data/parameter	Array Structure		X
Status	Recipe [#].Ingredients.Parameter_STRING [#].ID	Ingredients Parameter ID	DINT		X
Status	Recipe [#].Ingredients.Parameter_STRING [#].Name	Ingredients Name of parameter	STRING		X
Status	Recipe [#].Ingredients.Parameter_STRING [#].Unit	Ingredients Unit of measure	STRING[6]		X

²⁶ The data represented by the Status.Recipe can be used for Centerlining functionality.

²⁷ The ProcessVariables structure can be used for particular set points needed by the uni/machine for the processing of a specific recipe. ProcessVariables may include such things as speed, pressure, vacuum, time set points, limits. The ProcessVariables should be used by the Unit/Machine to configure according to the product going to be produced by the recipe.

²⁸ The Ingredients structure can be used to hold information for the raw materials that are used by the unit/machine in processing of a particular product according to the recipe. Ingredients may include number of raw materials.

PackTag type	PackTag	Example of End user term	Datatype IEC 61131-3	TR 88.00.02 Minimum tags	End user Minimum tags
Status	Recipe [#].Ingredients.Parameter_STRING [#].Value	Ingredients Value of parameter	STRING		X
Status	Recipe [#].Ingredients.Parameter_LREAL [#]	Ingredients Machine data/parameter	Array Structure		X
Status	Recipe [#].Ingredients.Parameter_LREAL [#].ID	Ingredients Parameter ID	DINT		X
Status	Recipe [#].Ingredients.Parameter_LREAL [#].Name	Ingredients Name of parameter	STRING		X
Status	Recipe [#].Ingredients.Parameter_LREAL [#].Unit	Ingredients Unit of measure	STRING[6]		X
Status	Recipe [#].Ingredients.Parameter_LREAL [#].Value	Ingredients Value of parameter	LREAL		X
Status	Recipe [#].Ingredients.Parameter_DINT [#]	Ingredients Machine data/parameter	Array Structure		X
Status	Recipe [#].Ingredients.Parameter_DINT [#].ID	Ingredients Parameter ID	DINT		X
Status	Recipe [#].Ingredients.Parameter_DINT [#].Name	Ingredients Name of parameter	STRING		X
Status	Recipe [#].Ingredients.Parameter_DINT [#].Unit	Ingredients Unit of measure	STRING[6]		X
Status	Recipe [#].Ingredients.Parameter_DINT [#].Value	Ingredients Value of parameter	DINT		X

X = Supported.

Table 20: Minimum ADMIN PackTags²⁹

PackTag type	PackTag	Example of End user term	Datatype IEC 61131-3	TR 88.00.02 Minimum tags	End user Minimum tags
Admin	Alarm[#]	Alarm	Array Structure		X
Admin	Alarm [#].Trigger	Trigger	BOOL		X
Admin	Alarm [#].ID	ID	DINT		X
Admin	Alarm [#].Value	Value	DINT		X
Admin	Warning[#]	Warning	Array Structure		X
Admin	Warning[#].Trigger	Trigger	BOOL		X
Admin	Warning[#].ID	ID	DINT		X
Admin	Warning[#].Value	Value	DINT		X
Admin	ProductData [#]	Product data array	Array Structure		X
Admin	ProductData [#].ID	Product identification	DINT		X
Admin	ProductData [#].ProcessedCount	Total count processed products	DINT	X	X
Admin	ProductData [#].DefectiveCount	Bad counted products	DINT	X	X
	StopReason	StopReasons variables	Array Structure		X
Admin	StopReason.ID	Event and stop reason	DINT	X	X
Admin	StopReason.Value	Detailed Error Information	DINT		X

X = Supported.

²⁹ ANSI/ISA TR88.00.02-2022 is the reference for all PackTags.

Table 21: Minimum COMMAND PackTags³⁰

PackTag type	PackTag	Example of End user term	Datatype IEC 61131-3	TR 88.00.02 Minimum tags	End user Minimum tags
Command	UnitMode	Mode	DINT	X	X
Command	UnitModeChangeRequest	Change mode	BOOL	X	X
Command	MachSpeed ³¹	Mach Speed	REAL	X	X
Command	CntrlCmd	Command	DINT	X	X
Command	CmdChangeRequest	Change command	BOOL	X	X
Command	Parameter_REAL [#]	Machine data/parameter	Array Structure		X
Command	Parameter_REAL [#].ID	Parameter ID	DINT		X
Command	Parameter_REAL [#].Name	Name of parameter	STRING		X
Command	Parameter_REAL [#].Unit	Unit of measure	STRING[6]		X
Command	Parameter_REAL [#].Value	Value of parameter	REAL		X
Command	Parameter_STRING [#]	Machine data/parameter	Array Structure		X
Command	Parameter_STRING [#].ID	Parameter ID	DINT		X
Command	Parameter_STRING [#].Name	Name of parameter	STRING		X
Command	Parameter_STRING [#].Unit	Unit of measure	STRING[6]		X
Command	Parameter_STRING [#].Value	Value of parameter	STRING		X
Command	Parameter_LREAL [#]	Machine data/parameter	Array Structure		X
Command	Parameter_LREAL [#].ID	Parameter ID	DINT		X
Command	Parameter_LREAL [#].Name	Name of parameter	STRING		X
Command	Parameter_LREAL [#].Unit	Unit of measure	STRING[6]		X
Command	Parameter_LREAL [#].Value	Value of parameter	LREAL		X
Command	Parameter_DINT [#]	Machine data/parameter	Array Structure		X
Command	Parameter_DINT [#].ID	Parameter ID	DINT		X
Command	Parameter_DINT [#].Name	Name of parameter	STRING		X
Command	Parameter_DINT [#].Unit	Unit of measure	STRING[6]		X
Command	Parameter_DINT [#].Value	Value of parameter	DINT		X
Command	SelectedRecipe	Number of Recipe	DINT		X
Command	RecipeChangeRequest	Requested Recipe No	DINT		X
Command	Recipe ³² [#]	Recipe structure	Array Structure		X
Command	Recipe [#].ID	Recipe ID	DINT		X
Command	Recipe [#].Name	Recipe Name	STRING		X
Command	Recipe [#].Unit	Unit of Measure (E.g., Pc, Crtn, Case, ...)	STRING[6]		X
Command	Recipe [#].PrimaryQty	Primary	REAL		X
Command	Recipe [#].ProcessVariables ³³ .	Process variables			X
Command	Recipe [#].ProcessVariables.Parameter_REAL [#]	Process variables Machine data/parameter	Array Structure		X
Command	Recipe [#].ProcessVariables.Parameter_REAL [#].ID	Process variables Parameter ID	DINT		X
Command	Recipe [#].ProcessVariables.Parameter_REAL [#].Name	Process variables Name of parameter	STRING		X
Command	Recipe [#].ProcessVariables.Parameter_REAL [#].Unit	Process variables Unit of measure	STRING[6]		X
Command	Recipe [#].ProcessVariables.Parameter_REAL [#].Value	Process variables Value of parameter	REAL		X
Command	Recipe [#].ProcessVariables.Parameter_STRING [#]	Process variables Machine data/parameter	Array Structure		X
Command	Recipe [#].ProcessVariables.Parameter_STRING [#].ID	Process variables Parameter ID	DINT		X
Command	Recipe [#].ProcessVariables.Parameter_STRING [#].Name	Process variables Name of parameter	STRING		X
Command	Recipe [#].ProcessVariables.Parameter_STRING [#].Unit	Process variables Unit of measure	STRING[6]		X
Command	Recipe [#].ProcessVariables.Parameter_STRING [#].Value	Process variables Value of parameter	STRING		X
Command	Recipe [#].ProcessVariables.Parameter_LREAL [#]	Process variables Machine data/parameter	Array Structure		X
Command	Recipe [#].ProcessVariables.Parameter_LREAL [#].ID	Process variables Parameter ID	DINT		X

³⁰ ANSI/ISA TR88.00.02-2022 is the reference for all PackTags.

³¹ In case the unit/machine is able to run different products with different sizes of primary packages within one line, the calculation and view of MachSpeed needs to be part of the Machine Recipe Parameters, to help the operator to understand the machine and its speed. The reason is that the speed of e.g. a small feeding system may change because the primary package size changes.

³² The Command.Recipe is used for configuring the unit/machine and can be used for Order Handling functionality from a Supervisory Control System

³³ The ProcessVariables structure can be used for particular set points needed by the unit/machine for the processing of a specific recipe. ProcessVariables may include such things as speed, pressure, vacuum, time set points, limits. The ProcessVariables should be used by the Unit/Machine to configure according to the product going to be produced by the recipe.

PackTag type	PackTag	Example of End user term	Datatype IEC 61131-3	TR 88.00.02 Minimum tags	End user Minimum tags
Command	Recipe [#].ProcessVariables.Parameter_REAL [#].Name	Process variables Name of parameter	STRING		X
Command	Recipe [#].ProcessVariables.Parameter_REAL [#].Unit	Process variables Unit of measure	STRING[6]		X
Command	Recipe [#].ProcessVariables.Parameter_REAL [#].Value	Process variables Value of parameter	LREAL		X
Command	Recipe [#].ProcessVariables.Parameter_DINT [#]	Process variables Machine data/parameter	Array Structure		X
Command	Recipe [#].ProcessVariables.Parameter_DINT [#].ID	Process variables Parameter ID	DINT		X
Command	Recipe [#].ProcessVariables.Parameter_DINT [#].Name	Process variables Name of parameter	STRING		X
Command	Recipe [#].ProcessVariables.Parameter_DINT [#].Unit	Process variables Unit of measure	STRING[6]		X
Command	Recipe [#].ProcessVariables.Parameter_DINT [#].Value	Process variables Value of parameter	DINT		X
Command	Recipe [#].Ingredients ³⁴ .	Ingredients variables			X
Command	Recipe [#].Ingredients.Parameter_REAL [#]	Ingredients/parameter	Array Structure		X
Command	Recipe [#].Ingredients.Parameter_REAL [#].ID	Ingredients Parameter ID	DINT		X
Command	Recipe [#].Ingredients.Parameter_REAL [#].Name	Ingredients Name of parameter	STRING		X
Command	Recipe [#].Ingredients.Parameter_REAL [#].Unit	Ingredients Unit of measure	STRING[6]		X
Command	Recipe [#].Ingredients.Parameter_REAL [#].Value	Ingredients Value of parameter	REAL		X
Command	Recipe [#].Ingredients.Parameter_STRING [#]	Ingredients Machine data/parameter	Array Structure		X
Command	Recipe [#].Ingredients.Parameter_STRING [#].ID	Ingredients Parameter ID	DINT		X
Command	Recipe [#].Ingredients.Parameter_STRING [#].Name	Ingredients Name of parameter	STRING		X
Command	Recipe [#].Ingredients.Parameter_STRING [#].Unit	Ingredients Unit of measure	STRING[6]		X
Command	Recipe [#].Ingredients.Parameter_STRING [#].Value	Ingredients Value of parameter	STRING		X
Command	Recipe [#].Ingredients.Parameter_REAL [#]	Ingredients Machine data/parameter	Array Structure		X
Command	Recipe [#].Ingredients.Parameter_REAL [#].ID	Ingredients Parameter ID	DINT		X
Command	Recipe [#].Ingredients.Parameter_REAL [#].Name	Ingredients Name of parameter	STRING		X
Command	Recipe [#].Ingredients.Parameter_REAL [#].Unit	Ingredients Unit of measure	STRING[6]		X
Command	Recipe [#].Ingredients.Parameter_REAL [#].Value	Ingredients Value of parameter	LREAL		X
Command	Recipe [#].Ingredients.Parameter_DINT [#]	Ingredients Machine data/parameter	Array Structure		X
Command	Recipe [#].Ingredients.Parameter_DINT [#].ID	Ingredients Parameter ID	DINT		X
Command	Recipe [#].Ingredients.Parameter_DINT [#].Name	Ingredients Name of parameter	STRING		X
Command	Recipe [#].Ingredients.Parameter_DINT [#].Unit	Ingredients Unit of measure	STRING[6]		X
Command	Recipe [#].Ingredients.Parameter_DINT [#].Value	Ingredients Value of parameter	DINT		X

X = Supported.

11.1 DESCRIPTION AND DEFINITION OF PACKTAGS

See details about the PackTags in the reference *ANSI/ISA-TR88.00.02-2022, Machine and unit/machine States, An Implementation example of ANSI/ISA-88.00.01*.

11.2 MIGRATION FROM 2015 TOWARD 2022 PACKTAGS

Below is an overview of the differences between PackTags 2015 and PackTags 2022.

This tables below represent the Status, Admin and Commands Tags according to the ISA-TR88.00.02-2015 standard. The individual table compare the 2015 and 2022 PackTags..The tables have the columns with the following text are specified:

- **ISA-TR88-2015 PackTags:** PackTags in the column are the 2015 PackTags.
- **ISA-TR88-2022 PackTags alignment**

³⁴ The Ingredients structure can be used to hold information for the raw materials that are used by the unit/machine in processing of a particular product according to the recipe. Ingredients may include number of raw materials.

- Cell with x and Green background indicates that ISA-TR88.00.02-2022 is in accordance with ISA-TR88.00.02-2015.
- Cell with (-) and red background indicate that it is not included in ISA-TR88.00.02-2022.
- Cell with (x) and yellow background indicate that the content is included in ISA-TR88.00.02-2022 but is not of the same structure and/or data types.
- **Comments:** Describes the difference between the two standard PackTags.

Table 22 Status PackTags

ISA-TR88-2015 PackTags	ISA-TR88-2022 PackTags alignment	Comments
UnitName	(x)	Changed to PACKML v2022
UnitName.Status	x	
UnitName.Status.UnitModeCurrent	x	
UnitName.Status.UnitModeRequested	x	
UnitName.Status.UnitModeChangeInProgress	x	
UnitName.Status.StateCurrent	x	
UnitName.Status.StateRequested	x	
UnitName.Status.StateChangeInProgress	x	
UnitName.Status.MachSpeed	x	
UnitName.Status.CurMachSpeed	x	
UnitName.Status.MaterialInterlock[#]	x	
UnitName.Status.EquipmentInterlock	x	
UnitName.Status.EquipmentInterlock.Blocked	x	
UnitName.Status.EquipmentInterlock.Starved	x	
UnitName.Status.RemoteInterface[#]	-	Remote Interface is not included in the ISA-TR88.00.02-2022.
UnitName.Status.RemoteInterface[#].Number	-	
UnitName.Status.RemoteInterface[#].ControlCmdNumber	-	
UnitName.Status.RemoteInterface[#].CmdValue	-	
UnitName.Status.RemoteInterface[#].Parameter[#]	-	
UnitName.Status.RemoteInterface[#].Parameter[#].ID	-	
UnitName.Status.RemoteInterface[#].Parameter[#].Name.	-	
UnitName.Status.RemoteInterface[#].Parameter[#].Unit	-	
UnitName.Status.RemoteInterface[#].Parameter[#].Value	-	
UnitName.Status.Parameter[#]	(x)	The PackTag array for parameters are separated into the different types of variables that need to be included. These types are: UnitName.Status.Parameter_REAL[#] UnitName.Status.Parameter_STRING[#] UnitName.Status.Parameter_LREAL[#] UnitName.Status.Parameter_DINT[#]
UnitName.Status.Parameter[#].ID	(x)	
UnitName.Status.Parameter[#].Name	(x)	
UnitName.Status.Parameter[#].Unit	(x)	
UnitName.Status.Parameter[#].Value	(x)	
UnitName.Status.Product[#]	(x)	The PackTag array "Product" is replaced by <i>UnitName.Status.Recipe[#]</i>
UnitName.Status.Product[#].ProductID	(x)	Changed to <i>UnitName.Status.Recipe[#].ID</i>
UnitName.Status.Product[#].ProcessVariables[#]	(x)	The PackTag array is changed to UnitName.Status.Recipe[#].ProcessVariables.Parameter_REAL[#], UnitName.Status.Recipe[#].ProcessVariables.Parameter_STRING[#], UnitName.Status.Recipe[#].ProcessVariables.Parameter_LREAL[#], UnitName.Status.Recipe[#].ProcessVariables.Parameter_DINT[#]
UnitName.Status.Product[#].ProcessVariables[#].ID	(x)	
UnitName.Status.Product[#].ProcessVariables[#].Name	(x)	
UnitName.Status.Product[#].ProcessVariables[#].Unit	(x)	
UnitName.Status.Product[#].ProcessVariables[#].Value	(x)	
UnitName.Status.Product[#].Ingredients[#]	(x)	The PackTag array is changed to UnitName.Status.Recipe[#].Ingredients.Parameter_REAL[#], UnitName.Status.Recipe[#].Ingredients.Parameter_STRING[#], UnitName.Status.Recipe[#].Ingredients.Parameter_LREAL[#], UnitName.Status.Recipe[#].Ingredients.Parameter_DINT[#]
UnitName.Status.Product[#].Ingredients[#].IngredientID	(x)	
UnitName.Status.Product[#].Ingredients[#].Parameter[#]	(x)	
UnitName.Status.Product[#].Ingredients[#].Parameter[#].ID	(x)	
UnitName.Status.Product[#].Ingredients[#].Parameter[#].Name	(x)	
UnitName.Status.Product[#].Ingredients[#].Parameter[#].Unit	(x)	
UnitName.Status.Product[#].Ingredients[#].Parameter[#].Value	(x)	

Table 23 Admin PackTags

ISA-TR88-2015 PackTags	ISA-TR88-2022 PackTags alignment	Comments
UnitName	(x)	Changed to PACKML v2022
UnitName.Admin	x	
UnitName.Admin.Parameter[#]	(x)	The PackTag array for parameters are seperated into the different types of variables that need to be included. These types are: UnitName.Admin.Parameter_REAL[#] UnitName.Admin.Parameter_STRING[#] UnitName.Admin.Parameter_LREAL[#] UnitName.Admin.Parameter_DINT[#]
UnitName.Admin.Parameter[#].ID	(x)	
UnitName.Admin.Parameter[#].Name	(x)	
UnitName.Admin.Parameter[#].Unit	(x)	
UnitName.Admin.Parameter[#].Value	(x)	
UnitName.Admin.Alarm[#]	x	
UnitName.Admin.Alarm[#].Trigger	x	
UnitName.Admin.Alarm[#].ID	x	
UnitName.Admin.Alarm[#].Value	x	
UnitName.Admin.Alarm[#].Message	x	
UnitName.Admin.Alarm[#].Category	x	
UnitName.Admin.Alarm[#].DateTime[#]	(x)	DateTime adn AckDateTime is not an array, but put into a structure UnitName.Admin.Alarm[#].DateTime.Year UnitName.Admin.Alarm[#].DateTime.Month UnitName.Admin.Alarm[#].DateTime.Day UnitName.Admin.Alarm[#].DateTime.Hour UnitName.Admin.Alarm[#].DateTime.Minute UnitName.Admin.Alarm[#].DateTime.Second
UnitName.Admin.Alarm[#].AckDateTime[#]	(x)	
UnitName.Admin.AlarmExtent	x	
UnitName.Admin.AlarmHistory[#]	x	
UnitName.Admin.AlarmHistory[#].Trigger	x	
UnitName.Admin.AlarmHistory[#].ID	x	
UnitName.Admin.AlarmHistory[#].Value	x	
UnitName.Admin.AlarmHistory[#].Message	x	
UnitName.Admin.AlarmHistory[#].Category	x	
UnitName.Admin.AlarmHistory[#].DateTime[#]	(x)	DateTime adn AckDateTime is not an array, but put into a structure UnitName.Admin.AlarmHistory[#].DateTime.Year UnitName.Admin.AlarmHistory[#].DateTime.Month UnitName.Admin.AlarmHistory[#].DateTime.Day UnitName.Admin.AlarmHistory[#].DateTime.Hour UnitName.Admin.AlarmHistory[#].DateTime.Minute UnitName.Admin.AlarmHistory[#].DateTime.Second
UnitName.Admin.AlarmHistory[#].AckDateTime[#]	(x)	
UnitName.Admin.AlarmHistoryExtent	x	
UnitName.Admin.StopReason[#]	(x)	The StopReason Array are changed to a simple datatype, and no array. UnitName.Admin.StopReason
UnitName.Admin.StopReason[#].Trigger	(x)	
UnitName.Admin.StopReason[#].ID	(x)	DateTime adn AckDateTime is not an array, but put into a structure UnitName.Admin.StopReason.DateTime.Year UnitName.Admin.StopReason.DateTime.Month UnitName.Admin.StopReason.DateTime.Day UnitName.Admin.StopReason.DateTime.Hour UnitName.Admin.StopReason.DateTime.Minute UnitName.Admin.StopReason.DateTime.Second
UnitName.Admin.StopReason[#].Value	(x)	
UnitName.Admin.StopReason[#].Message	(x)	
UnitName.Admin.StopReason[#].Category	(x)	
UnitName.Admin.StopReason[#].DateTime[#]	(x)	
UnitName.Admin.StopReason[#].AckDateTime[#]	(x)	
UnitName.Admin.StopReasonExtent	-	Removed - No longer part of the PackTags
UnitName.Admin.Warning[#]	x	
UnitName.Admin.Warning[#].Trigger	x	
UnitName.Admin.Warning[#].ID	x	
UnitName.Admin.Warning[#].Value	x	
UnitName.Admin.Warning[#].Message	x	
UnitName.Admin.Warning[#].Category	x	
UnitName.Admin.Warning[#].DateTime[#]	(x)	DateTime adn AckDateTime is not an array, but put into a structure UnitName.Admin.Warning.DateTime.Year UnitName.Admin.Warning.DateTime.Month UnitName.Admin.Warning.DateTime.Day UnitName.Admin.Warning.DateTime.Hour UnitName.Admin.Warning.DateTime.Minute UnitName.Admin.Warning.DateTime.Second
UnitName.Admin.Warning[#].AckDateTime[#]	(x)	
UnitName.Admin.WarningExtent	x	

ISA-TR88-2015 PackTags	ISA-TR88-2022 PackTags alignment	Comments
UnitName.Admin.ModeCurrentTime[#]	(x)	The StopReason Array are changed to a simple datatype, and no array. UnitName.Admin.StopReason
UnitName.Admin.ModeCumulativeTime[#]	(x)	
UnitName.Admin.StateCurrentTime[#,#]	(x)	
UnitName.Admin.StateCumulativeTime[#,#]	(x)	
UnitName.Admin.ProdConsumedCount[#]	(x)	The structure is change to <i>UnitName.Admin.ProductData[#]</i> .
UnitName.Admin.ProdConsumedCount[#].ID	(x)	
UnitName.Admin.ProdConsumedCount[#].Name	(x)	
UnitName.Admin.ProdConsumedCount[#].Unit	(x)	And the counters are changed to <i>UnitName.Admin.ProductData[#].PrimaryQty</i> <i>UnitName.Admin.ProductData[#].ConsumedCount</i> <i>UnitName.Admin.ProductData[#].ProcessedCount</i> <i>UnitName.Admin.ProductData[#].DefectiveCount</i> <i>UnitName.Admin.ProductData[#].AccConsumedCount</i> <i>UnitName.Admin.ProductData[#].AccProcessedCount</i> <i>UnitName.Admin.ProductData[#].AccDefectiveCount</i>
UnitName.Admin.ProdConsumedCount[#].Count	(x)	
UnitName.Admin.ProdConsumedCount[#].AccCount	(x)	
UnitName.Admin.ProdProcessedCount[#]	(x)	
UnitName.Admin.ProdProcessedCount[#].ID	(x)	
UnitName.Admin.ProdProcessedCount[#].Name	(x)	
UnitName.Admin.ProdProcessedCount[#].Unit	(x)	
UnitName.Admin.ProdProcessedCount[#].Count	(x)	
UnitName.Admin.ProdProcessedCount[#].AccCount	(x)	
UnitName.Admin.ProdDefectiveCount[#]	(x)	
UnitName.Admin.ProdDefectiveCount[#].ID	(x)	
UnitName.Admin.ProdDefectiveCount[#].Name	(x)	
UnitName.Admin.ProdDefectiveCount[#].Unit	(x)	
UnitName.Admin.ProdDefectiveCount[#].Count	(x)	
UnitName.Admin.ProdDefectiveCount[#].AccCount	(x)	
UnitName.Admin.ProdDefectiveCount[0].Count	(x)	
UnitName.Admin.ProdDefectiveCount[1].Count	(x)	
UnitName.Admin.ProdDefectiveCount[2].Count	(x)	
UnitName.Admin.ProdDefectiveCount[3].Count	(x)	
UnitName.Admin.ProdDefectiveCount[4].Count	(x)	
UnitName.Admin.ProdDefectiveCount[5].Count	(x)	
UnitName.Admin.ProdDefectiveCount[6].Count	(x)	
UnitName.Admin.AccTimeSinceReset	(x)	
UnitName.Admin.MachDesignSpeed	X	
UnitName.Admin.StatesDisabled	(x)	Is changed into two parameteres: <i>UnitName.Admin.DisabledStatesCfg[#]</i> <i>UnitName.Admin.CurDisabledStates</i> As well as Mode configuration is added: <i>UnitName.Admin.ModeTransitionCfg[#]</i> <i>UnitName.Admin.EnabledModesCfg</i>
UnitName.Admin.PLCDateTime[#]	(x)	PLCDateTime is not an array, but put into a structure <i>UnitName.Admin.Warning.DateTime.Year</i> <i>UnitName.Admin.Warning.DateTime.Month</i> <i>UnitName.Admin.Warning.DateTime.Day</i> <i>UnitName.Admin.Warning.DateTime.Hour</i> <i>UnitName.Admin.Warning.DateTime.Minute</i> <i>UnitName.Admin.Warning.DateTime.Second</i>

Table 24 Command PackTags

ISA-TR88-2015 PackTags	ISA-TR88-2022 PackTags alignment	Comments
UnitName	(x)	Changed to PACKML v2022
UnitName.Command	x	
UnitName.Command.UnitMode	x	
UnitName.Command.UnitModeChangeRequest	x	
UnitName.Command.MachSpeed	x	
UnitName.Command.MaterialInterlock[#]	x	
UnitName.Command.CntrlCmd	x	
UnitName.Command.CmdChangeRequest	x	
UnitName.Command.RemoteInterface[#]	-	Remote interface removed from ISA-TR88.00.02-2022.
UnitName.Command.RemoteInterface[#].Number	-	
UnitName.Command.RemoteInterface[#].ControlCmdNumber	-	
UnitName.Command.RemoteInterface[#].CmdValue	-	
UnitName.Command.RemoteInterface[#].Parameter[#]	-	
UnitName.Command.RemoteInterface[#].Parameter[#].ID	-	
UnitName.Command.RemoteInterface[#].Parameter[#].Name	-	
UnitName.Command.RemoteInterface[#].Parameter[#].Unit	-	
UnitName.Command.RemoteInterface[#].Parameter[#].Value	-	
UnitName.Command.Parameter[#]	(x)	The PackTag array for parameters are seperated into the different types of variables that need to be included. These types are: UnitName.Command.Parameter_REAL[#] UnitName.Command.Parameter_STRING[#] UnitName.Command.Parameter_LREAL[#] UnitName.Command.Parameter_DINT[#]
UnitName.Command.Parameter[#].ID	(x)	
UnitName.Command.Parameter[#].Name	(x)	
UnitName.Command.Parameter[#].Unit	(x)	
UnitName.Command.Parameter[#].Value	(x)	
UnitName.Command.Product[#]	(x)	The PackTag array "Product" is replaced by <i>UnitName.Command.Recipe[#]</i>
UnitName.Command.Product[#].ProductID	(x)	Changed to <i>UnitName.Command.Recipe[#].ID</i>
UnitName.Command.Product[#].ProcessVariables[#]	(x)	The PackTag array is changed to UnitName.Command.Recipe[#].ProcessVariables.Parameter_REAL[#], UnitName.Command.Recipe[#].ProcessVariables.Parameter_STRING[#], UnitName.Command.Recipe[#].ProcessVariables.Parameter_LREAL[#] UnitName.Command.Recipe[#].ProcessVariables.Parameter_DINT[#]
UnitName.Command.Product[#].ProcessVariables[#].ID	(x)	
UnitName.Command.Product[#].ProcessVariables[#].Name	(x)	
UnitName.Command.Product[#].ProcessVariables[#].Unit	(x)	
UnitName.Command.Product[#].ProcessVariables[#].Value	(x)	
UnitName.Command.Product[#].Ingredients[#]	(x)	The PackTag array is changed to UnitName.Command.Recipe[#].Ingredients.Parameter_REAL[#], UnitName.Command.Recipe[#].Ingredients.Parameter_STRING[#], UnitName.Command.Recipe[#].Ingredients.Parameter_LREAL[#] UnitName.Command.Recipe[#].Ingredients.Parameter_DINT[#]
UnitName.Command.Product[#].Ingredients[#].IngredientID	(x)	
UnitName.Command.Product[#].Ingredients[#].Parameter[#]	(x)	
UnitName.Command.Product[#].Ingredients[#].Parameter[#].ID	(x)	
UnitName.Command.Product[#].Ingredients[#].Parameter[#].Name	(x)	
UnitName.Command.Product[#].Ingredients[#].Parameter[#].Unit	(x)	
UnitName.Command.Product[#].Ingredients[#].Parameter[#].Value	(x)	

12. EXAMPLE OF UNIT CONTROLLER FUNCTIONALITY

The elements in this section illustrates eight main functions for the Unit Controller functionality:

1. Prepare unit/machine (Prepare for production)
2. Start unit/machine (Start production)
3. Stop unit/machine (Stop production)
 - a. Stop unit/machine (Stop production)
 - b. Complete unit/machine (Complete production)
4. Pause on unit/machine
5. Stop and restart unit/machine
6. Download new machine parameters (Order change)
7. Error and warnings on unit/machine
8. Safety circuit activated

The illustrations below are based on one unit, with material flows from upstream units and to downstream units.

12.1 SYNTAX AND SYMBOLIC DESCRIPTION

Below is an illustration of the syntax and symbolic used in the use case scenarios.

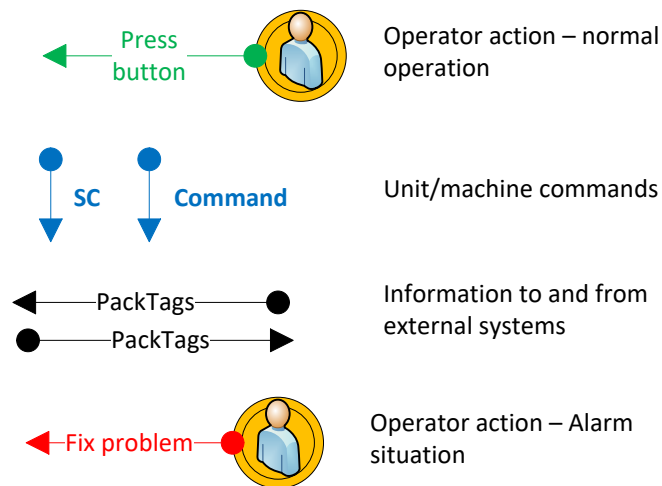


Figure 36: syntax and symbolic for use case scenarios

The operator starts the unit/machine for production from the unit panel/HMI or from an external system. When the operator starts the selected program on the unit/machine, the parameters are loaded into the unit. The unit/machine parameters can either be typed in on the unit/machine panel/HMI or the parameters can be downloaded to the unit/machine from an external system.

The operator or the external system starts the unit/machine according to the PackML standard.

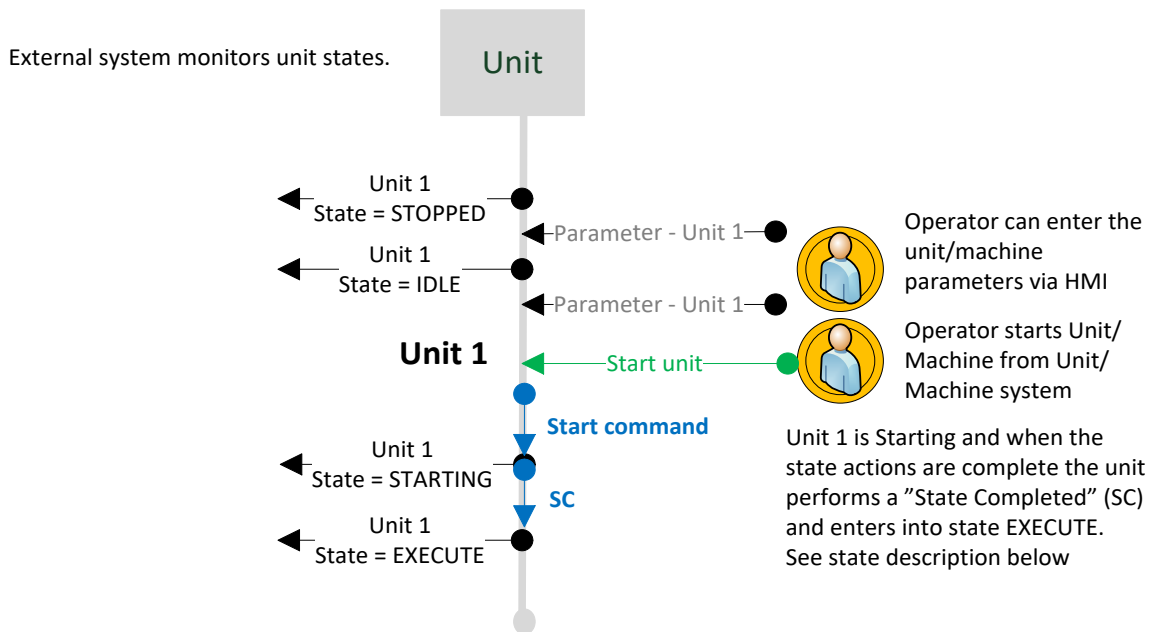


Figure 39: Start unit/machine from unit panel/HMI

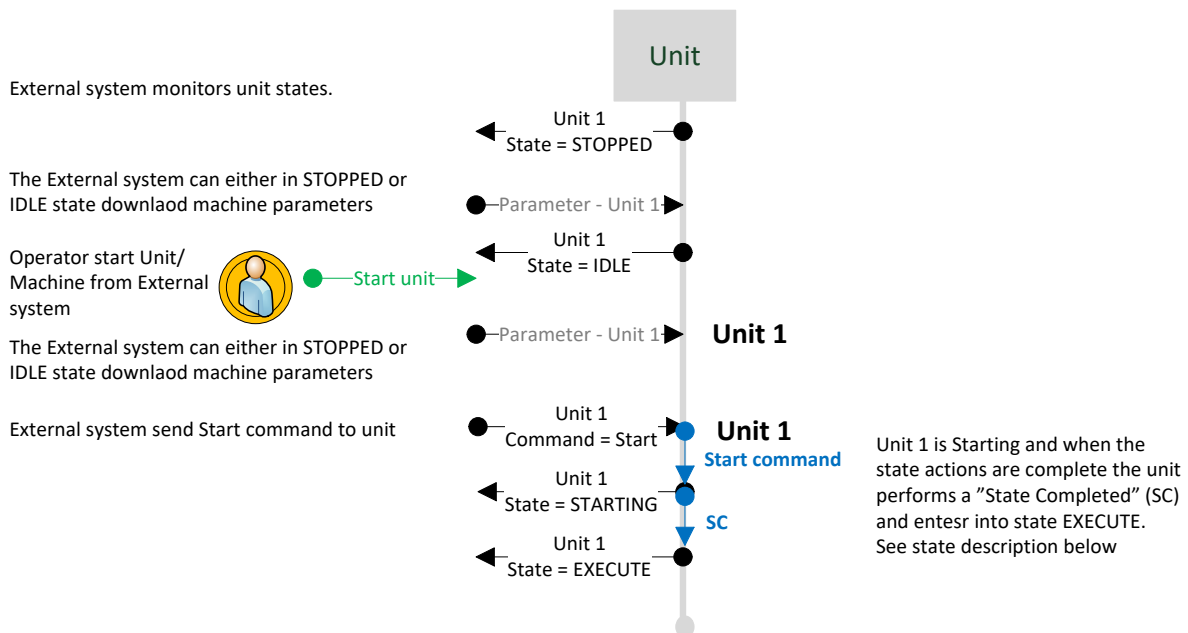


Figure 40: Start unit/machine from external system

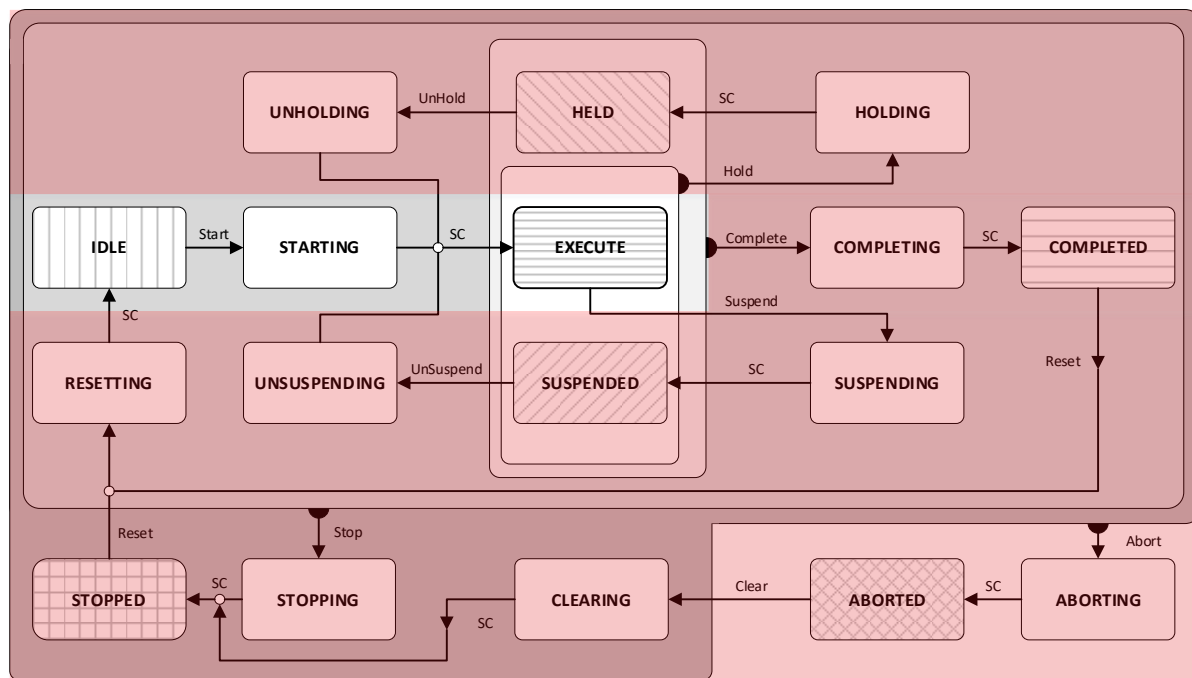


Figure 41: The states in the PackML state model in focus for Start unit/machine

12.4 STOP UNIT/MACHINE FROM UNIT PANEL/HMI

The unit can move to the COMPLETE state when the product counter reaches the designated value in the machine parameters. The unit will complete processing, empty, and stop producing products and then enter the COMPLETE state.

Hint: It can be necessary to define what is going to happen at the unit if there is remaining products on the inlet/infeed when the unit is completing.

Hint: The Machine Supplier and End User of a unit/machine need to define if a unit/machine needs to be stopped after a completion of a production order (product counter reached according to a production order). It may depend on when the unit/machine parameters are downloaded or typed into the unit/machine by operator.

Hint: A unit/machine in suspended state because of starvation can only complete the production order by a stop command from the operator – either on the unit/machine itself or from the supervisory control system.

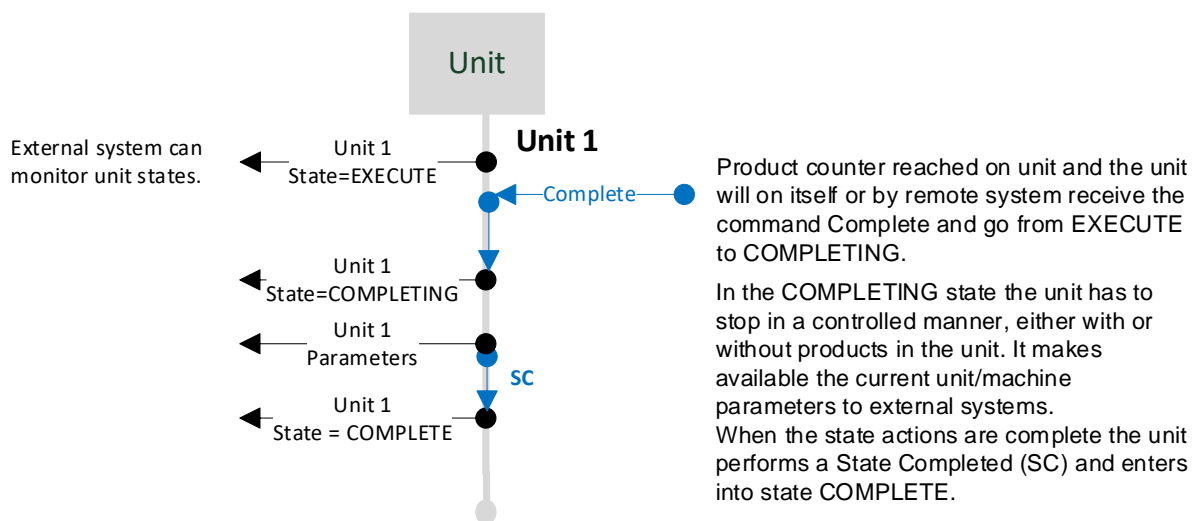


Figure 42: Completion from unit

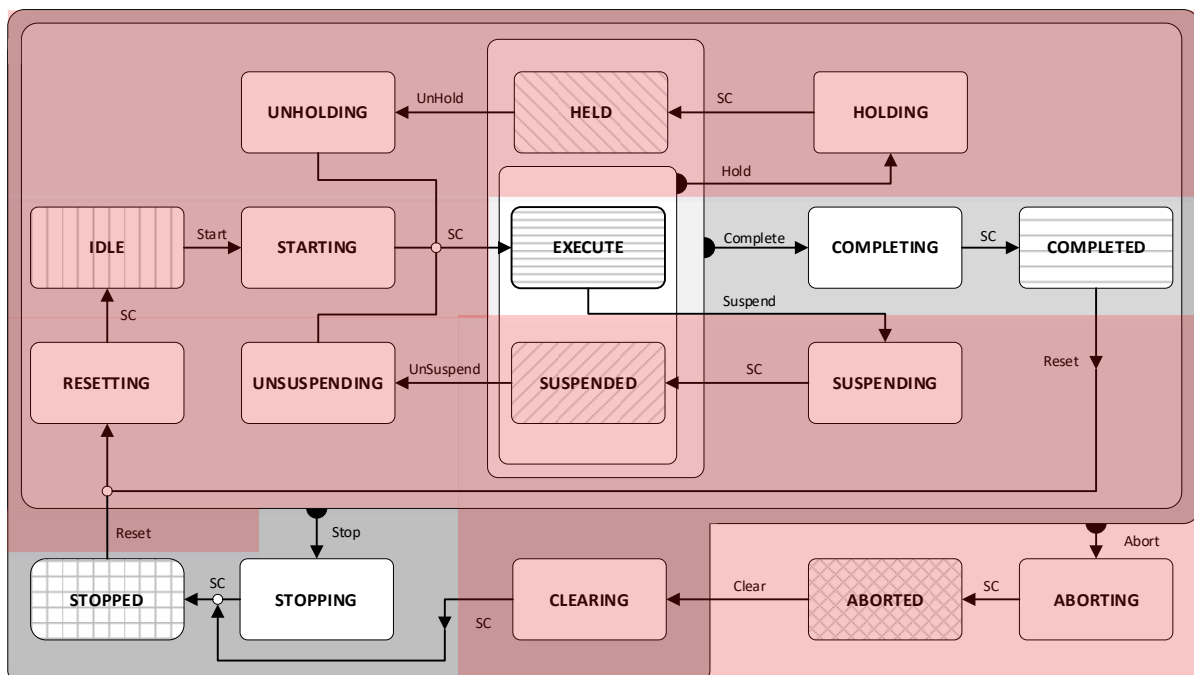


Figure 43: The states in the PackML state model in focus for Completion from unit or Remote Control

12.5 STOP UNIT/MACHINE FROM EXTERNAL SYSTEM

An external system can continuously acquire the number of produced products from the units. The external system also knows the number of products in the process. The external system can compare the number of produced products with the parameter value received from production

order data. When the number of products equals the parameter value (100%), then the external system can send a message to the operator.

The operator can stop the unit or the production can be stopped from the external system. The unit will stop, enter the STOPPING state, empty out (if necessary) and stop production and enter the STOPPED state. When the unit stops and has completed the production, the unit/machine parameters can be uploaded into the external system.

Hint: It can be necessary to decide what is going to happen at the unit if there are remaining products on the inlet/infeed, when the unit is stopping.

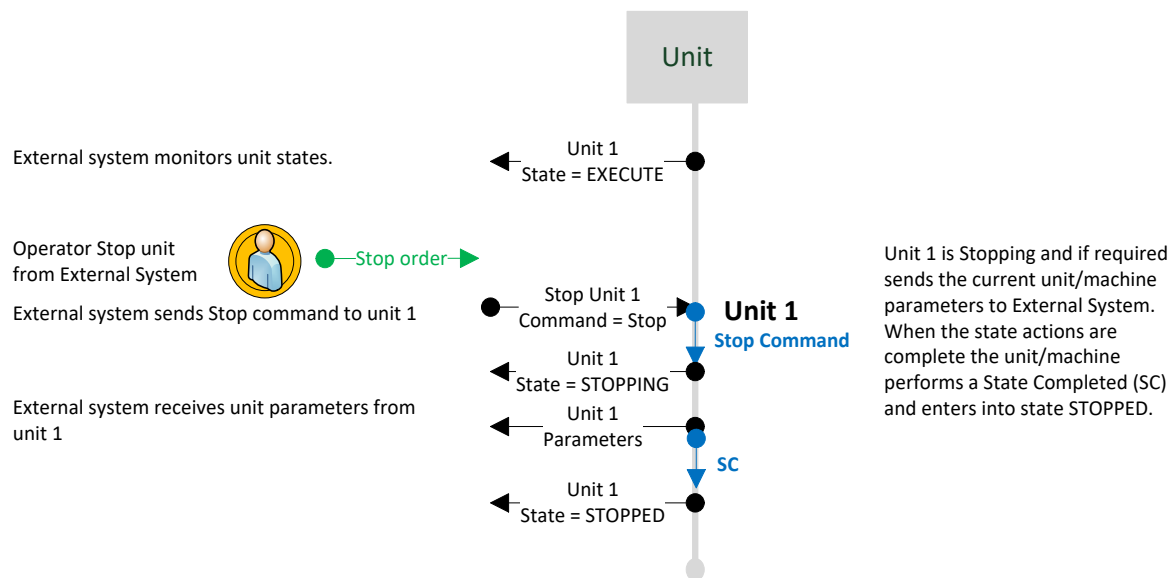


Figure 44: Completion from External system

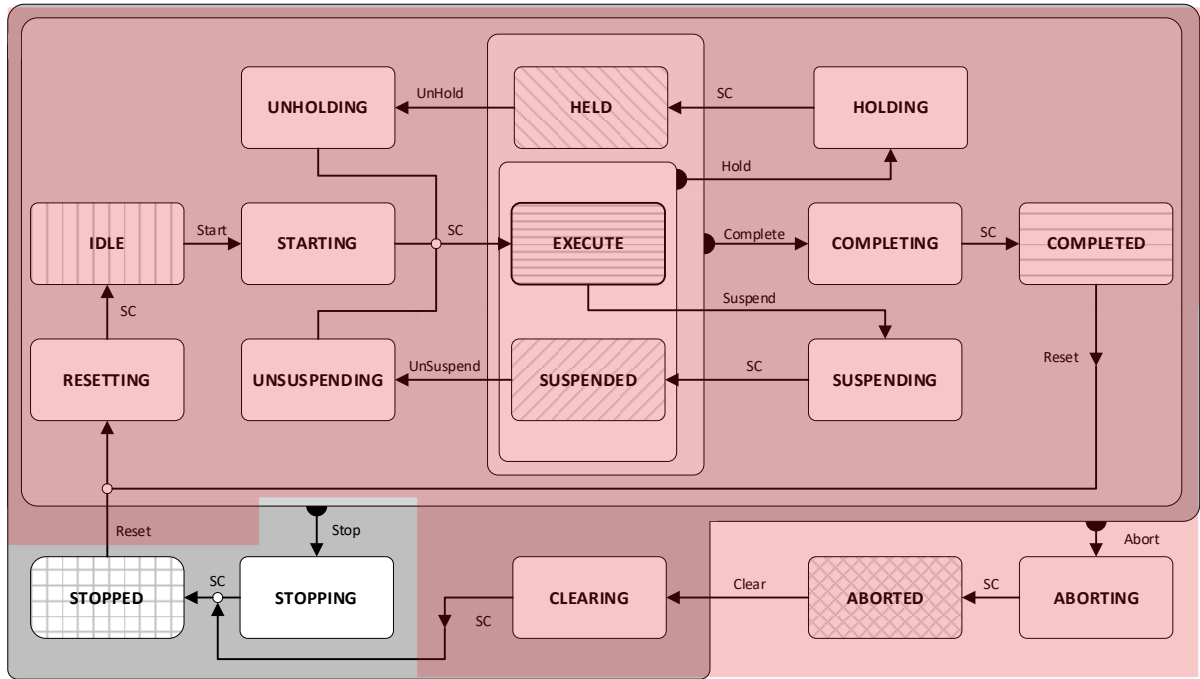


Figure 45: The states in the PackML state model in focus for Stop Order from External System

12.6 OPERATOR BREAK AND PAUSE ON UNIT/MACHINE

The operator decides to make a temporary stop and the unit/machine is suspended. Suspension of a unit/machine is typically used for shorter breaks or meetings.

The external system can suspend the unit.

Hint: The machine Supplier and End User need to decide what is going to happen with the OEE parameters in the unit.

The StopReason for the unit stop is to be generated directly in the units, and is sent unchanged to the external system.

It is also to be defined if the products can remain in the unit or the unit has to empty out.

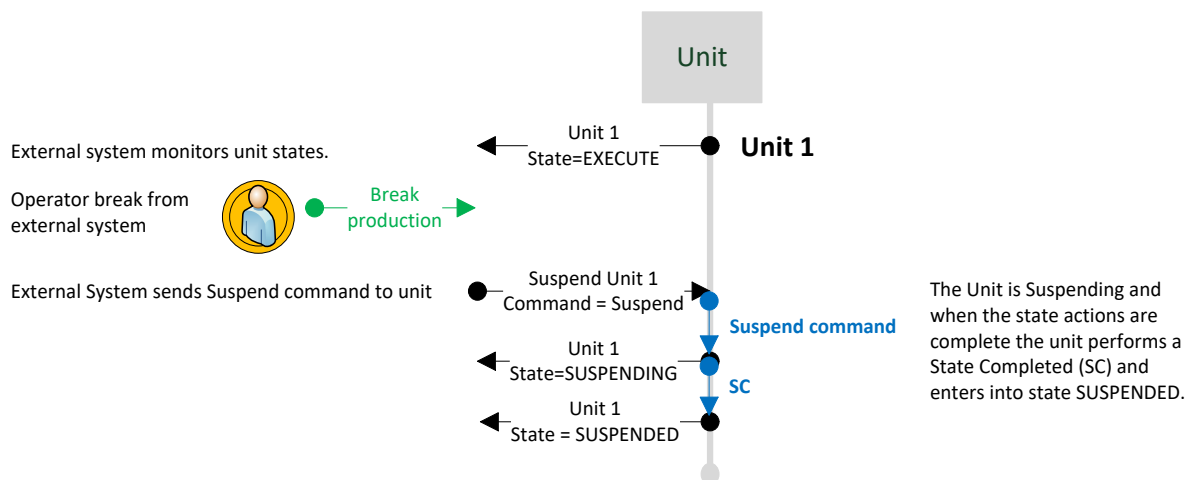


Figure 46: Operator break from External System

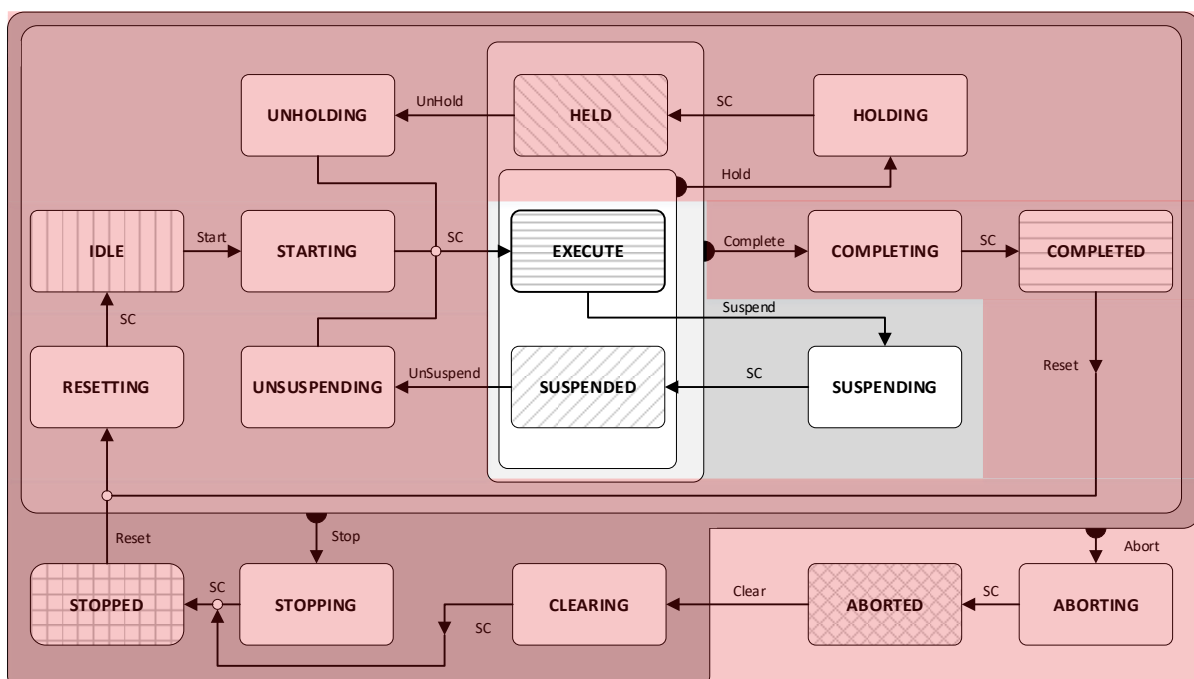


Figure 47: The states in the PackML state model in focus for Operator break from External System³⁵

12.7 RESUME UNIT/MACHINE AFTER BREAK FROM EXTERNAL SYSTEM

The operator restarts (unsuspend) the unit/machine. The External system unsuspends the unit by sending an Unsuspend command. The unit/machine will enter the EXECUTE state.

³⁵ The unit/machine Supplier dependent Machine State mode is able to be Stopped.

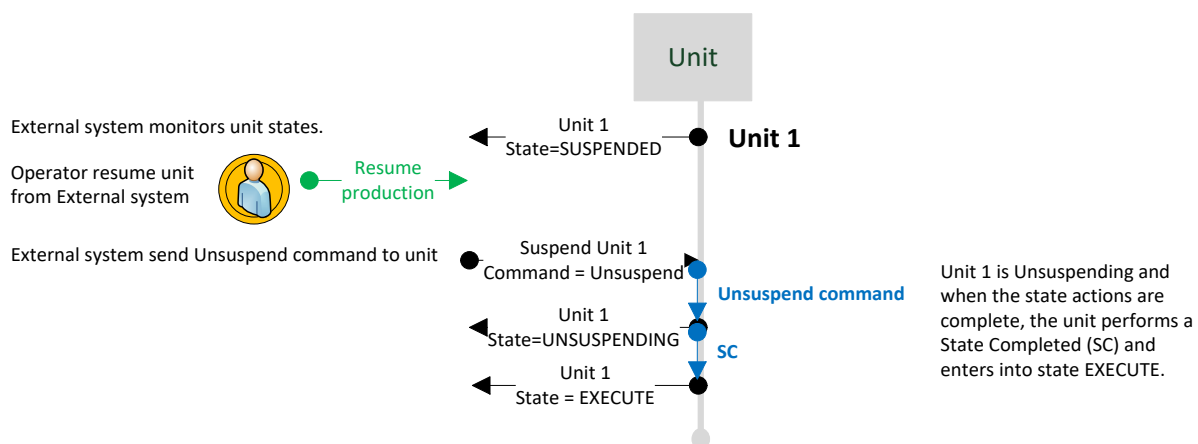


Figure 48: Resume unit/machine after break

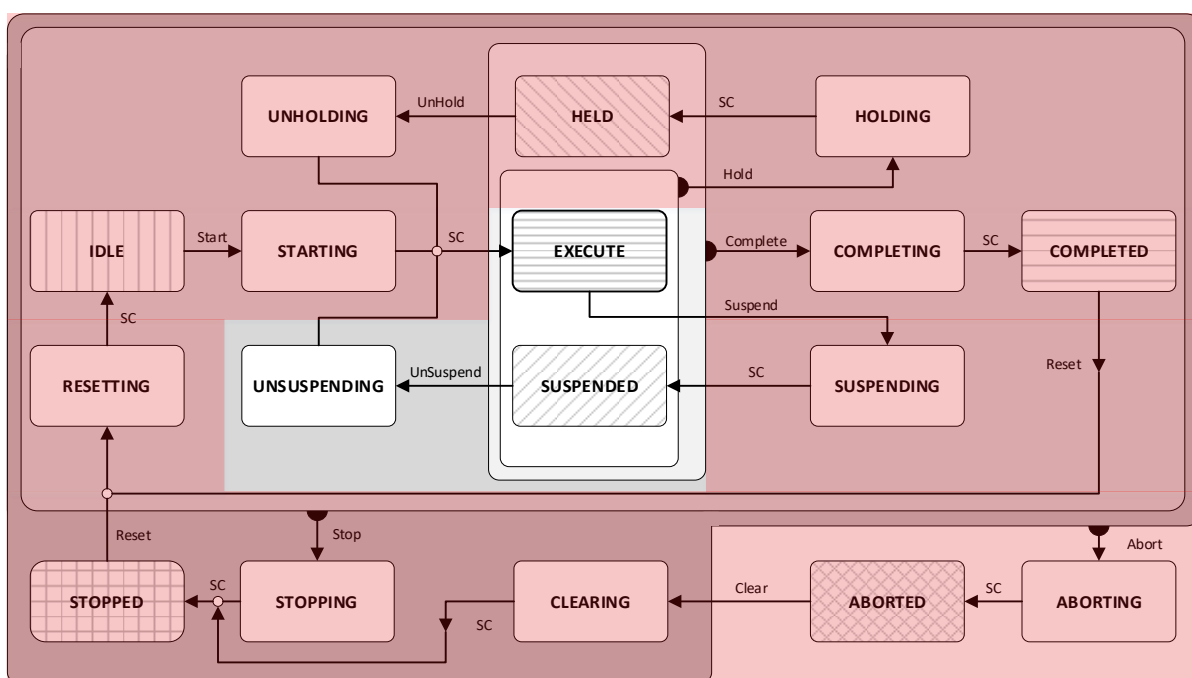


Figure 49: The states in the PackML state model in focus for resume unit/machine after break

12.8 STOP UNIT FROM UNIT PANEL/HMI

The operator stops the unit/machine by pressing Stop on the panel/HMI on the unit/machine. When the operator want to start the unit, the operator has to reset the unit, and make the unit ready for production. The illustration below illustrates that an error condition has required that the operator stops the unit. Normally, when an error occurs on the unit, the unit will enter the HELD state. As an example, this could happen in a situation, where the unit is running without the correct packaging material.

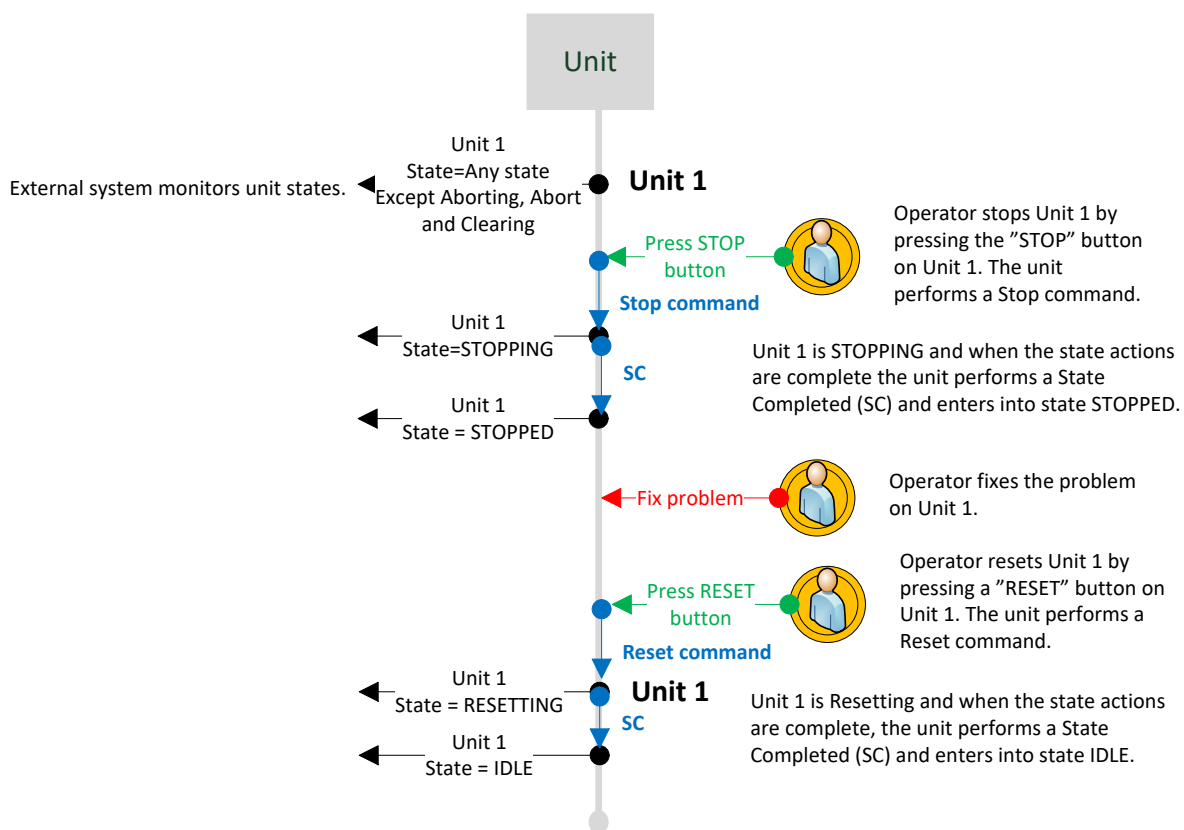


Figure 50: Stop unit from unit

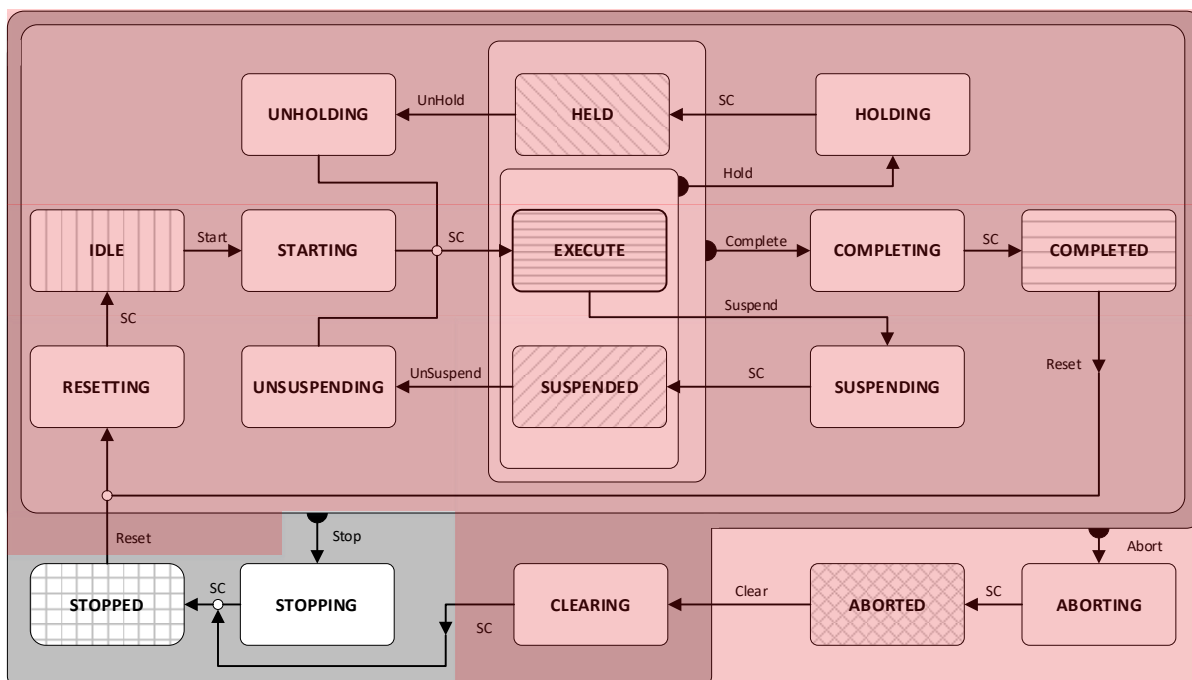


Figure 51: The states in the PackML state model in focus for Stop Unit

12.9 ABORT UNIT/MACHINE

The operator press E-stop on a unit/machine. The operator fixes what may be wrong on the unit/machine. It is not recommended to use E-stop to make a stop on a unit because the E-stop is not a controlled stop, as is done in stopping or holding. When the E-stop issue is corrected the operator clears the safety circuit, and resets the unit/machine, and finally makes the unit ready for execute production.

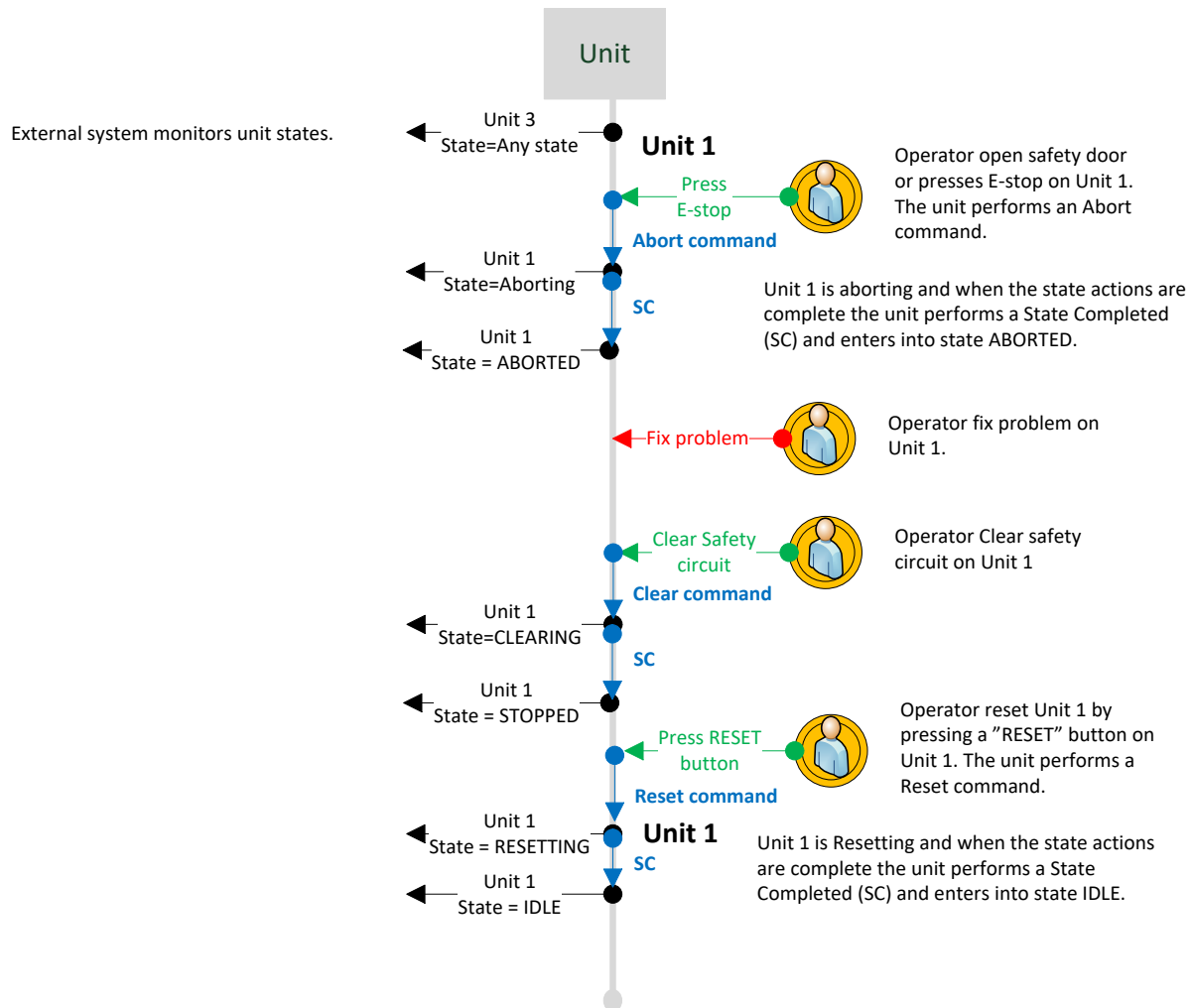


Figure 52: Abort Unit

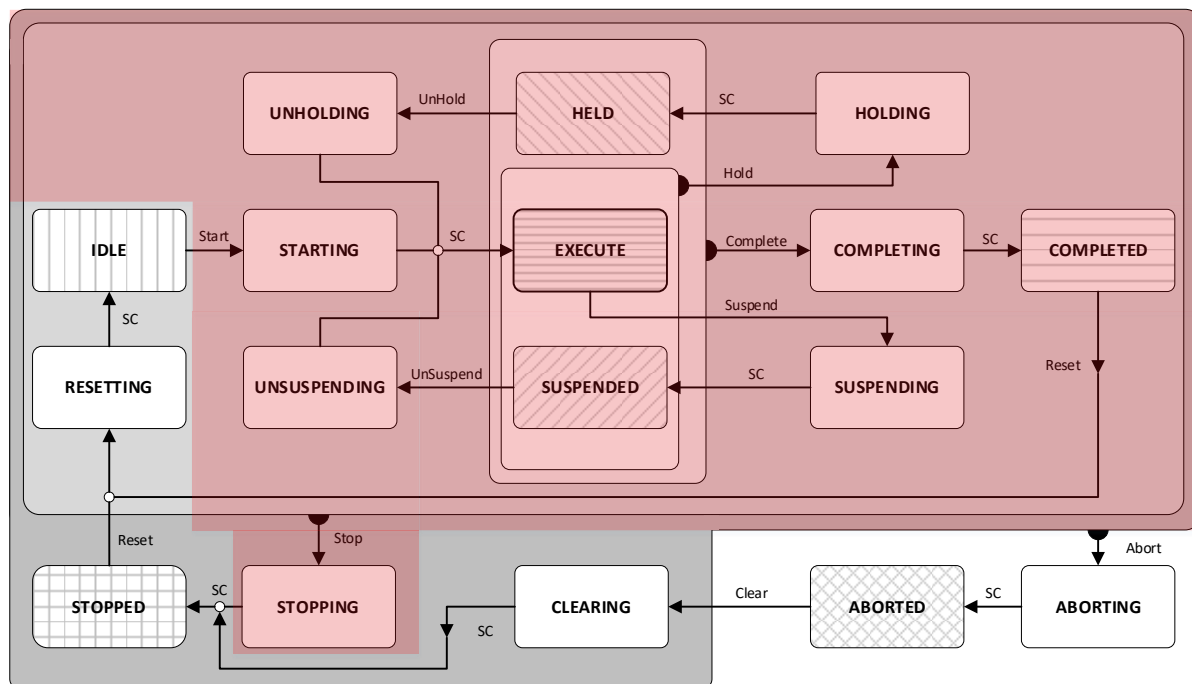


Figure 53: The states in the PackML state model in focus for Aborting and resetting Production

12.10 RE-START UNIT FROM UNIT PANEL/HMI AFTER STOP OR ABORT

It is possible to restart the unit from the unit panel/HMI.

Below is illustrated the situation when the operator starts the unit from the unit panel/HMI. The operator presses start on the unit. The unit restores parameters and it will enter the EXECUTE state.

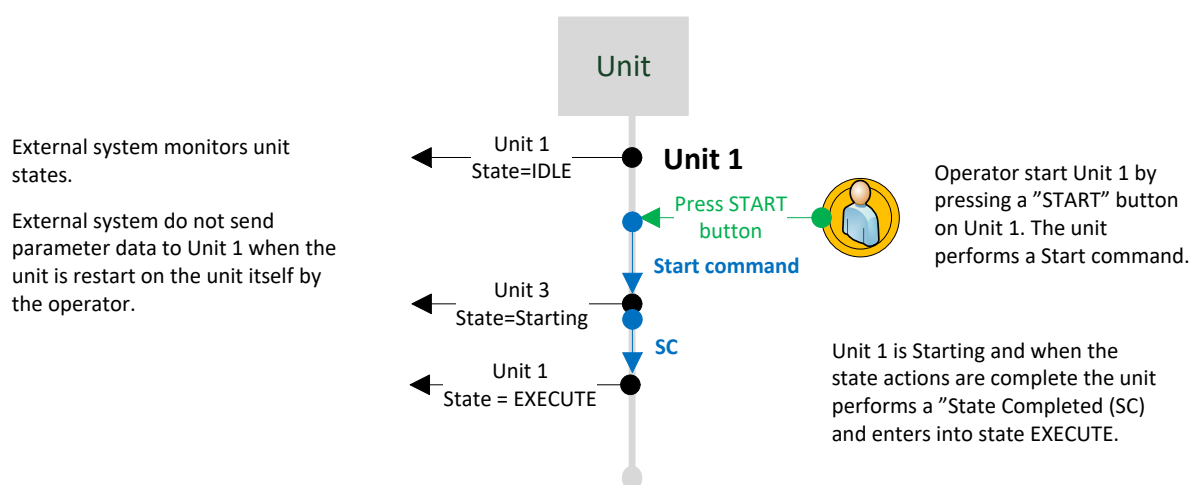


Figure 54 Re-start unit from unit after stop or abort

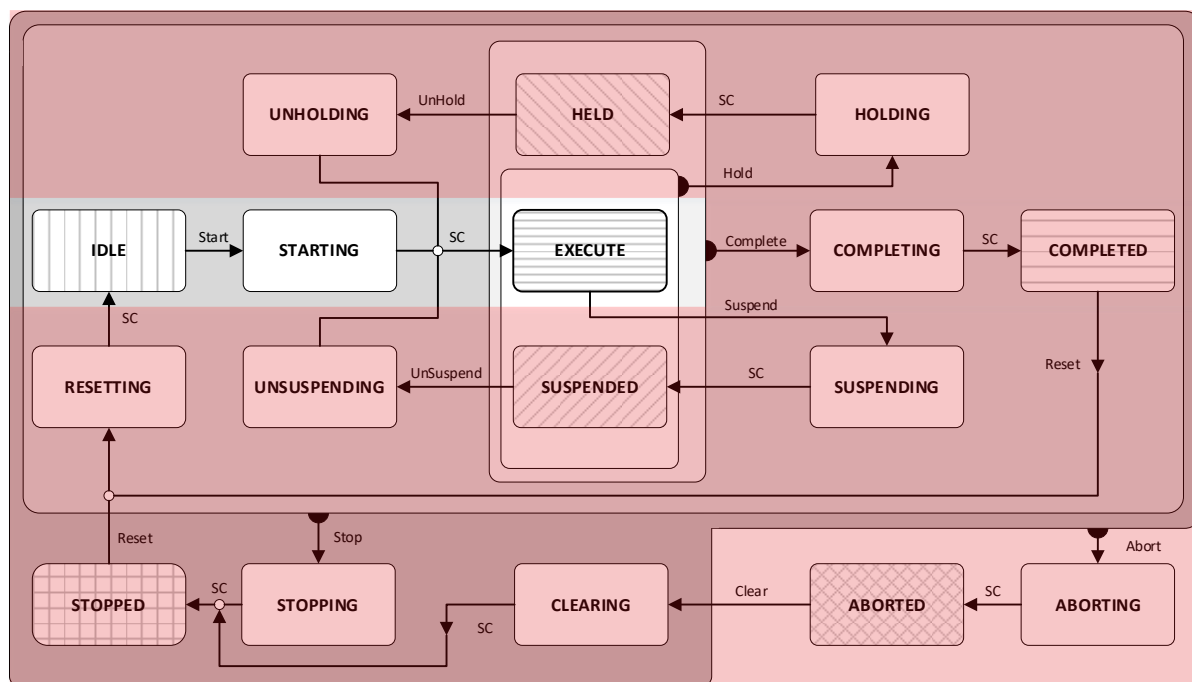


Figure 55: The states in the PackML state model in focus for Re-start unit from unit panel/HMI.

12.11 RE-START UNIT FROM EXTERNAL SYSTEM AFTER STOP OR ABORT

It is also possible to restart the unit from the External System.

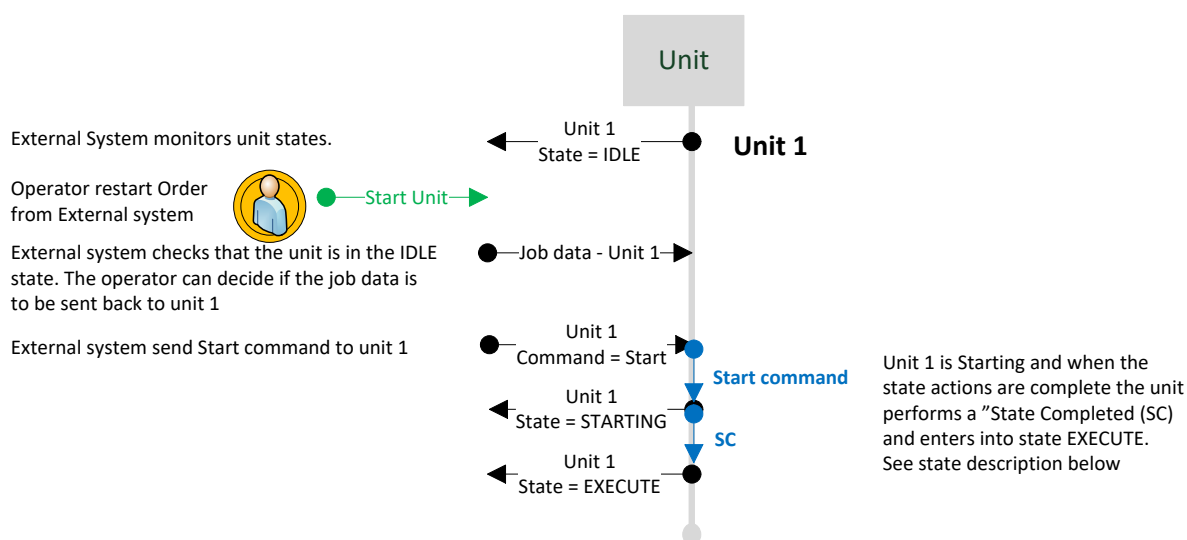


Figure 56: Re-start Unit after stop or abort

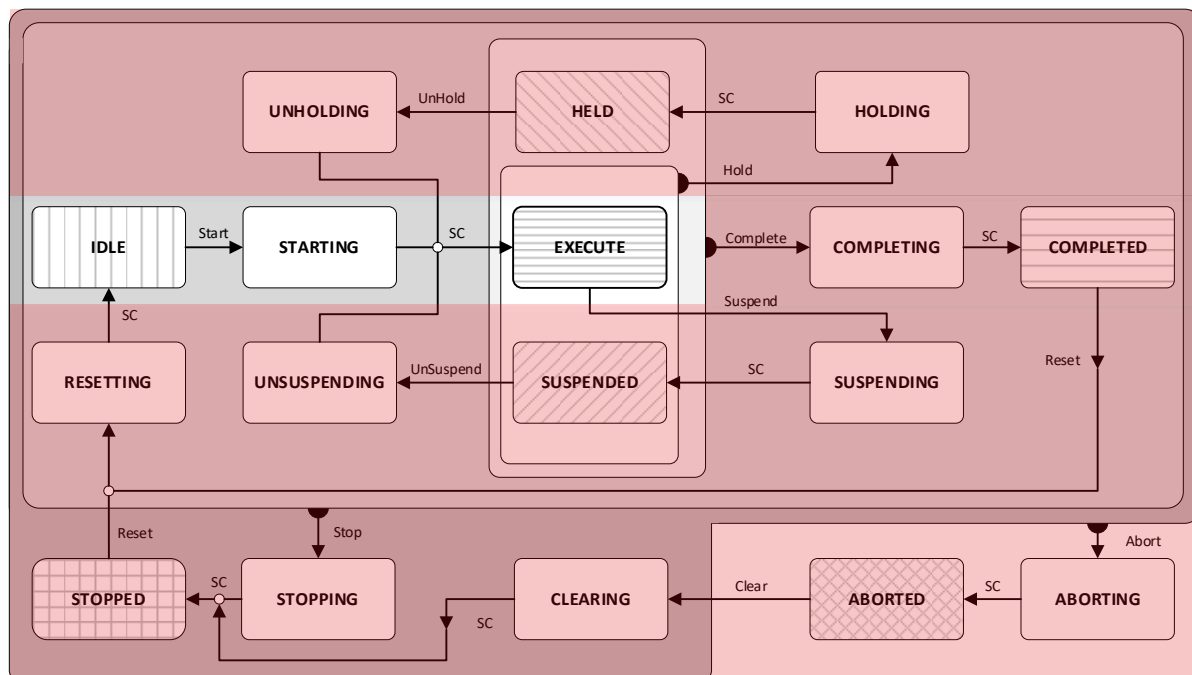


Figure 57: The states in the PackML state model in focus for re-start Unit from external supervisory system.

12.12 ALARM AND WARNING ON UNIT

The data flow for an alarm is shown on Figure 60. All alarm and warning data sent from the unit must be defined and described, if an external system needs to use the alarm and warning data for HMI or Historian purpose.

The operator can be directed to a unit/machine to a situation, were an alarm or warning is activated. Depending on the situation the unit/machine has to perform certain actions. The actions to be taken can be defined in an event state management table. See more in section 12.12.1. The reason for the event state management table, as shown in Table 25, is the possibility to handle different interpretations of alarms and warnings between different End Users. End Users may not all agree on what is an alarm and a warning, therefore a table gives the flexibility to Machine Suppliers to map to current End Users requirements.

For example the End Users definitions could be as follows:

Internal Event:

- Unit/machine alarm e.g. drive failure
- Unit/Machine warnings e.g. Missing packaging material

Configure internal Events in categories:

- Safety related Event: Alarm
- Non Safety related events: Alarms and Warnings (technical)
- Material related Event: Alarm and Warning

12.12.1 PackML alarm and event state management table

It is possible to create a table that allows Machine Suppliers to configure which types of alarms and events will lead to a state change. It can be used to document when a machine should switch to the states in the PackML state model. It is especially useful when the unit has to jump to HELD, STOPPED, SUSPENDED and ABORTED, because the End Users sometimes have different interpretations and expectations of when the different states should be entered.

Table 25: Example Alarm/Event state management table

Alarm / Event	State change ID = Commands	Reaction
1	8	Abort
2	8	Abort
3	4	Held – Error on material
4	4	Held – Error on glue
5	3	Stop – Drive failure
6	3	Stop – Sensor failure
7	6	Suspend - Blocked
8	6	Suspend – Queue
9	20	Pop-up information on HMI – Executing state
10	20	Pop-up information on HMI – Executing state
.....

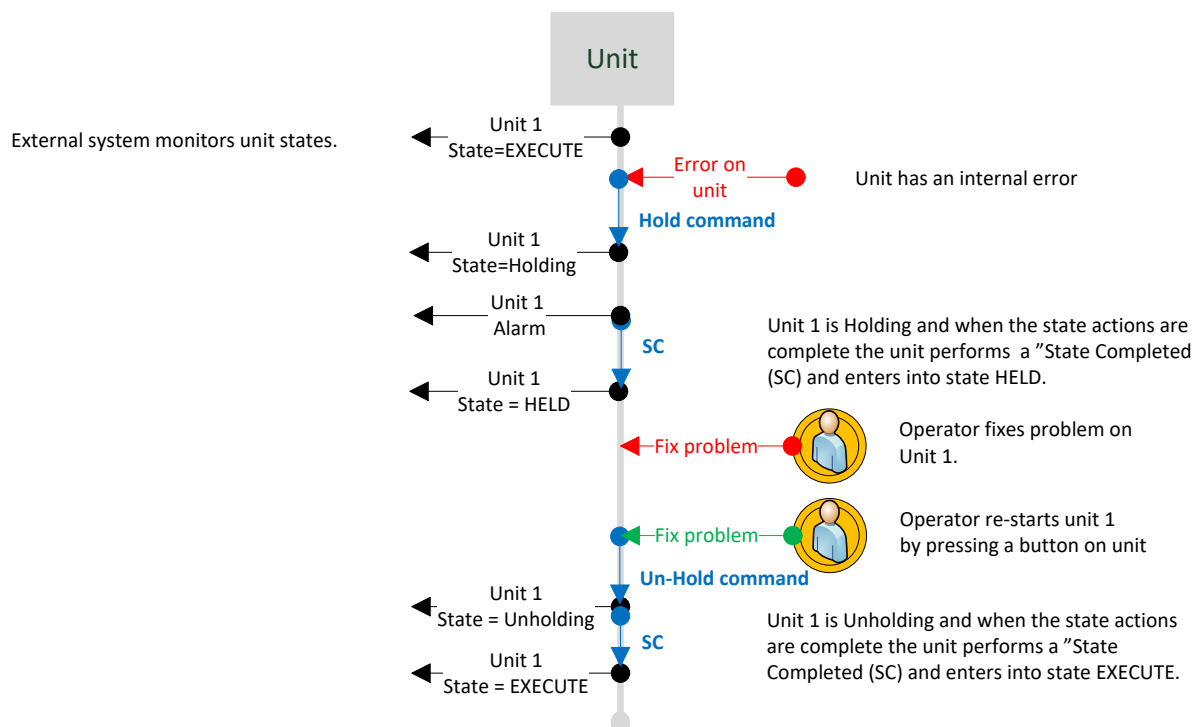


Figure 58: Alarm on unit

The data flow for warnings is shown below.

Hint: All warning data sent from the unit must be defined if the external system needs to use the alarm data.

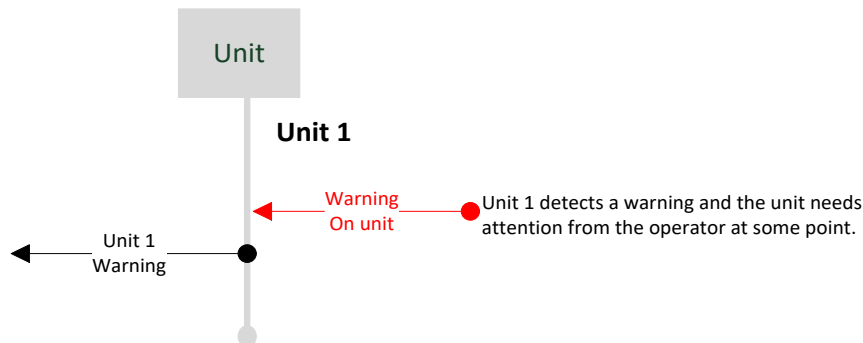


Figure 59: Warning on unit

Note: Alarms and warnings can be sent in any PackML state, therefore all PackML states are in focus.

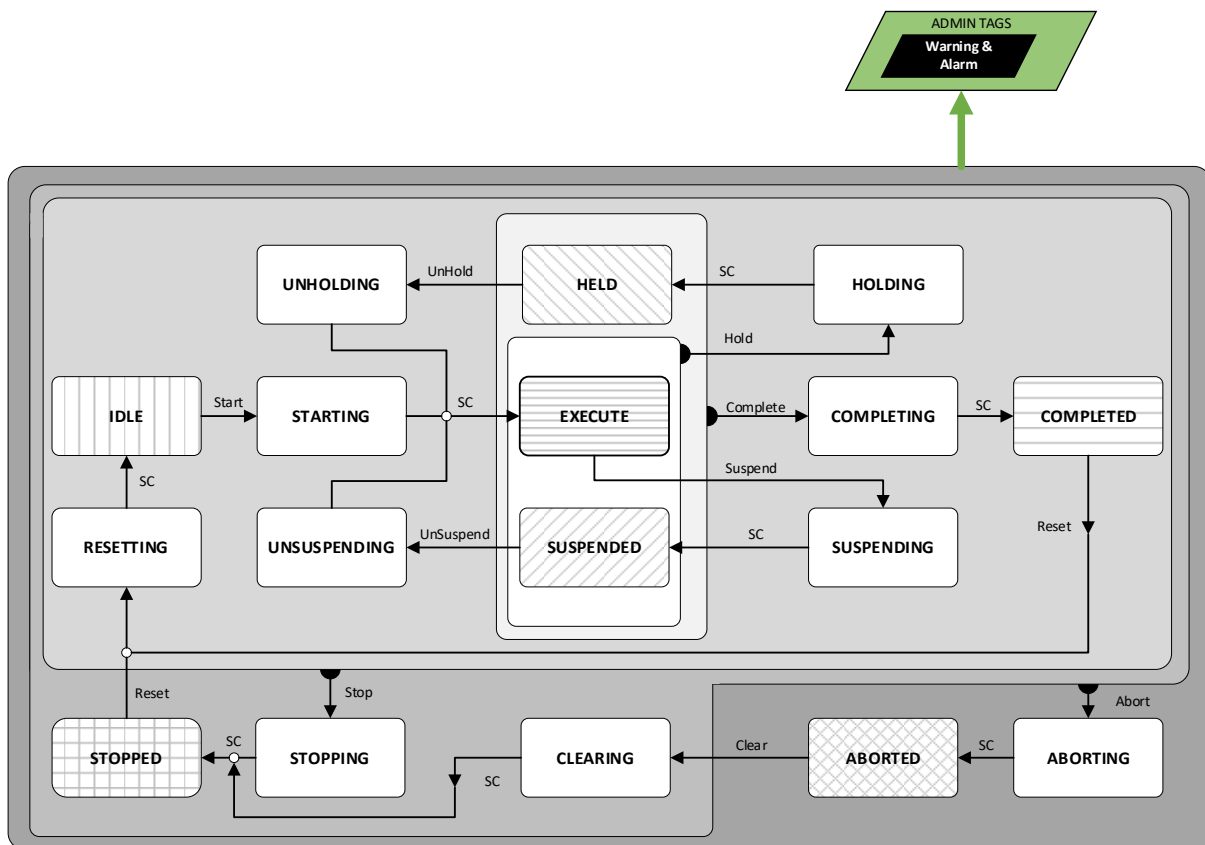


Figure 60: The states in the PackML state model in focus for alarm and warning handling

12.13 BACKGROUND TASK: OEE

This example illustrates the data flow of raw data for OEE calculations. The unit generates the data, which is made available to the external system using the PackTags: StopReason.Value, ProdDefectiveCount, ProdProcessedCount.

In this situation the unit/machine only sends OEE parameters as PackTags and all calculation of OEE KPIs are performed by an external system for OEE analysis.

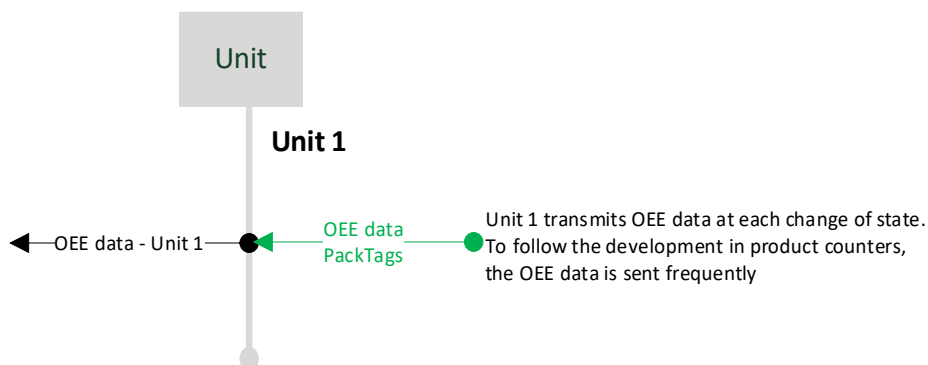


Figure 61: Background task – OEE

Note: OEE data can be sent in any PackML state, therefore all PackML states are in focus.

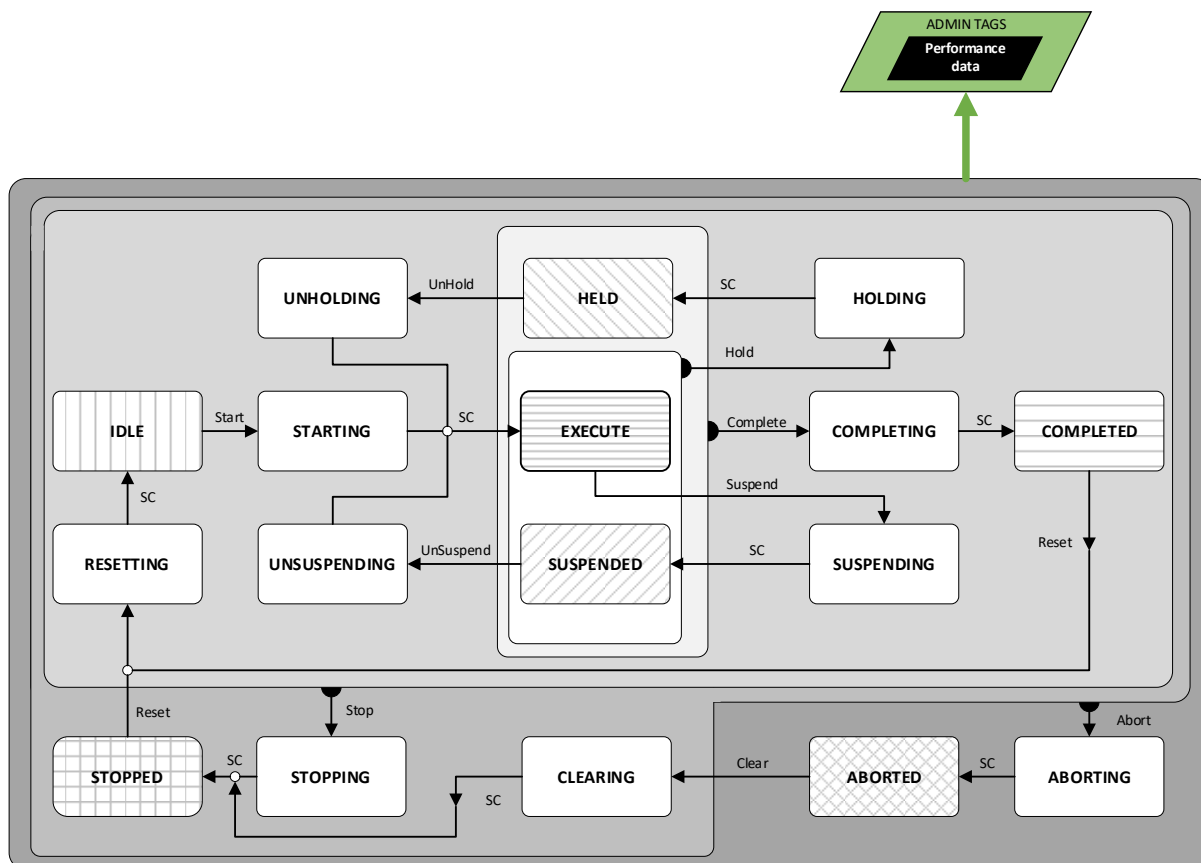


Figure 62: The states in the PackML state model in focus for Background Task: OEE

12.14 BACKGROUND TASK: MODE & STATE

In this example the PackML state and mode is available from a unit. The external system can collect the state and mode from the unit and can use it to generate an overview of the unit or an entire line of units.

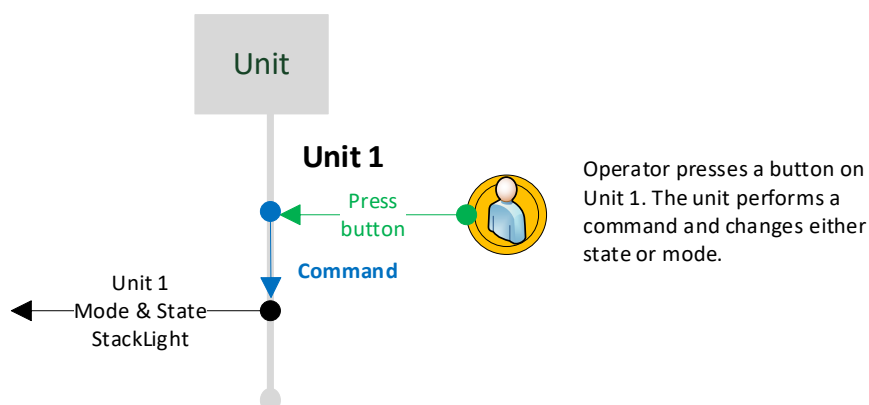


Figure 63: Change mode or state, and the unit will send Mode & State information

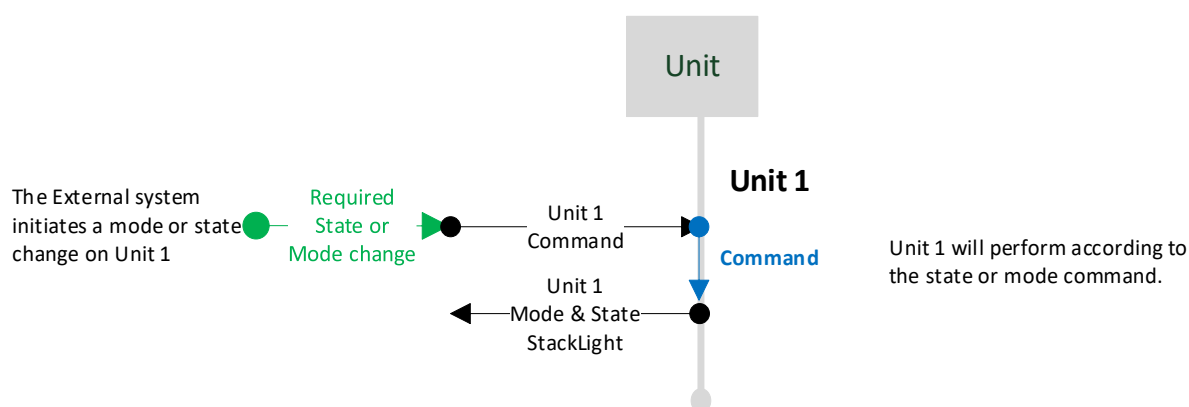


Figure 64: Send Mode & State information

Note: State and Mode can be send in any PackML state, therefore all PackML states are in focus.

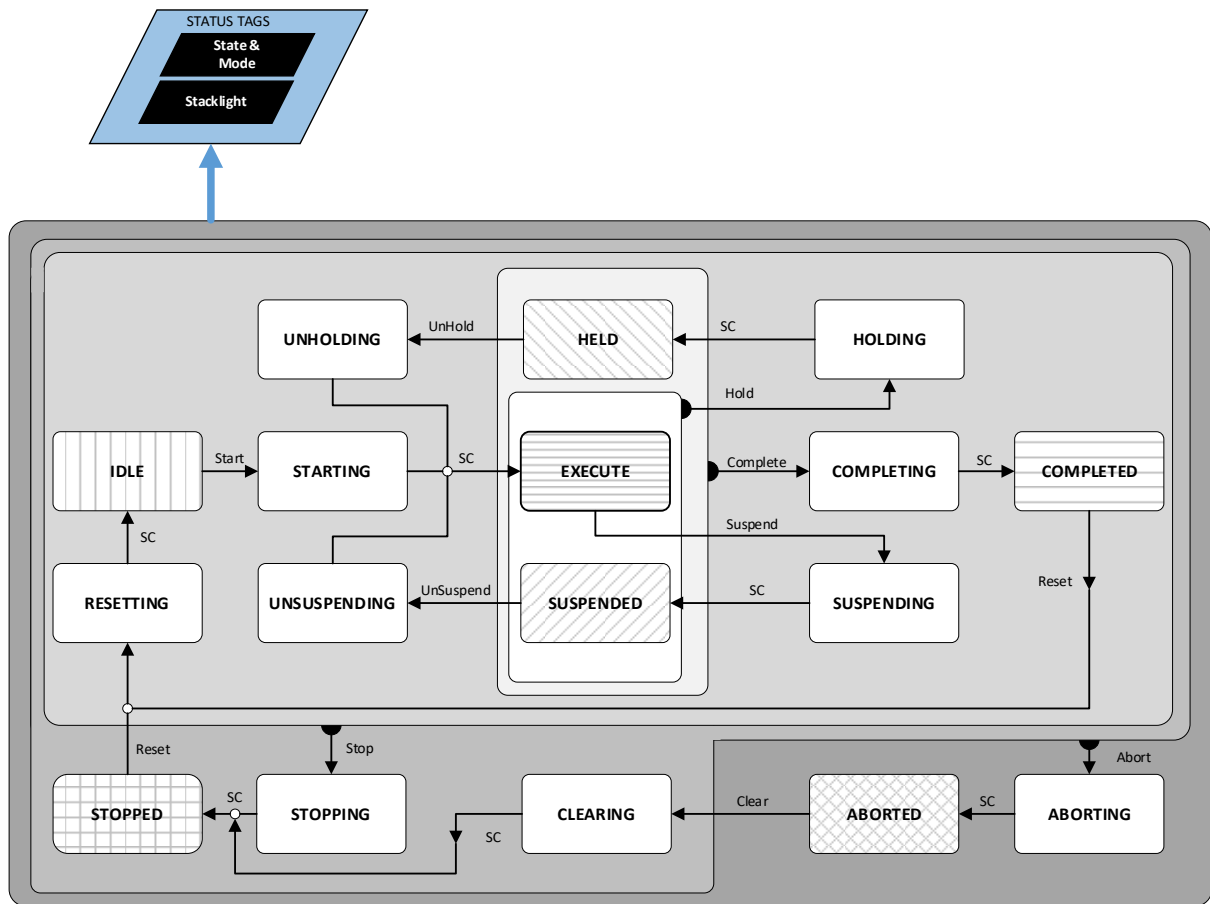


Figure 65: The states in the PackML state model in focus for mode & state reporting

12.15 SUSPEND: STARVATION OR SATURATION

In this example, a product queue guard on a unit detects starvation or saturation (blockages, queue) and the unit enters the SUSPENDED state.

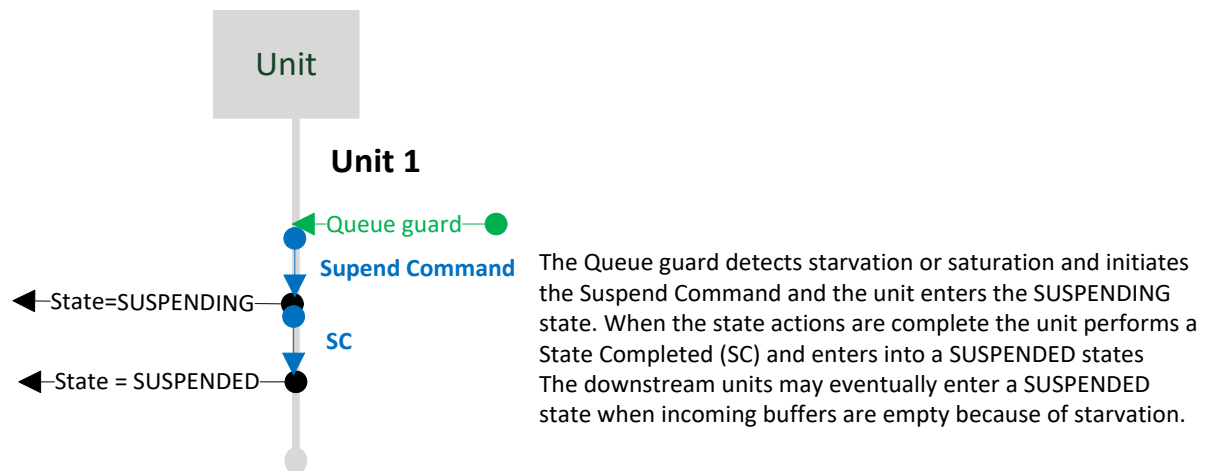


Figure 66: Suspend Production because of starvation or saturation

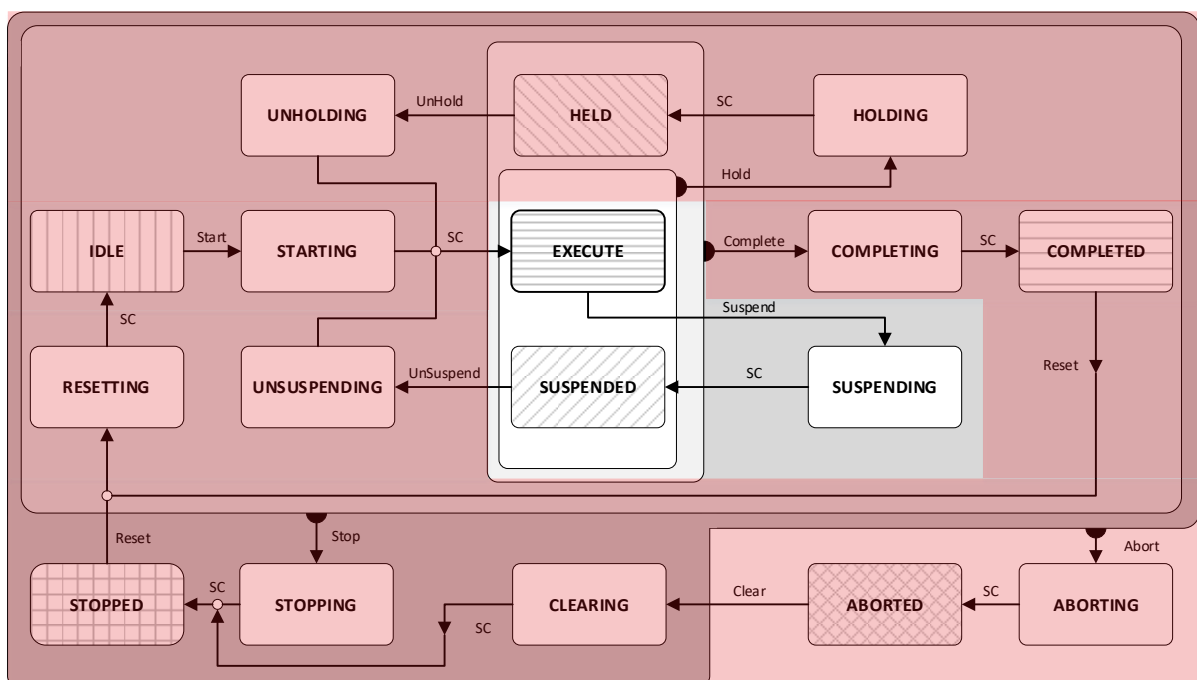


Figure 67: The states in the PackML state model in focus for Suspending Production

12.16 UNSUSPEND

When the queue guard on a unit no longer detects starvation or saturation (blockages, queue) it unsuspends to return to the EXECUTE state.

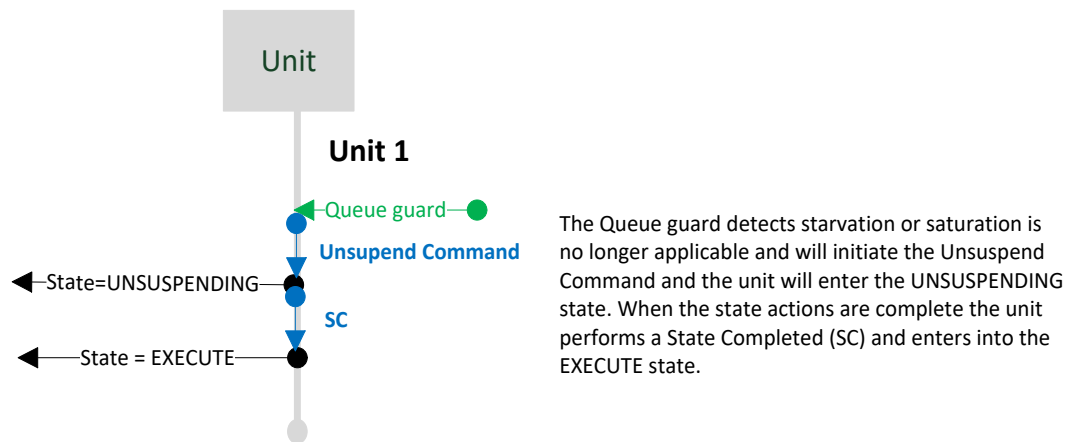


Figure 68: Re-start Production after suspension

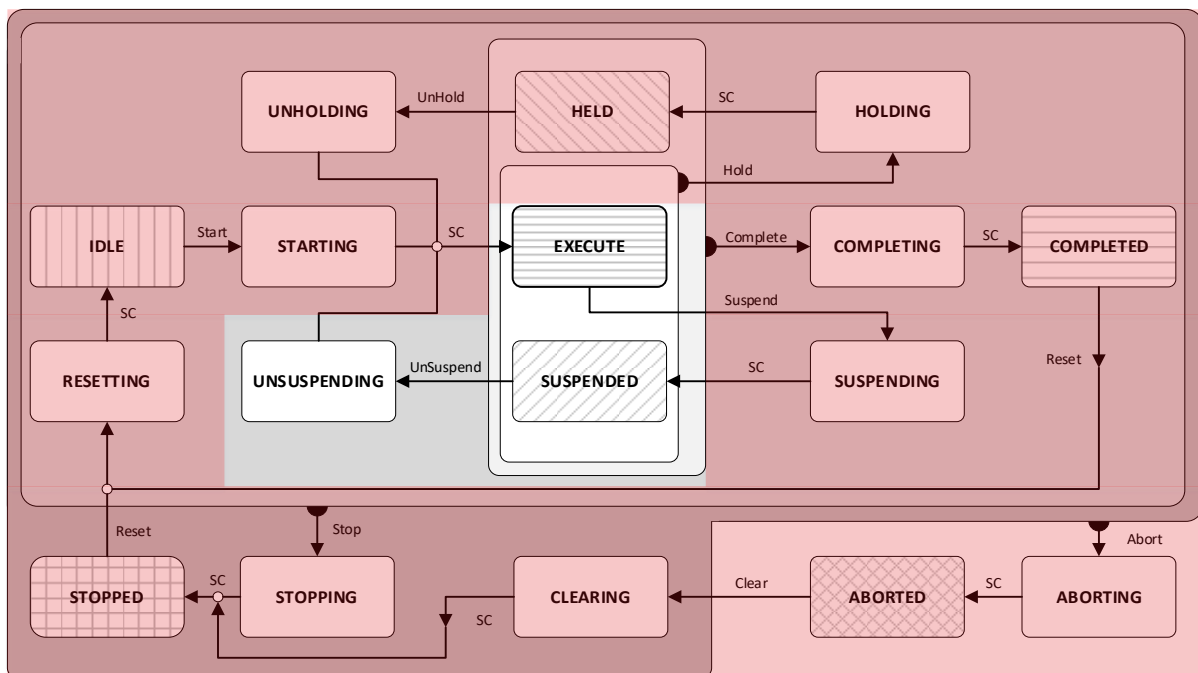


Figure 69: The states in the PackML state model in focus for Unsuspending production.

13. REFERENCE INFORMATION

13.1 DEFINITIONS AND ABBREVIATIONS

Table 26 List of definitions and abbreviations

Abbreviation	Description	Explanation
FAT	Factory Acceptance Testing	Test of the unit/machine being carried out before delivery. The test is carried out to ensure that delivery is in accordance with stated requirements.
FMEA	Failure Mode and Effect Analysis	A FMEA is often the first step of a system reliability study. It involves reviewing of as many components, assemblies, and subsystems as possible to identify failure modes, and their causes and effects.
HMI	Human Machine Interface	Human machine interface is the part of the unit/machine that handles the human to machine interactions.
I/O test	Input / Output test	I/O test is a test of electrical connection at place of installation.
ISA	The International Society of Automation	ISA is a non-profit technical society for engineers, technicians, businesspeople, educators and students, who work, study or are interested in industrial automation and pursuits related to it.
OEE	Overall Equipment Effectiveness	OEE measurement is commonly used as a key performance indicator (KPI) in conjunction with manufacturing efforts to provide an indicator of success.
OEM	Original Equipment Manufacturer	OEM are manufacturing products or components bought by another company and sold under the purchasing company's brand or company name.
OMAC	Organization for Machine Automation and Control	OMAC is the organization for automation and manufacturing professionals that is dedicated to supporting the machine automation and operation needs of manufacturing.
PackML	Packaging Machine Language	The primary objective of PackML is to bring a common "look and feel" and operational consistency to all machines.
PackML GW	PackML Gateway	A gateway that represents the PackML Interface State Manager and maps to the existing Machine State Manager of a specific machine (unit/machine).
PackML SDS	PackML Software Design Specification	Function specification for a unit/machine, that specifies the behaviour of a unit/machine and specifies the Interface to the unit/machine from a PackML prospective.
PackTags	PackML data identifier	PackTags are data elements that have been defined by the OMAC Packaging Workgroup to facilitate easy data exchange between packaging machines.
PLC	Programmable Logic Controller	A computer controlling instruments, motors, valves, etc.
SAT	Site Acceptance Testing	Test of the unit/machine being carried out on the site where it is installed. The test is carried out to ensure that delivery is in accordance with stated requirements.
SCS	Supervisory Control System	Manufacturing Execution System (MES)

Abbreviation	Description	Explanation
		<p>System software to integrate customer orders in the ERP system and to program sequences for SCADA and PLC systems.</p> <p>Supervisory Control and Data Acquisition (SCADA) The SCADA system normally log data from units/machines and monitors the unit status. The SCADA is often a HMI for a set of units/machines put together in a Production Line.</p> <p>Line Controller (LC) Line Controller can be part of a Supervisory Control System. A Line Controller controls a number of Units in a line.</p> <p>The Supervisory Control System could be a PLC or part of existing PLC, PC or Server.</p>
TR 88.00.02	Technical Report	Technical Report is an informative document on S88 implementation on discrete machines – It is the specification of PackML.

13.2 REFERENCES

Table 27 List of references

Ref. no.	Document
[S88-1]	ANSI/ISA-88.00.01-2010, Batch Control, Part 1: Models and Terminologies.
[TR88]	ANSI/ISA-TR88.00.02-2022, Machine and unit/machine States, An Implementation example of ANSI/ISA-88.00.01.
[S88-5]	ISA/DRAFT-88.00.05-2013, Working Draft 08, Batch Control, Part 5: Implementation Models & Terminologies for Modular Equipment Control.

13.3 LIST OF PEOPLE INVOLVED IN PREPARATION OF THE GUIDELINE

The following people served as active participants in preparation of this technical report:

Table 28 List of people involved in the preparation of this technical report

Name	Surname	Company
Joachim	Bergmeyer	A+F Automation
Engelbert	Freiberger	Alpma
Jürgen	Wegscheider	Alpma
Arne	Svendsen	Arla Foods
Claus	Norup	Arla Foods
Enrico	Paolucci	B&R
Lazarus	Patsakas	B&R
Miodrag	Veselic	B&R
Sari	Germanos	B&R
Rob	Rawlyk	Beckhoff
Pascal	Witprächter	Bosch
Wolfgang	Martin	Bosch
Carl	Bostrom	Bosch Rexroth
Thomas	Haberkorn	Bosch Rexroth
Imbriaco	Giacomo	Coesia Engineering Center
Peder	Stocklund	Coloplast
Arthur C.	Smith	Coming
Emilian	Axinia	COPA-DATA
Van Dam	Maarten	COPA-DATA
Pierre	Henri	Danone Engineering Worldwide
Pasquali	Stefano	Gebo Cermex.
Fabian	Elsässer	Harro Höfliger
Manfred	Zoehrer	Haas Food Equipment
Karsten	Vollmer	KHS
Andreas	Gschrey	Krones
Christian	Rott	Krones
Uwe	Keiter	Lenze
Christoph	Riedmann	Lti-motion
Frank	Polky	Mars
Lee	Smith	Mettler Toledo
Heinz-Hermann	Bruemmer	Meurer
Jochen	Schohaus	Meurer
Malte	Schlüter	Mitsubishi
Claus	Botzenhardt	Multivac
Fabrice	Bertin	Nestlé

Name	Surname	Company
Geoff	Gooding	Nestlé
Jean-Claude	Pourchet	Nestlé
Kris	Matlock	Nestlé
Markus	Gehrke	Nestlé
Peter	Jarlvik	Norden Machinery
Carlos	Ruiz	Omron
Josep	Lario	Omron
Adal	Tecleab	P&G
Jason	Debruler	P&G
Jim	Blundy	P&G
Paul	Southman	P&G
Jay	Joyner	P&G
Mark	Ruberg	Pro Mach
Gord	Davison	Pro Mach
Gunther	Saelzer	Rockwell Automation
Christian	Chatel	Schneider Electric
Rainer	Beudert	Schneider Electric
Anthony	Esnault	Serac
Carsten	Nøkleby	SESAM-World
Michel-René	Kübler	SEW-EURODRIVE
Heiko	Soehner	Siemens
Kim	Meyer-Jacobsen	Siemens
Mike	Pieper	Siemens
Uwe	Zell	SIG Combibloc
Thomas	Rummel	Softing
Filippo	Sarafini	Tetra Pak
Heinrich	Iben	Tetra Pak
Kasper Korsholm	Christiansen	Velux
Ole	Riis	Velux
David	Tobon	Yaskawa

13.4 SUPPORT

The preparation of this version2 of the document is supported by **Mitsubishi Electric** and **SESAM-World**.

13.5 THE AUTHOR

Carsten Nøkleby is a senior consultant with focus on Automation and Production IT. The consulting jobs are done for customers like Beiersdorf, Arla Foods, Tetra Pak, Novo Nordisk, Nestlé, Grundfos, Velux, Baader & Linco, Danish Hospitals and many more. The consulting is related to strategic and tactical issues related to the introduction computers and new business processes on the factory floor. The technical systems handled by Carsten Nøkleby are PLC, SCADA, DCS and MES.

Carsten Nøkleby is responsible for running the Industrial networks SESAM with approx. 3.000 members. The network has its focus on distributing knowledge between the members. The main focus in on Automatization and Digitalization of manufacturing processes.

Carsten Nøkleby made a Ph.D. Thesis in 1992 at the Technical University of Denmark. The title was “Structure and Information for a flexible assembly cell – Issues in the analysis, design and implementation of virtual manufacturing devices for remote controlled application processes”.

The visionary

I have always worked with the Smart Industry applications. From the old days we called it Computer Integrated Manufacturing during my Ph.D. What we see today is more emphasis on integration of people, machinery, equipment and products. The internet is here and IoE is the future. The potential of combining existing and new technology with business processes and culture will make the change in the manufacturing environment.

14. VERSION HISTORY

Table 29 Version log

Date [YYYY-MM-DD]	Initials	Description	Version
2015-12-10	CAN	First version – Input from OMAC workshops in 2015	01A
2016-03-11	CAN	Second version – Document rewritten based on the definition of terminology and scope defined by the OMAC working group <ul style="list-style-type: none"> Claus Norup, Arla Foods Arne Svendsen, Arla Foods and OMAC Kris Matlock (Nestlé) Pascal Witprächtiger, Bosch Packaging Uwe Keiter, Lenze and OMAC Malte Schlüter, Mitsubishi Electric 	02A
2016-03-18	CAN	Sections added: <ul style="list-style-type: none"> Understand and define a Unit The PackML Interface State Model List of participants 	02B
2016-03-22	CAN	Review comments from Uwe Keiter and Claus Norup Section added: <ul style="list-style-type: none"> Section 6.1 The syntax of the PackML State Model Section 7 PackML Event State Manager Section 8 PackML Control Command definitions Section 9 PackML Interface State definitions 	02C
2016-04-08	CAN	<ul style="list-style-type: none"> unit/machine control modes PackTags Overview of use case scenarios 	02D
2016-04-21	CAN	<ul style="list-style-type: none"> Final version for review of whole reference group 	02E
2016-06-23	CAN	<ul style="list-style-type: none"> All 262 received review comments are put into the document. All figures are made neutral and the HMI part is presented in a separate section. All the comments on the workshop 21st and 22nd June at Arla Foods Copenhagen. 	03A 03B 03K
2016-08-01	CAN	<ul style="list-style-type: none"> The document has been review by Sofie Ambrosius, Arla Foods. 	03L
2016-08-03	CAN	<ul style="list-style-type: none"> Minor details changed, e.g., participation list and mapping alarms to event triggers to PackML 	03M
2016-08-04	CAN	<ul style="list-style-type: none"> Review and minor details are changed. 	03N
2016-08-05	CAN	<ul style="list-style-type: none"> Final version for 2nd review 	03O 03P
2016-10-13	CAN	<ul style="list-style-type: none"> Review comments are included. The comments are from the companies: Alpma, Arla Foods, BoschRexRoth, COPA-DATA, Corning, Krones, Mettler-Toledo, Nestlé, SEW-Eurodrive and Siemens. 	03Q ... 03Y
2016-11-02	CAN/DB	Linguistic review by Dennis Brandl, BR&L Consulting	1.00
2021-08-20	CAN	Updated version according to the updated ISA TR88.00.02-2021. Major changes to state model & PackTags.	1.01
2021-12-08	CAN	Reviews of document is included	2.00
2022-09-02	CAN	Update according to final approved ISA TR88.00.02-2021	2.01
2022-10-12	CAN	Updated according to the final agreed year for release of ISA TR88.00.02-2022.	2.02
2022-10-19	CAN	Updated in relation to STRING definition. Change STRING[5] to STRING[6].	2.03

