Standards
Certification
Education & Training
Publishing
Conferences & Exhibits

Setting the Standard for Automation™

ISA TECHNICAL REPORT

ISA-TR88.00.02-2022

Machine and Unit States: An implementation example of ISA-88.00.01

This ISA Technical Report was approved on 6 September 2022

ISA-TR88.00.02-2022

Machine and Unit States: An implementation example of ISA-88.00.01

ISBN: 978-1-64331-224-8

ISA
3252 S. Miami Blvd., Suite 102
Durham, North Carolina 27703 US

Email: standards@isa.org

PREFACE

This preface, as well as all footnotes, are included for information purposes only and are not part of ISA-TR88.00.02-2022.

This technical report has been prepared as part of the service of ISA, the International Society of Automation, toward a goal of uniformity in the field of instrumentation. To be of real value, this document should not be static but should be subject to periodic review. Toward this end, the Society welcomes all comments and criticisms and asks that they be addressed to the Secretary, Standards and Practices Board, email: standards@isa.org.

This ISA Standards and Practices Department is aware of the growing need for attention to the metric system of units in general, and the International System of Units (SI) in particular, in the preparation of instrumentation standards, recommended practices, and technical reports. The Department is further aware of the benefits of USA users of ISA standards of incorporating suitable references to the SI (and the metric system) in their business and professional dealings with other countries. Toward this end, the Department will endeavor to introduce SI and acceptable metric units in all new and revised standards to the greatest extent possible. The Metric Practice Guide, which has been published by the Institute of Electrical and Electronics Engineers (IEEE) as ANSI/IEEE Std. 268-1992, and future revisions, will be the reference guide for definitions, symbols, abbreviations, and conversion factors.

It is the policy of ISA to encourage and welcome the participation of all concerned individuals and interests in the development of ISA standards. Participation in the ISA standards-making process by an individual in no way constitutes endorsement by the employer of that individual, of ISA, or of any of the standards, recommended practices, and technical reports that ISA develops.

CAUTION — ISA ADHERES TO THE POLICY OF THE AMERICAN NATIONAL STANDARDS INSTITUTE WITH REGARD TO PATENTS. IF ISA IS INFORMED OF AN EXISTING PATENT THAT IS REQUIRED FOR USE OF THE DOCUMENT, IT WILL REQUIRE THE OWNER OF THE PATENT TO EITHER GRANT A ROYALTY-FREE LICENSE FOR USE OF THE PATENT BY USERS COMPLYING WITH THE DOCUMENT OR A LICENSE ON REASONABLE TERMS AND CONDITIONS THAT ARE FREE FROM UNFAIR DISCRIMINATION.

EVEN IF ISA IS UNAWARE OF ANY PATENT COVERING THIS DOCUMENT, THE USER IS CAUTIONED THAT IMPLEMENTATION OF THE DOCUMENT MAY REQUIRE USE OF TECHNIQUES, PROCESSES, OR MATERIALS COVERED BY PATENT RIGHTS. ISA TAKES NO POSITION ON THE EXISTENCE OR VALIDITY OF ANY PATENT RIGHTS THAT MAY BE INVOLVED IN IMPLEMENTING THE DOCUMENT. ISA IS NOT RESPONSIBLE FOR IDENTIFYING ALL PATENTS THAT MAY REQUIRE A LICENSE BEFORE IMPLEMENTATION OF THE DOCUMENT OR FOR INVESTIGATING THE VALIDITY OR SCOPE OF ANY PATENTS BROUGHT TO ITS ATTENTION. THE USER SHOULD CAREFULLY INVESTIGATE RELEVANT PATENTS BEFORE USING THE DOCUMENT FOR THE USER'S INTENDED APPLICATION.

HOWEVER, ISA ASKS THAT ANYONE REVIEWING THIS DOCUMENT WHO IS AWARE OF ANY PATENTS THAT MAY IMPACT IMPLEMENTATION OF THE DOCUMENT NOTIFY THE ISA STANDARDS AND PRACTICES DEPARTMENT OF THE PATENT AND ITS OWNER.

ADDITIONALLY, THE USE OF THIS DOCUMENT MAY INVOLVE HAZARDOUS MATERIALS, OPERATIONS OR EQUIPMENT. THE DOCUMENT CANNOT ANTICIPATE ALL POSSIBLE APPLICATIONS OR ADDRESS ALL POSSIBLE SAFETY ISSUES ASSOCIATED WITH USE IN HAZARDOUS CONDITIONS. THE USER OF THIS DOCUMENT MUST EXERCISE SOUND PROFESSIONAL JUDGMENT CONCERNING ITS USE AND APPLICABILITY UNDER THE USER'S PARTICULAR CIRCUMSTANCES. THE USER MUST ALSO CONSIDER THE

APPLICABILITY OF ANY GOVERNMENTAL REGULATORY LIMITATIONS AND ESTABLISHED SAFETY AND HEALTH PRACTICES BEFORE IMPLEMENTING THIS DOCUMENT.

THE USER OF THIS DOCUMENT SHOULD BE AWARE THAT THIS DOCUMENT MAY BE IMPACTED BY ELECTRONIC SECURITY ISSUES. THE COMMITTEE HAS NOT YET ADDRESSED THE POTENTIAL ISSUES IN THIS VERSION.

ISA (www.isa.org) is an international professional association that sets the standard for those who apply engineering and technology to improve the management, safety, and cybersecurity of modern automation and control systems used across industry and critical infrastructure. Founded in 1945, ISA develops widely used international standards for industrial automation, safety, and cybersecurity; certifies industry professionals; provides education and training; publishes books and technical articles; hosts conferences and events; and provides networking and career development programs for its members and customers around the world.

ISA owns Automation.com, a leading online publisher of automation-related content. In addition, through a wholly owned subsidiary, ISA bridges the gap between standards and their implementation with the ISA Security Compliance Institute (www.isasecure.org) and the ISA Wireless Compliance Institute (www.isa100wci.org).

The following people served as active members in the 2022 revision of this document:

NAME	AFFILIATION
Uwe Keiter	A+F Automation & Foerdertechnik
Mads Olsen	B&R Automation
Frank Apolito	Beckhoff
Rob Rawlyk	Beckhoff
Carl Bostrom	Bosch Rexroth
Paul Redwood	Church & Dwight
Arthur Smith	Corning
Jeffrey Sponenberg	Corning
Adem Kulauzovic	Domino
Bill Anagnostopoulas	Emerson
Chris Mills	E-tech Group
Michael Rauscher	Harro-Höfliger
Mark Will	ID Technology
Marc Wolf	Matrix Packaging
Adam Griffen	Mettler-Toledo
John Uber	Mettler-Toledo
Lee Smith	Mettler-Toledo
Patrick Toohey	Mettler-Toledo
Tom Dorward	Mettler-Toledo
Malte Schlüter	Mitsubishi Electric
Ron MacDonald	Nestlé
Ken Truluck	Pepsico
Bryan Griffen	PMMI
Jason DeBruler	Procter & Gamble
Brian McMullen	Rockwell Automation
Bruce Kane	Rockwell Automation
Dan Aardsma	Rockwell Automation
Dan Seger	Rockwell Automation
Joachim Thomsen	Rockwell Automation
John Dart	Rockwell Automation
Michel-Rene Kübler	SEW Eurodrive
Rick Simer	SEW Eurodrive

Kevin Bartuska	Siemens
Mike Pieper	Siemens
Christophe Nophut	Weihenstephan Group
Christophe Le Gallic	Yaskawa
Doug Meyer	Yaskawa
Imran Mohamed	Yaskawa

The following people served as active members in the 2015 revision of this document:

NAME	AFFILIATION
David Bell	ATR Distributing Co (Wonderware)
Jerry Golden	B&R Automation
John Kowal	B&R Automation
Marc Wolf	B&R Automation
Ryan Stell	B&R Automation
Carl Bostrom	Bosch Rexroth
Dave Chappell	CMAa
Arthur Smith	Corning
James Hoban	Eaton
Kevin Pitts	Eaton
Doug Buschor	E-Technologies
John Spengler	GE
Niels Andersen	Invensys
Kent Knudsen	Krones
Brian Leboeuf	Krones
Uwe Keiter	Lenze
Lee Smith	Mettler Toledo
Steven Abramowski	Miller Coors
Jerry Yen	Mitsubishi Electric
David Kaley	Mitsubishi Electric
Vili Taukolo	Mitsubishi Electric
Fabrice Bertin	Nestlé
Tom Doney	Nestlé
Bryan Griffen	Nestlé
Toshiya Kitagawa	Omron
Tatsuru Hyodo	Omron
Rick Schoonover	Patti Engineering
Jeff Russell	PepsiCo
Murugan Govindasamy	Pfizer
Eelco van der Wal	PLCopen
Tom Egan	PMMI
Mark Ruberg	Pro Mach
Christopher Thomas	Pro Mach (Axon)
Gord Davison	Pro Mach (Edson)
Jason DeBruler	Procter & Gamble
Dan Amundson	Procter & Gamble
John Dart	Rockwell Automation
Dan Seger	Rockwell Automation
Yann Renard	Sidel
Stephane Hacpille	Sidel
Mike Pieper	Siemens Industry
Rana, Ajay S	Siemens Industry
Alexander Stukenkemper	Siemens Industry
Oliver Welle	Siemens AG
Dewayne A Bruschi	SMC

R Johnson	SMC
Doug Meyer	Yaskawa

The following people served as active members in the initial creation of this document:

NAME	AFFILIATION
Joseph Jablonski	Acumence Inc
Barry Kluener	Alexander & Associates
David Bell	ATR Distribution (Wonderware)
Thomas Hopfgartner	B&R Automation
Uwe Keiter	B&R Automation
Gerd Hoppe	Beckhoff
David Arens	Bosch Rexroth
Dennis Brandl	BR&L Consulting
David Chappell	CMAa-LLC
Joe Faust	Douglas Machine Company
Dominik Gludowatz	Elau
Tom Jensen	Elau
Andre Uhl	Elau
David Bauman	ISA / OMAC
Alex Pereira	KHS
Randy Dwiggins	Maverick Technologies
Ron MacDonald	Nestlé
Eelco VanDerWal	PLCopen
Mike Lamping	Procter & Gamble
Ulrich Arlt	Rockwell Automation
Darren Elliott	Rockwell Automation
Brian Hedges	Rockwell Automation
John Dart	Rockwell Automation
Paul Nowicki	Rockwell Automation
Dan Seger	Rockwell Automation
Garth Basson	SAB Miller
Larry Trunek	SAB Miller
Willie Lotz	SAB Miller Brewing Co.
Fabian Ochoa M.	SAB Miller Brewing Co.
Mario Broucke	Siemens AG, A&D
Roland Heymann	Siemens AG, A&D
Detlef Rausch	Siemens AG, A&D
Robert Freller	Siemens AG, F&B
Dr. Holger Grzonka	Siemens Energy & Automation
Mike Pieper	Siemens Energy & Automation
Mark DeCramer	WAGO
Dr. Tobias Voigt	Weihenstephan University

This document was approved for publication by the ISA Standards and Practices Board on 6 September 2022.

NAME	AFFILIATION
D. Zetterberg, Vice President	Consultant
D. Bartusiak	Collaborative Systems Integration, LLC
D. Brandl	BR&L Consulting, Inc.
P. Brett	Honeywell Inc.
D. Dunn	WS Nelson
J. Federlein	Consultant
S. Fluchs	admeritia GmbH
J.-P. Hauet	Hauet.com
D. Lee	Emerson Process Management
G. Lehmann	AECOM
T. McAvinew	Consultant
V. Mezzano	Fluor Corp.
C. Monchinski	Automated Control Concepts Inc.
G. Nasby	Canadian National Railway Company
D. Reed	Rockwell Automation
N. Sands	DuPont
H. Sasajima	FieldComm Asia LLC
I. Verhappen	Willowglen Systems Inc.
D. Visnich	Burns & McDonnell
W. Weidman	Consultant
J. Weiss	Applied Control Solutions LLC
M. Wilkins	Yokogawa UK Ltd.

This page intentionally left blank.

CONTENTS

Figures

Tables

This page intentionally left blank.

FOREWORD

The ISA88 committee has defined a series of standards addressing the batch industry and providing terminology and a consistent set of concepts and models for batch manufacturing plants and batch control. These standards, however, were not defined in the context of packaging machines, or machines that perform discrete operations. As the ISA88 batch standard continues to evolve, the context of the standard models may be extended to include the entire plant, integrating the software definitions of batch, packaging, converting, and warehousing. Currently, as noted in this report, there is a need to begin consideration of the ISA88 standard in the context of differing automated machinery.

This is an informative document. This document contains implementation guidelines in order to establish a common presentation and high-level software architecture or layout. The terms and definitions used in this document are harmonized, as much as possible, with ISA-88.00.01; the document is not definitive in this respect. The models used, and applied, in this document are an extension of the models presented in ISA-88.00.01 and are shown how they are applied to differing machine functionality. Discrete machine functionality is expressed graphically in several situations and described. The intent of this document is proposing specific implementation options and indicating a preference for a specific set of machine types.

In 2015 this document was updated for three reasons: to simplify the document and enable easier adoption, to clarify existing materials and make them easier to apply, and to make it more complete, including the best approaches being used to implement. Major changes included the addition of a minimum set of PackTags, a minimum set of states, and the removal of examples and MES definitions not central to the document's purpose. Other changes addressed in this revision were: improved definition of the SUSPENDING and HOLDING states, the concept of allowable transition states between modes, the renaming of Producing Mode to Production Mode, the addition of Blocked and Starved PackTags, stop reasons, and warnings.

In 2022 this document was further updated to clarify and improve the usability of the specification based on field feedback and usage reports. Major changes include a revised state model diagram, the removal of the "Remote" interface, the addition of PackTags that describe the machine configuration to a supervisory system, new examples for mode definitions, and several other updates to the existing PackTag structures for functional improvement and clarity.

This document is not an American National Standard and the material contained herein is not normative in nature. Comments on the content of this document should be sent to ISA, email: standards@isa.org.

Abstract

The approaches used in programming discrete machines today may vary widely and often become solely dependent on the individual machine and the software engineer, or control systems programmer. Such variability offers little additional value and generally increases the total costs, from the designing and building of the process to operating and maintaining the system by the end user. This document on the implementation of ISA-88.00.01 in discrete machines demonstrates how to apply the ISA-88.00.01 standard concepts to automated machine states and modes. This document gives examples of general and specific machine state models and procedural methods. The document cites real control examples as implementations and provides specific tag naming conventions; it also cites a number of common terms that are consistent with batch processing and ISA-88.00.01.

Key words

state machine, state model, mode manager, machine state, unit control mode, PackML, state commands, command tags, status tags, administration tags, base state model, functional programming, modular programming, machine control software, discrete machine software, PackTags, Weihenstephan, Production Data Acquisition, PDA, ISA88, TR88.

Introduction

When the ISA-88.00.01 standard is applied to applications across a plant, there is a need to align the terminologies, models, and key definitions between different process types: continuous, batch, and discrete processes. Discrete processes involve machines found in the packaging, converting, and material handling applications. The operation of these machines is typically defined by the original equipment manufacturer (OEM), system integrator, end user, or is industry specific.

A task group with members from technology providers, OEMs, system integrators, and end users was chartered by the OMAC (Organization for Machine Automation and Control) Packaging Workgroup. The task group generated the PackML guidelines as a method to show how the ISA-88.00.01 concepts could be extended into packaging machinery. This document is intended to build upon and formalize the concepts of the PackML guidelines and to show application examples.

The purpose of the document is to:

- define a standard state-based model for automated machines,
- identify definitions for common terminology,
- explain to practitioners how to use state programming for automated machines, and
- identify a common tag structure for automated machines in order to:
  - provide for “connect & pack” functionality
  - provide functional interoperability and a consistent look and feel across the plant floor.
  - provide consistent tag structure for connection to plant MES and enterprise systems.

This page intentionally left blank.

1 Scope

Since its inception, the OMAC Packaging Machine Language (PackML) group has been using a variety of information sources and technical documents to define a common approach, or machine language, for automated machines. The primary goals are to encourage a common “look and feel” across a plant floor, and to enable and encourage industry innovation. The PackML group is recognized globally and consists of control vendors, OEMs, system integrators, universities, and end users, which collaborate on definitions that endeavor to be consistent with the ISA88 standards and consistent with the technology and the changing needs of a majority of automated machinery. The term “machine” used in this report is equivalent to an ISA88 “unit.”

This has led to the following:

a) a definition of machine/unit state types,

b) a definition of machine/unit control modes,

c) a definition of unit control mode management,

d) state models, state descriptions, and mode and state transitions, and

e) a definition of the minimum PackTags required for performance monitoring.

2 References

The following documents contain provisions that are referenced in this text. At the time of publication, the editions indicated were valid. All documents are subject to revision, and parties to agreements based on this technical report are encouraged to investigate the possibility of applying the most recent editions of the reference documents indicated below in addition to seeking out other relevant standards that apply to their equipment or installation.

ISA-88.00.01-2010, *Batch Control – Part 1: Models and Terminologies*

ISA-88.00.02-2001, *Batch Control – Part 2: Data Structures and Guidelines for Languages*

ISA-88.00.03-2003, *Batch Control – Part 3: General and Site Recipe Models and Representation*

ISA-88.00.04-2006, *Batch Control – Part 4: Batch Production Records*

IEC 61131-1, *Programmable controllers – Part 1: General Information*

IEC 61131-3, *Programmable controllers – Part 3: Programming Languages*

IEC TR61131-4, *Programmable controllers – Part 4: User Guidelines*

PLCopen TC5 Safety Software

Weihenstephan Standard – WS Pack Version 08, http://www.weihenstephaner-standards.de/index.php?id=2&L=1

ISA-95.00.01-2010 (IEC 62264-1 Mod), *Enterprise – Control System Integration – Part 1: Models and Terminologies*

ISA-95.00.02-2010 (IEC 62264-2 Mod), *Enterprise – Control System Integration – Part 2: Object Model Attributes*

ISA-95.00.05-2013, *Enterprise – Control System Integration – Part 5: Business-to-Manufacturing Transactions*

DIN 8782, *Beverage Packaging Technology; Terminology Associated with Filling Plants and their Constituent Machines*

www.omac.org – Organization for Machine Automation and Control Website

ISA-TR88.00.02-2008 – The original edition of this technical report.

ISA-TR88.00.02-2015 – The first revision of this technical report.

ISO 22400, *Automation Systems and Integration – Key performance indicators (KPIs) for manufacturing operations management*

ISO/IEC9899 (C99), *Programming languages — C*

OPC 30050: *OPC UA PackML Companion Specification*

3 Overview

3.1 Introduction

Automated machine programming is typically done by software engineers, machine designers, and system integrators. The form and style of the machine software ranges from modular, to monolithic in nature. The objective of this report is to specify the application of a common software methodology that employs reusable objects to program automated machinery based on the ISA-88.00.01 models for modular equipment control. The naming of specific software components, or operational aspects, is dependent on the needs of the automated machine. This report should be interpreted in a general sense to encompass all automated machinery. It is focused on the overall operation and functionality of automated machines. This document enables a consistent method of machine interconnection and operability. The diagrams and examples shown in the report are specific in terms of the functionality they provide but can be implemented in various ways to fit most automated machinery and machine controllers; therefore, the figures do not follow ISO/IEC 19501:2005 for UML depiction of software flow.

If automated machinery is modelled in an ISA-88.00.01 physical hierarchy, the example mapping shown in Figure 1 is possible. The example in this document will assume that a machine can represent the unit level in the ISA88 hierarchy.

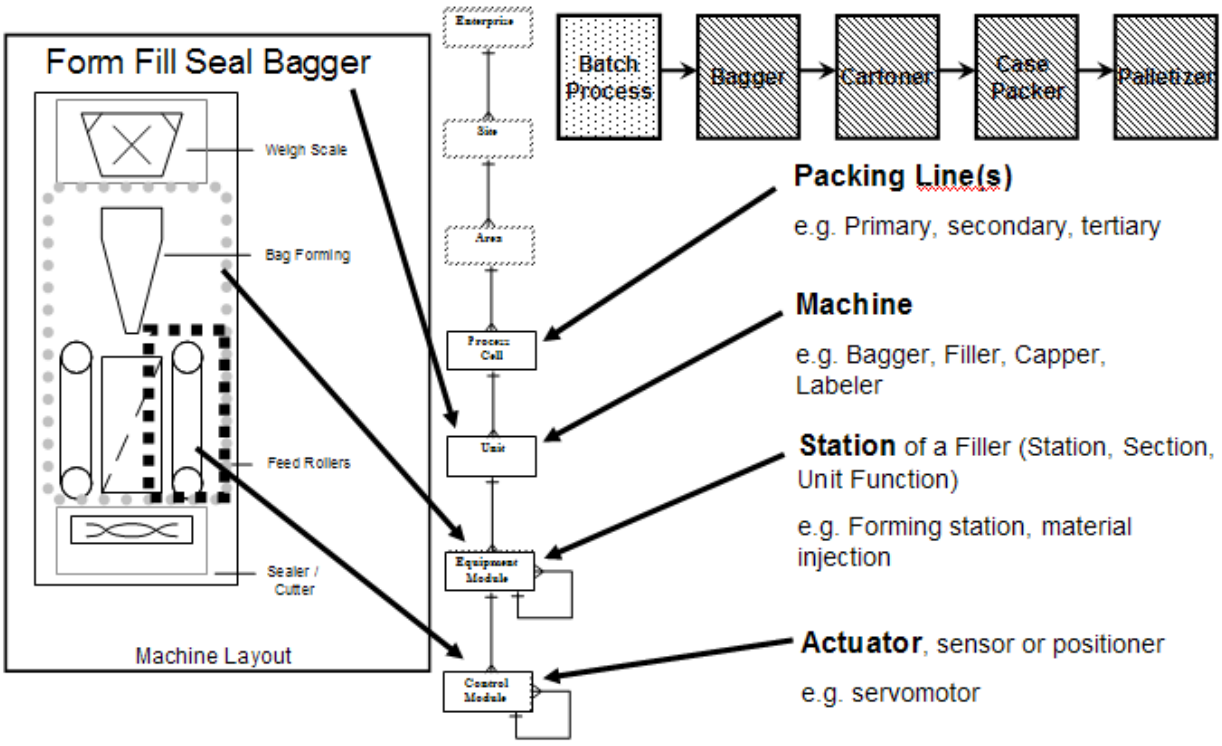

Figure 1 – Automated machines applied to ISA-88.00.01 physical model

Furthermore, the objective of this document is to provide a definition of machine (unit) modes and states, as well as state models corresponding to the different machine (unit) modes. In this example application of ANIS/ISA-88.00.01, the use of the state model and unit modes are extensible, but the methods governing the way in which the modes and states are used is not. This technical report demonstrates the flexibility and ease in which this method can be implemented in terms of ISA88, as well as how it provides the "common look and feel" desired in automated machines. This model only constrains the standard names and semantics for commonly used high level machine states as per a base state model.

The ISA-88.00.01 standard describes example modes and states as applied to equipment entities and procedural elements. This report identifies unit/machine modes and states, which should be considered an extension of the examples in the ISA-88.00.01 standard in order to meet the needs of automated machine processing.

3.2 Personnel and environmental protection

The personnel and environmental protection control activity provides safety for people and the environment. No control activity should intervene between personnel and environmental protection and the field hardware it is designed to operate with. Personnel and environmental protection is, by definition, separate from the higher-level control activities in this document. It may map to more than one software level of the equipment as desired.

A complete discussion of personnel and environmental protection, the classification of these types of systems, and the segregation of levels of interlocks within these systems is a topic of its own and beyond the scope of this document.

3.3 Control system compatibility

The minimum PackTag requirements that are identified in this document can be implemented on a majority of PLC systems used by packaging machine manufacturers. It can also be

implemented on systems such as check weighers, date coders, robots, and other ancillary devices that may utilize proprietary control platforms. Even though the PackTags naming convention is using logical names, it can be implemented using register memory systems.

4 Unit/machine states

4.1 Definition

A unit/machine state completely defines the current condition of a machine. A machine state is determined by an ordered procedure, or programming routine, that can consist of one or more commands to other procedural elements[1] or equipment entities, or be affected by the status of a procedural element[1] or equipment entity, or both. In performing the function specified by the state, the machine software will issue a set of commands to the machine procedural elements[1] or equipment entities, which in turn can report status.

Only one major processing activity may be active in one machine at any time[2]. The linear sequence of major activities will drive a strictly sequentially ordered flow of control from one state to the next state—no more than one state of the base model is allowed to be active in one machine at the same time.

Note: At a lower level, the minor sub-activities (or control procedures) that are combined to form a major activity at the machine operation level, may indeed be taking place in parallel as well as in sequence as defined in ISA-88.00.01 for equipment phases.

4.2 Types of states

For the purposes of understanding, two machine state types are defined:

- **Acting state:** A state that represents some processing activity. 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. In ISA-88.00.01 these are referred to as transient states, generally those states ending in “ING”.
- **Wait state:** A state used to identify that a machine has achieved a defined set of conditions. In such a state, the machine is maintaining a status until transitioning to an acting state. In ISA-88.00.01 this was referred to as a “final” or “quiescent” state.

4.3 Defined states

There are a fixed number of states defined in the base state model. This report establishes an example enumerated set of possible unit/machine states and corresponding examples of transitions between those states. As listed in Table 1, this set of states is similar in principle to the ISA-88.00.01-2010 example states and has a more direct correlation to the more expansive procedural state model provided in ISA-88.00.01-2010 Annex D. See Figure 2 for a visualization of the states and transitions within the complete TR88.00.02-2022 state model. Additional details on transitions are provided in 4.5.

[1] Term “procedural element” defined (ISA-88.00.01)

[2] A “major processing activity” corresponds to the term “equipment operation” as defined in ISA-88.00.01.

Table 1 – Example states for automated machines

ISA-88.00.01-2010 Procedural States	ISA-88.00.01-2010 Annex D Procedural States	ISA-TR88.00.02-2022 Equipment States				
		Value	Unit/Machine States	Wait	Acting	Minimum Required
<not defined>	CLEARING	1	CLEARING		x	
STOPPED	STOPPED	2	STOPPED	x		x
<not defined>	STARTING	3	STARTING		x	
IDLE	IDLE	4	IDLE	x		x
HELD	SUSPENDED	5	SUSPENDED	x		
RUNNING	RUNNING	6	EXECUTE		x	x
STOPPING	STOPPING	7	STOPPING		x	
ABORTING	ABORTING	8	ABORTING		x	
ABORTED	ABORTED	9	ABORTED	x		x
HOLDING	HOLDING	10	HOLDING		x	
HELD	HELD	11	HELD	x		
RESTARTING	UNHOLDING	12	UNHOLDING		x	
HOLDING	SUSPENDING	13	SUSPENDING		x	
RESTARTING	UNSUSPENDING	14	UNSUSPENDING		x	
<not defined>	RESETTING	15	RESETTING		x	
<not defined>	COMPLETING	16	COMPLETING		x	
COMPLETE	COMPLETE	17	COMPLETED	x		
PAUSING	PAUSING		<not defined>			
PAUSED	PAUSED		<not defined>			

Formal definitions of these states are given in Table 2:

Table 2 – Description of TR88 machine states

State Name	Description
STOPPED	**State Type: Wait** The machine is powered and stationary after completing the STOPPING state. All communications with other systems are functioning (if applicable). A Reset command will cause a transition from STOPPED to the RESETTING state.
STARTING	**State Type: Acting** The machine completes the steps needed to start. This state is entered as a result of a Start command (local or remote). When STARTING completes, the machine will transition to the EXECUTE state.
IDLE	**State Type: Wait** This is the state that indicates that RESETTING is complete. The machine will maintain the conditions that were achieved during the RESETTING state, and perform operations required when the machine is in IDLE.

State Name	Description
SUSPENDING	**State Type: Acting** This state is used when *external* (outside this unit/machine but usually on the same integrated production line) process conditions do not allow the machine to continue producing, that is, the 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 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 Suspend command must be saved in the machine controller when executing the SUSPENDING procedure.
SUSPENDED	**State Type: Wait** Refer to SUSPENDING for when this state is used. In this state the machine does not produce product. 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.
UNSUSPENDING	**State Type: Acting** 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.
EXECUTE	**State Type: Acting** Once the machine is processing materials it is in the EXECUTE state until a transition command is received. 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 produced, while perhaps in a user-defined “Clean Out” mode the EXECUTE state would result in the action of cleaning the machine.
STOPPING	**State Type: Acting** This state is entered in response to a Stop command. While in this state the machine executes the logic that brings it to a controlled stop as reflected by the STOPPED state. Normal STARTING of the machine cannot be initiated unless RESETTING has taken place.
ABORTING	**State Type: Acting** The ABORTING state can be entered at any time in response to the Abort command, typically triggered by the occurrence of a machine event that warrants an aborting action. The aborting logic will bring the machine to a rapid safe stop.
ABORTED	**State Type: Wait** The machine maintains status information relevant to the abort condition. The machine can only exit the ABORTED state after an explicit Clear command, subsequent to manual intervention to correct and reset the detected machine faults.

State Name	Description
HOLDING	**State Type: Acting** This state is 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, that is, the machine leaves the EXECUTE or SUSPENDED states due to internal conditions. This is typically used for routine machine conditions that require minor operator servicing to continue production. This state can be initiated automatically or by an operator and can be easily 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 Hold command must be saved in the machine controller when executing the HOLDING procedure.
HELD	**State Type: Wait** Refer to HOLDING for when this state is used. In this state the machine does not produce product. It will either stop running or continue to dry cycle. A transition to the UNHOLDING state will occur when internal machine conditions change or an Unhold command is initiated by an operator.
UNHOLDING	**State Type: Acting** Refer 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 is required to perform minor servicing to replenish materials or make adjustments, then the Unhold command may be initiated by the operator.
COMPLETING	**State Type: Acting** This state is the result of a Complete command from the EXECUTE, HELD or SUSPENDED states. The Complete command may be internally generated, such as reaching the end of a predefined production count where normal operation has run to completion, or externally generated, such as by a supervisory system. The COMPLETING state is often used to end a production run and summarize production data.
COMPLETED	**State Type: Wait** The machine has finished the COMPLETING state and is now waiting for a Reset command before transitioning to the RESETTING state.
RESETTING	**State Type: Acting** 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. No hazardous motion should happen in this state.
CLEARING	**State Type: Acting** Initiated by a Clear command to clear faults that may have occurred and are present in the ABORTED state before proceeding to a STOPPED state.

4.4 State model

A state model completely defines the behavior of a machine in one unit control mode. In a state model, states are arranged in an ordered fashion that is consistent with the specified operation of the machine. The specific states required are dependent on the machine operation.

The compilation of all defined functional states and their transitions is called the Base State Model.

4.4.1 Base state model

The base state model represents the complete set of defined states, state commands, and state transitions. All unit control modes will be defined by subsets of the base state model. The base state model is depicted in Figure 2.

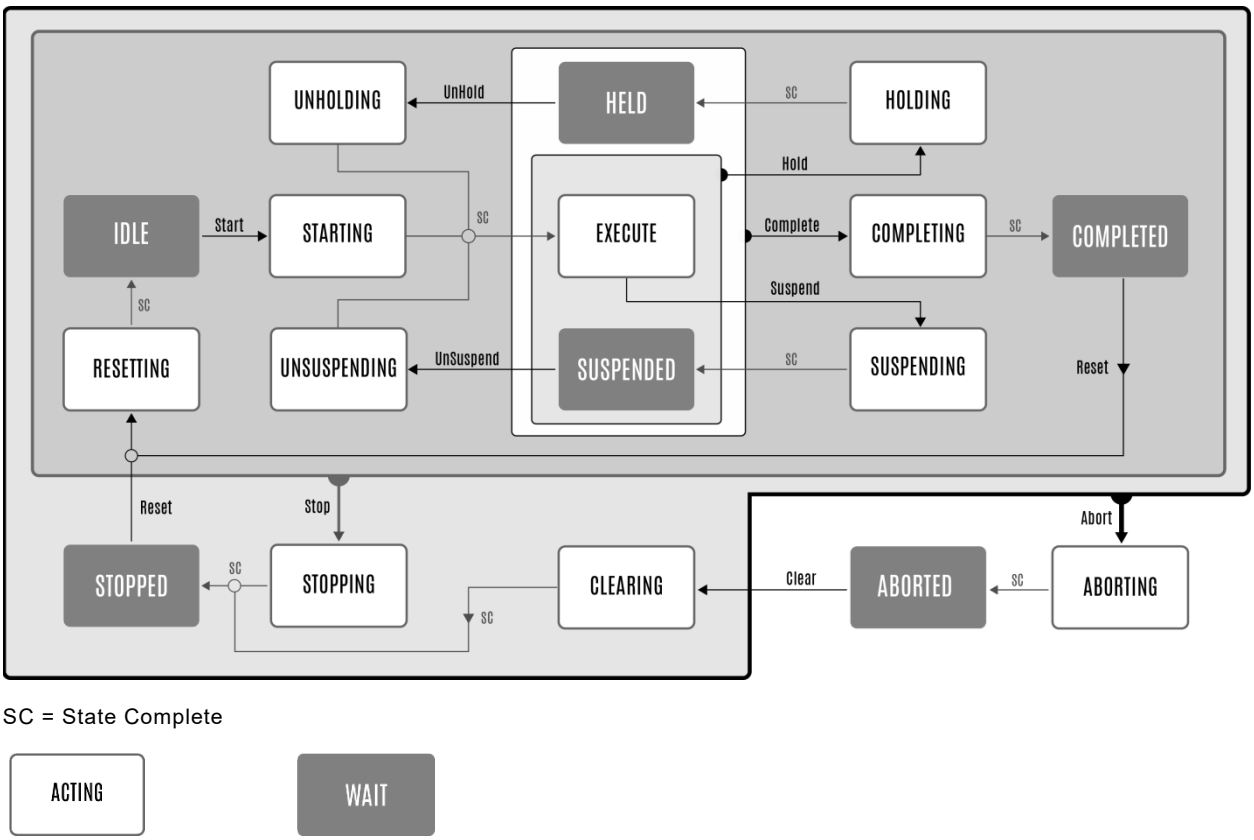

Figure 2 – Base state model visualization

Area boundaries serve to group States together for transitional purposes. A transition is allowed for any State within a boundary of which the base of a transition is anchored.

4.5 State transitions and state commands

4.5.1 Definition

A state transition is defined as a passage from one state to another. Transitions between states will occur as a result of a local, remote, or procedural state command. State commands are procedural elements that in effect cause a state transition to occur.

4.5.2 Types of state commands

State commands are comprised of one or a combination of the following types:

- Operator intervention.
- Response to the status of one or more procedural elements.
- Response to machine conditions.
- The completion of an acting state procedure.
- Supervisory or remote system intervention.

4.5.3 State transition rules

- If an acting state is disabled (not included) in the current unit/machine control mode, the state machine bypasses that state completely.
- If a wait state is disabled (not included) in the current unit/machine control mode, the associated acting states and transitions are also automatically disabled.
- If a state is enabled (included) in the current unit/machine control mode, and the condition to transition out of that state is true upon arriving at that state, the state machine is allowed to immediately transition out of that state without reporting having been in that state.

Example:
The current unit/machine control mode includes the HELD, UNHOLDING, EXECUTE, SUSPENDING and SUSPENDED states. The unit/machine is in the HELD state, receives an Unhold command, and the current state becomes UNHOLDING.

Scenario 1>
UNHOLDING_SC becomes True and the Suspend command is False
→ the current state becomes EXECUTE

Scenario 2>
UNHOLDING_SC becomes True and the Suspend command is True
→ the current state becomes SUSPENDING

In Scenario 2 the state machine transitions directly from UNHOLDING to SUSPENDING, effectively bypassing the EXECUTE state.

4.5.4 Examples of state transitions

An example transition matrix for local or remote state commands generated by an operator is shown in Table 3. After every acting state, except for EXECUTE, a procedural element is required that will indicate the acting state is complete, or a command is required to stop or abort the acting state. The state complete indication within the acting state procedure will cause a state transition to occur.

TR88.00.02-2022 contains three important transitional differences compared to TR88.00.02-2015 and ISA-88.00.01-2010 Annex D.

- The addition of a distinct Complete command to replace EXECUTE_SC. This is a new transition command that allows for a more direct and recognizable departure from the EXECUTE state either by the local control program or from a supervisory system.
- The Complete command also triggers a transition to COMPLETING from the HELD or SUSPENDED states. This allows the control system to wrap up production prematurely if needed even though it is not producing at the time due to an internal or external condition.
- The Hold command triggers a transition directly from SUSPENDED to HOLDING. This allows the machine state to be more accurately reported, especially for OEE calculations, if the machine originally stopped executing due some external condition (blocked or starved) and then later experienced an internal event. This also allows the state model to move from one non-executing state to another without having to move through the EXECUTE state.

An example state command matrix of machine conditions activating a state command is shown in Table 4. The objective of this table is to depict the machine conditions that will cause a state transition using the commands defined in Table 3.

Table 3 – Example transition matrix of local or remote state commands

	State Commands										State Complete
Current State	**Start**	**Reset[3]**	**Hold**	**Unhold**	**Suspend**	**Unsuspend**	**Complete**	**Clear[3]**	**Stop[4]**	**Abort[4]**	
IDLE	STARTING								STOPPING	ABORTING	
STARTING									STOPPING	ABORTING	EXECUTE
EXECUTE			HOLDING		SUSPENDING		COMPLETING		STOPPING	ABORTING	
COMPLETING									STOPPING	ABORTING	COMPLETED
COMPLETED		RESETTING							STOPPING	ABORTING	
RESETTING									STOPPING	ABORTING	IDLE
HOLDING									STOPPING	ABORTING	HELD
HELD				UNHOLDING			COMPLETING		STOPPING	ABORTING	
UNHOLDING									STOPPING	ABORTING	EXECUTE
SUSPENDING									STOPPING	ABORTING	SUSPENDED
SUSPENDED			HOLDING			UNSUSPENDING	COMPLETING		STOPPING	ABORTING	
UNSUSPENDING									STOPPING	ABORTING	EXECUTE
STOPPING										ABORTING	STOPPED
STOPPED		RESETTING								ABORTING	
ABORTING											ABORTED
ABORTED								CLEARING			
CLEARING										ABORTING	STOPPED

[3] It is common practice for Clear and Reset to be initiated using the same physical operator interface device.

[4] It is common practice in packaging (but not process) applications to permit use of Stop and Abort commands while in the IDLE, COMPLETED, STOPPED, and RESETTING states.

Table 4 – Example matrix of machine conditions initiating a state command

	Example Machine Conditions										
Current State	**Operator Start**	**Carton Magazine Low**	**Carton Magazine Full**	**Downstream Not Ready**	**Downstream Ready**	**E-Stop or Fault**	**No Product Present**	**Product Present**	**Operator Stop**	**Product Count Reached**	**Clear Faults**
IDLE	Start					Abort			Stop		
STARTING						Abort			Stop		
EXECUTE		Hold		Suspend		Abort	Suspend		Stop	Complete	
COMPLETING						Abort			Stop		
COMPLETED						Abort			Stop		
RESETTING						Abort			Stop		
HOLDING						Abort			Stop		
HELD			Unhold			Abort			Stop		
UNHOLDING						Abort			Stop		
SUSPENDING						Abort			Stop		
SUSPENDED					Unsuspend	Abort		Unsuspend	Stop		
UNSUSPENDING						Abort			Stop		
STOPPING						Abort					
STOPPED						Abort					
ABORTING											
ABORTED											Clear
CLEARING						Abort					

5 Modes

The ISA-88.00.01 standard provides a set of example modes for equipment entities and procedural elements. This report establishes modes for automated machines that are considered different from the ISA-88.00.01 procedural modes (Automatic, Semi-automatic, and Manual). The ISA-88.00.01 procedural modes describe the way procedures operate. Procedural modes are not common in automated machinery. They are provided for in this report by noting they may exist but are not considered in the scope of this work and are not included in the tag table.

For automated machines, the example in this report establishes unit/machine modes in order to allow a machine designer to adjust the set of states, state commands and state transitions a machine may follow given different operating circumstances. The set of defined unit/machine control modes is shown in Table 5 and described below.

Table 5 – Unit/Machine control modes

ISA-88.00.01 Example Modes	ISA-TR88.00.02 Control Modes	
	Value	Unit/Machine Control Modes
<not defined>	0	Invalid
<not defined>	1	Production
<not defined>	2	Maintenance
<not defined>	3	Manual
<not defined>	4 – 31	User Definable

5.1 Unit/Machine control modes

A unit/machine control mode determines the subset of states, state commands, and state transitions that determine the strategy for carrying out a unit/machine's process.

Typical unit/machine control modes are Production, Maintenance, Manual, Clean In Place, Run Out, Semi-auto, Dry Cycle, etc. The distinguishing elements between these unit control modes are the selected subset of states, state commands, and state transitions.

The PackTag **Admin.DisabledStatesCfg[#]** is used to selectively disable (not include) individual states within a mode, where the array index is the mode number. The PackTag **Admin.CurDisabledStates** is to be populated based on the current mode.

The ordered procedures that control the states may be unique for each of the modes defined within the unit/machine. States of identical names may have different functions in different unit/machine control modes. For example, in a "Production" unit/machine control mode the definition of EXECUTE in a filling machine will mean it is producing product. In the "Manual" unit/machine control mode, the definition of the EXECUTE state may be jogging or indexing.

Examples of unit/machine control modes are:

Production mode: This represents the mode that is utilized for routine production. The machine executes relevant logic in response to commands that are either entered directly by the operator or issued by another supervisory system.

Maintenance mode: This mode may allow suitably authorized personnel the ability to run an individual machine independent of other machines in a production line. This mode would typically

be used for fault finding, machine trials or testing operational improvements. This mode would also allow the speed of the machine to be adjusted (where this feature is available).

Manual mode: This provides direct control of individual machine modules. This feature is available depending upon the mechanical constraints of the mechanisms being exercised. This feature may be used for the commissioning of individual drives, verifying the operation of synchronized drives, testing the drive as a result of modifying parameters etc.

5.2 Unit/Machine control mode management

Automated machinery has unit/machine control modes other than “Production,” as noted earlier. Each unit/machine control mode has its own state model. In order to manage the change from one mode to the next, a method of mode management must be defined. The mode management method determines how, and in what state, a machine may change unit/machine control modes; i.e., the mode management method includes interlocks that prevent the machine changing unit/machine control modes when in inappropriate states.

Unit/machine control mode management enables the machine designer to manage unit/machine control mode transitions. Specification on transitions between unit/machine control modes is left to the user, but typical transition points are at *wait* states. The specification of the unit/machine control mode manager is such that no state or control functions are carried out in this upper-level routine. The intent of the mode manager is to logically supervise when a change in mode can be done, command a mode change, and report status of the change request.

Transitions between unit/machine control modes can occur only at predetermined states:

- As a result of a local or remote operator command.
- As a result of a remote request from another automated unit.
- As a result of a state change. This is generated by change of state of one or a number of machine conditions, either directly from I/O or completion of a logic method. For example, if a filling machine has completed its production run in Production Mode of a given number of cases it may change to the Cleaning Mode to begin a clean cycle.
- When a unit/machine control mode change takes place, a machine state change is NOT allowed to occur at the same time. Mode transitions must take place in a state that is common to both modes. This is necessary to avoid unintended machine sequences from taking place.

Figure 3 visualizes mode management through *mode transition states* as selected by the machine designer. These can be configured using the tag Admin.ModeTransitionCfg[#]. Typically, most designers would choose to allow transitions only at a very limited set of wait states, such as at STOPPED or ABORTED. However, any state common to the current mode and target mode could technically be used in order to satisfy the unit/machine operational requirements. All considerations of a mode manager must be consistent with prevailing safe practices and standards.

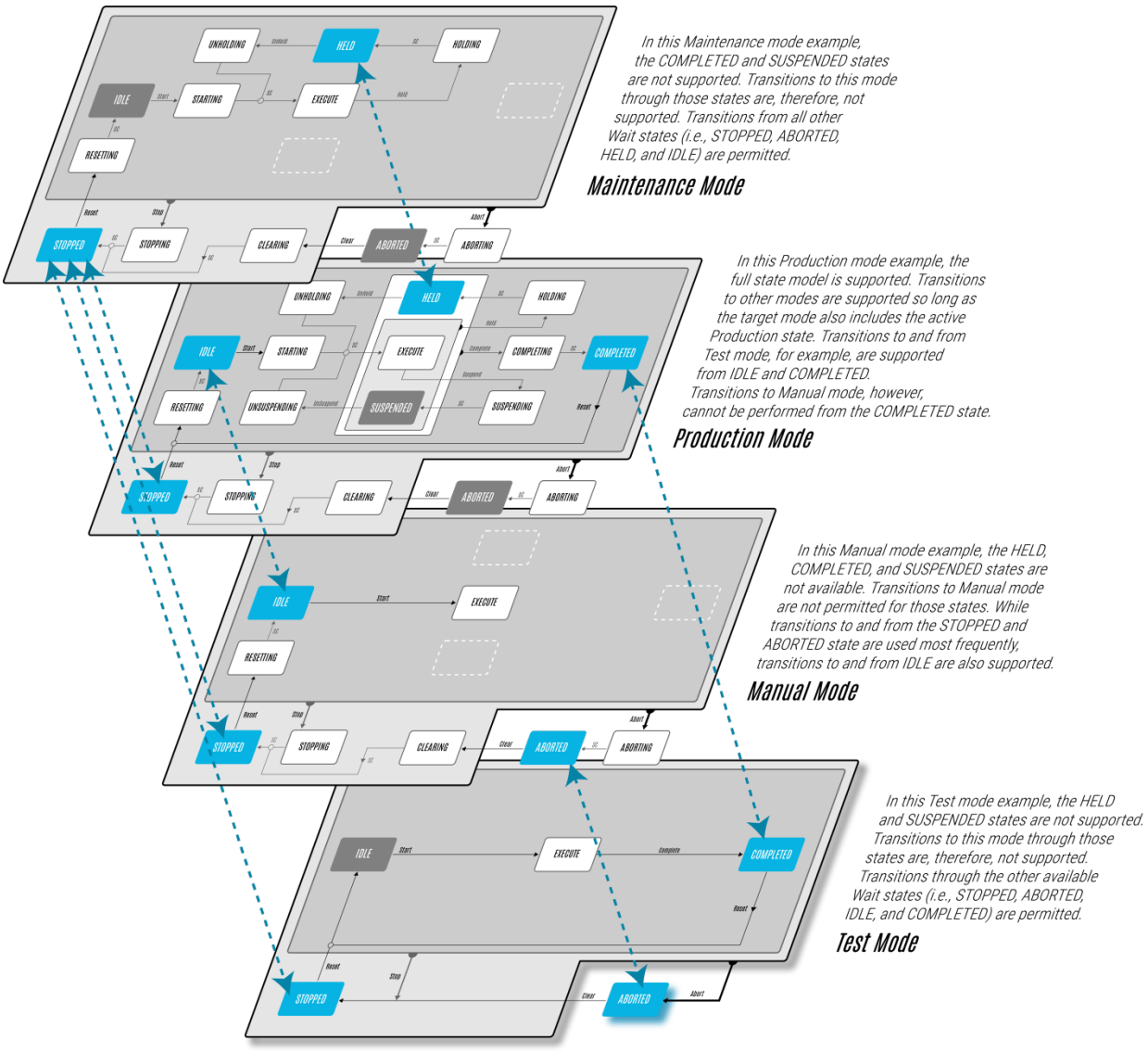

Figure 3 – Example mode management using selected mode transition states

6 Common Unit/Machine mode examples

All modes, either reserved or user defined, may be a subset of the full base state model depending on the machine operational requirements, so long as the minimum set of states as identified in Table 1 are included. Although modes share common state names, the processes or functions that occur in those states may differ for each mode. For example, the EXECUTE state may automatically run a full production sequence when in Production Mode, may run a 'Dry Cycle' sequence without consuming any materials when in Maintenance Mode, and may not run any sequence at all when in Manual Mode.

The following sections provide examples for the three reserved modes: Production, Maintenance and Manual, along with a user-defined mode.

6.1 Production Mode example

Production Mode is the primary operational mode of the machine used to deliver control of routine processing and production. It may contain all or a subset of the unit states in the base state model so long as the minimum required states are included as denoted in 4.3, Table 1. Figure 4 represents a Production Mode containing all states in the base model except COMPLETING and COMPLETED, indicating that the functions in the EXECUTE state run continuously until interrupted by some machine event.

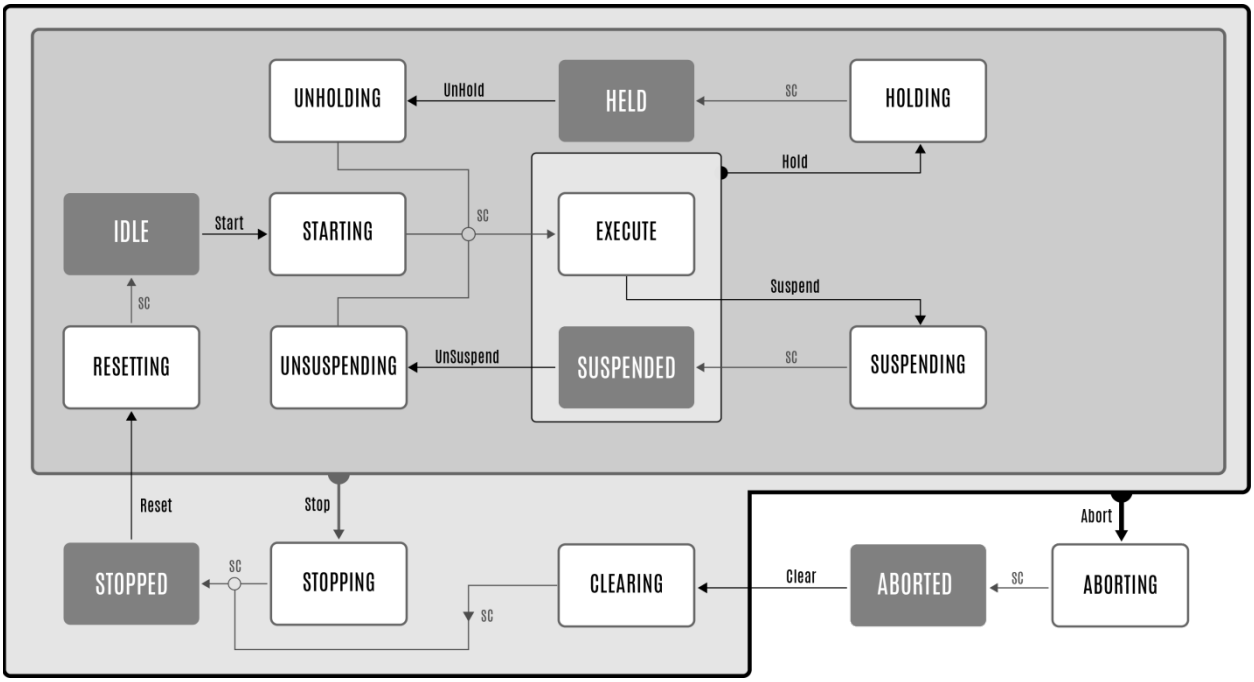

Figure 4 – Production Mode example state diagram

6.2 Maintenance Mode example

Maintenance Mode allows suitably authorized personnel the ability to run an individual machine independent of other machines in a production line. This would typically be used for fault finding, machine trials or testing operational improvements. It is expected that, because the machine will generally operate in its usual manner, it will need to undergo some or all of its routine starting up procedures. This mode is often synonymous with “Dry Run” operation.

Figure 5 represents a Maintenance Mode where the COMPLETE branch has been removed to reflect that no products are being produced, and the SUSPEND branch has been removed to reflect that there are no upstream or downstream dependencies for this mode.

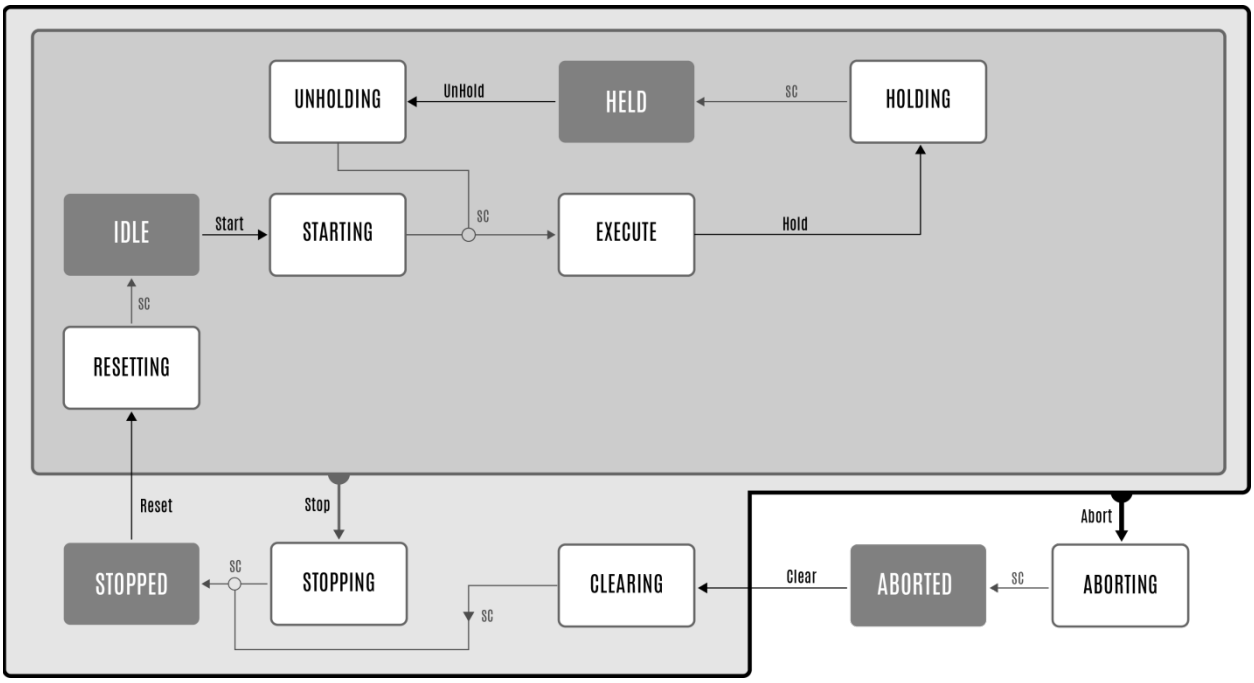

Figure 5 – Maintenance Mode example state diagram

6.3 Manual Mode example

Manual Mode provides suitably authorized personnel the ability to operate individual subordinate equipment controls (such as drive logic) within the machine under manual pushbutton control. Such controls in this mode may be on a “hold-to-run” basis such that removal of the run signal will cause the drive to be stopped. The ability to perform specific functions will be dependent upon mechanical constraints and interlocks. Manual Mode will be of particular use for setting up the machine to work.

Figure 6 represents a Manual Mode where the entire HOLDING, SUSPENDING, and COMPLETING branches have been disabled. In this example, a Reset button on the user interface can be used to issue both the Reset and Start commands simultaneously when in Manual Mode, if safe to do so. This will allow the state model to pass from STOPPED to EXECUTE where manual-type operations can be performed.

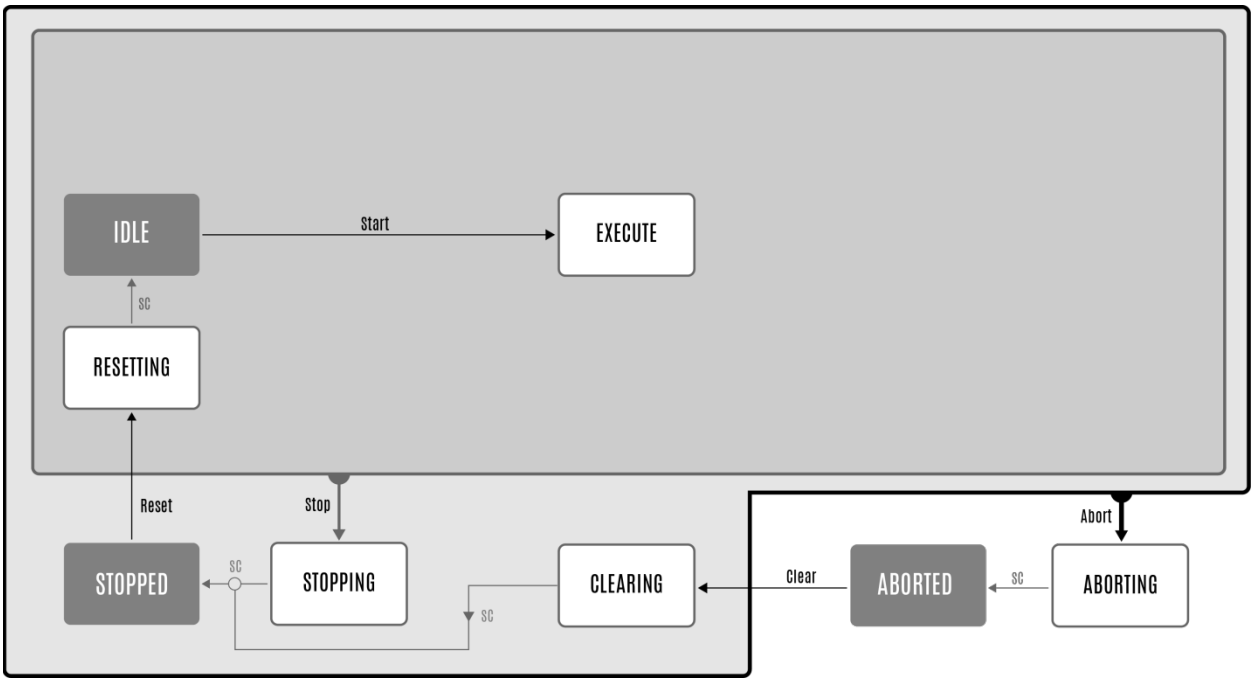

Figure 6 – Manual Mode example state diagram

6.4 User-defined mode example

This report allows for the creation of up to 28 user-defined modes for a unit/machine that provide the user with a great deal of flexibility to create specific machine operating functions. User-defined modes are assigned a mode number 4 to 31 and may contain a varying number of included states from the base state model, so long as at least the minimum states denoted in 4.3, Table 1 are present.

Figure 7 represents a user-defined mode named “Test.” In this example, the machine does not produce or consume any product, but instead runs a series of internal diagnostic checks. If a check fails, then the test will Abort. If all tests pass, then the state will be moved to COMPLETED.

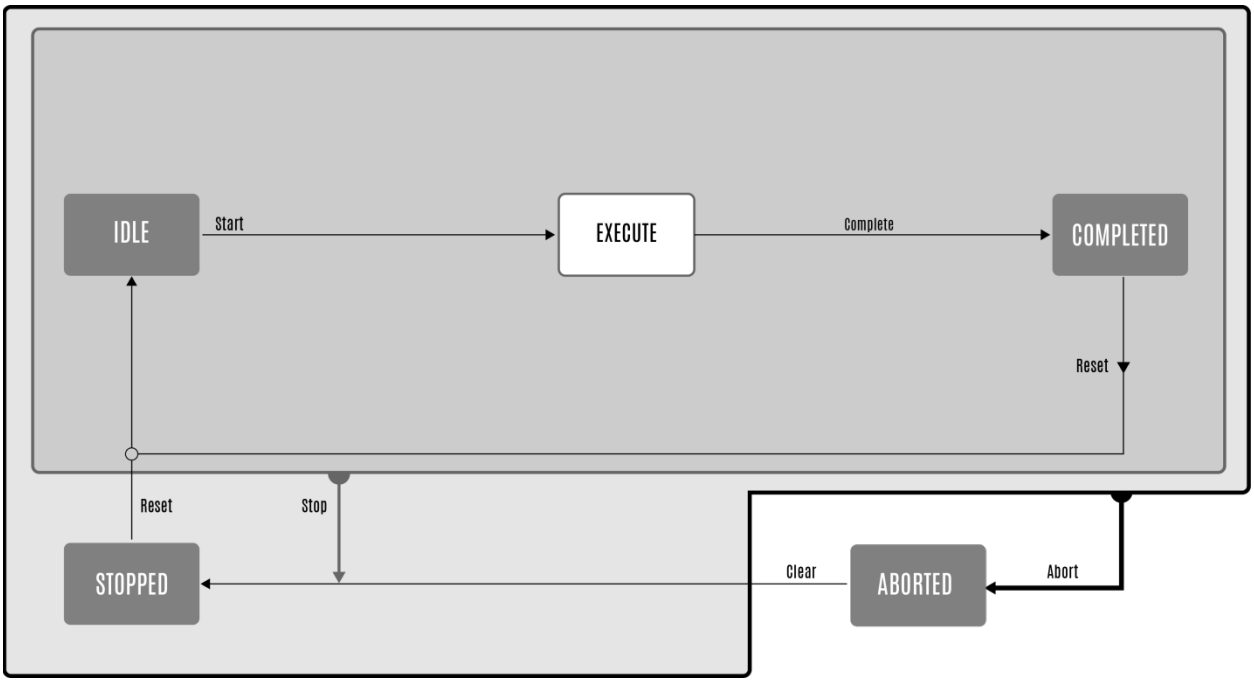

Figure 7 – User-defined "Test" Mode

In this use case, only the minimum states from the base state model are included, along with the COMPLETED state. Note that none of the acting states are present between the wait states. This is allowed per the state transition rules described in 4.4.3.

7 Automated machine functional tag description

7.1 Introduction to PackTags

PackTags provide a uniform set of naming conventions for data elements used within the procedural elements of the base state model. As seen earlier in the document the base state model provides a uniform set of machine states, so that all automated machinery can be looked at in a common way. PackTags are named data elements used for open architecture, interoperable data exchange in automated machinery. This document includes the fundamental names of the data elements as well as the data type, values, ranges and where necessary, data structures. PackTags are useful for machine-to-machine (intermachine) communications, for example between a filler and a capper. PackTags can also be used for data exchange between machines and higher-level information systems like manufacturing operations management and enterprise information systems.

This report defines all the PackTags necessary to navigate through a state model, as well as those that are required to define and manipulate the unit control mode. This report also defines a list of PackTags that will provide necessary information that might be available from a machine.

The use of the minimum set of required PackTags is needed to be consistent with the principles for integrated connectivity with systems using this same implementation method.[5]

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

7.2 Tag types

PackTags are broken out into three groups: Command, Status, and Administration. Command and Status tags contain data required for interfacing between machines and line control for coordination, or for recipe/parameter download. Command tags are “written” to and consumed by the machine program, as the “information receiver,” while Status tags are produced by and read from the machine program. Administration tags contain data collected by higher-level systems for machine performance analysis, or operator information.[6] Generally, informational data is passed using OPC on an Ethernet-based communication network.

- Command tags are prefixed by “Command”.
- Status tags are prefixed by “Status”.
- Administration tags are prefixed by “Admin”.

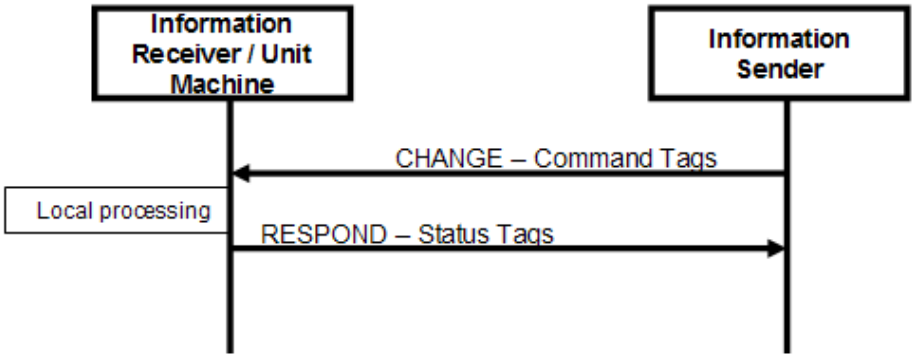

Figure 8 –Tag information flow

7.3 PackTags names

The first letter of each word is capitalized for readability. While IEC 61131 is not case sensitive, to ensure interoperability with all systems it is recommended that the mixed case format be adhered to. Optionally, underscores may also be used in place of the “dot” notation for legacy systems that do not support structured tag names.

Thus, the exact text strings that should be used as tag names should be as follows:

Status.StateCurrent
Status.UnitModeCurrent

7.4 Data types, units, and ranges

Table 6 – Standard data types used for PackTags

Data type IEC 61131-3	Data type ISO/IEC 9899 (C99)	Description	Size
DINT	Long	Signed double integer	32 bits
DWORD	Unsigned Long	Bit string of length 32	32 bits [0..31]

[6] Each grouping of data should be in a contiguous grouping of registers to optimize communications.

REAL	Float	Single precision floating point	32 bits
LREAL	Double	Double precision floating point	64 bits
BOOL	Bool	Boolean	1 bit*
STRING	Char	Null-terminated ASCII, 80 characters	80 byte + header bytes
STRING(6)	Char(6)	Null-terminated ASCII, 6 characters	6 byte + header bytes

*For some platforms and data areas, a BOOL consumes a BYTE (8 bits) of space when placed within a structure.

7.4.1 Structured data types

- PACKMLV2022 – is a placeholder for the unit/machine name and is the top level in the PackTag structure.
  - .Command [PMLc]
  - .Status [PMLs]
  - .Admin [PMLa]

- PMLc – is the collection of all command tags in the PackTag structure.
  - .UnitMode [DINT]
  - .UnitModeChangeRequest [BOOL]
  - .MachSpeed [REAL]
  - .MaterialInterlock [DWORD]
  - .CntrlCmd [DINT]
  - .CmdChangeRequest [BOOL]
  - .Parameter_REAL[#] [PARAMETER_REAL] Array
  - .Parameter_STRING[#] [PARAMETER_STRING] Array
  - .Parameter_LREAL[#] [PARAMETER_LREAL] Array
  - .Parameter_DINT[#] [PARAMETER_DINT] Array
  - .SelectedRecipe [DINT]
  - .RecipeChangeRequest [BOOL]
  - .Recipe[#] [RECIPE] Array

- PMLs – is the collection of all status tags in the PackTag structure.
  - .UnitModeCurrent [DINT]
  - .UnitModeRequested [BOOL]
  - .UnitModeChangeInProcess [BOOL]
  - .StateCurrent [DINT]
  - .StateRequested [DINT]
  - .StateChangeInProcess [BOOL]
  - .MachSpeed [REAL]
  - .CurMachSpeed [REAL]
  - .MaterialInterlock [DWORD]
  - .EquipmentInterlock [EQUIPMENT_INTERLOCK]
  - .Parameter_REAL[#] [PARAMETER_REAL] Array
  - .Parameter_STRING[#] [PARAMETER_STRING] Array
  - .Parameter_LREAL[#] [PARAMETER_LREAL] Array
  - .Parameter_DINT[#] [PARAMETER_DINT] Array

- .RecipeCurrent [DINT]
- .RecipeRequested [DINT]
- .RecipeChangeInProcess [BOOL]
- .Recipe[#] [RECIPE] Array
- .StackLight[#] [DWORD] Array

- PMLa – is the collection of all administration tags in the PackTag structure.
  - .Parameter_REAL[#] [PARAMETER_REAL] Array
  - .Parameter_STRING[#] [PARAMETER_STRING] Array
  - .Parameter_LREAL[#] [PARAMETER_LREAL] Array
  - .Parameter_DINT[#] [PARAMETER_DINT] Array
  - .Alarm[#] [EVENT] Array
  - .AlarmExtent [DINT]
  - .AlarmHistory[#] [EVENT] Array
  - .AlarmHistoryExtent [DINT]
  - .StopReason [EVENT]
  - .Warning[#] [EVENT] Array
  - .WarningExtent [DINT]
  - .ModeTimeCurrent [DINT]
  - .ModeStateCurrent [DINT]
  - .CumulativeTimes[#] [CUMULATIVE_TIMES] Array
  - .ProductData[#] [PRODUCT_DATA] Array
  - .MachDesignSpeed [REAL]
  - .DisabledStatesCfg[#] [DWORD] Array
  - .CurDisabledStates [DWORD]
  - .EnabledModesCfg [DWORD]
  - .ModeTransitionCfg[#] [DWORD] Array
  - .PLCDateTime [DATE_TIME]

- DATE_TIME – is a collection of tags that are used to capture timestamp information about events. This is a user-defined data type that may be adapted to optimize performance on specific platforms, as long as:
  1) The data type consists of a defined structure of integer-type data elements (i.e., SINT, INT, DINT) and is not solely a string that must be parsed. Unique data types may be used for each element,
  2) The data type is not a single number or numbers representing epoch time for which further calculations must be done,
  3) The data type structure contains a minimum set of required elements:
     - .Year [Any Integer]
     - .Month [Any Integer]
     - .Day [Any Integer]
     - .Hour (24hr format) [Any Integer]
     - .Minute [Any Integer]
     - .Second [Any Integer]

  Additional elements may be included by the user at their discretion. Examples of other optional elements are .Millisecond (1/1,000 sec), .Microsecond (1/1,000,000 sec), .DayOfWeek. The structure information and individual element data types must be provided to any other interested parties upon request.

- PARAMETER_REAL – is a collection of tags that are used to describe parameters of data type REAL in the machine unit.
  - .ID [DINT]

- .Name [STRING]
- .Unit [STRING(6)]
- .Value [REAL]

- PARAMETER_STRING – is a collection of tags that are used to describe parameters of data type STRING in the machine unit.
  - .ID [DINT]
  - .Name [STRING]
  - .Unit [STRING(6)]
  - .Value [STRING]

- PARAMETER_LREAL – is a collection of tags that are used to describe parameters of data type LREAL in the machine unit.
  - .ID [DINT]
  - .Name [STRING]
  - .Unit [STRING(6)]
  - .Value [LREAL]

- PARAMETER_DINT – is a collection of tags that are used to describe parameters of data type DINT in the machine unit.
  - .ID [DINT]
  - .Name [STRING]
  - .Unit [STRING(6)]
  - .Value [DINT]

- RECIPE – is a collection of tags used to describe the product that the machine is making or can make.
  - .ID [DINT]
  - .Name [STRING]
  - .Unit [STRING(6)]
  - .PrimaryQty [REAL]
  - .ProcessVariables [PROCESS_VARIABLES]
  - .Ingredients [INGREDIENTS]

- PRODUCT_DATA – is a collection of tags that are used to provide descriptions and usage information for all the product input and output streams of the machine unit.
  - .ID [DINT]
  - .Name [STRING]
  - .Unit [STRING(6)]
  - .PrimaryQty [REAL]
  - .ConsumedCount [DINT]
  - .ProcessedCount [DINT]
  - .DefectiveCount [DINT]
  - .AccConsumedCount [DINT]
  - .AccProcessedCount [DINT]
  - .AccDefectiveCount [DINT]

- CUMULATIVE_TIMES – is a collection of tags that are used to track the total amount of cumulative time a machine unit has spent in each Mode and State. This data type is used inside the PMLa structure as an array. The user is able to decide the extent of the array depending on the strategy for resetting times. The minimum array extent is 1 (Admin.CumulativeTimes[0]).
  - .AccTimeSinceReset [DINT]

  - .ModeStateTimes[#] [MODESTATE_TIMES] Array

- MODESTATE_TIMES – is a structure of tags that are used to store cumulative time for each Mode and State. This data type is used inside the CUMULATIVE_TIMES structure as an array. The user is able to decide the extent of the array depending on how many Modes are defined for the system. The minimum array extent is 4 [0..3].
  - Mode [DINT]
  - State[#] [DINT] Array. The array extent is fixed at [0..17]

- PROCESS_VARIABLES – is a collection of tags used to describe the key process variables for a given recipe, such as speed or time setpoints. The user is able to define the extent of the parameter arrays.
  - Parameter_REAL [#] [PARAMETER_REAL] Array
  - Parameter_STRING [#] [PARAMETER_STRING] Array
  - Parameter_LREAL [#] [PARAMETER_LREAL] Array
  - Parameter_DINT [#] [PARAMETER_DINT] Array

- INGREDIENTS – is a collection of tags used to describe the raw materials that are needed for a given recipe. Each Parameter can be considered to be a single ingredient and may include such things as a particular weight or quantity in relation to each output unit of the recipe. The user is able to define the extent of the parameter arrays.
  - Parameter_REAL [#] [PARAMETER_REAL] Array
  - Parameter_STRING [#] [PARAMETER_STRING] Array
  - Parameter_LREAL [#] [PARAMETER_LREAL] Array
  - Parameter_DINT [#] [PARAMETER_DINT] Array

- EVENT – is the collection tags needed to capture machine events such as Alarms, Warnings, or other occurrences.
  - .Trigger [BOOL]
  - .ID [DINT]
  - .Value [DINT]
  - .Message [STRING]
  - .Category [DINT]
  - .DateTime [DATE_TIME]
  - .AckDateTime [DATE_TIME]

- EQUIPMENT_INTERLOCK – is a collection of tags that are used to provide an indicator of Blocked or Starved conditions at the machine unit when integrated into a complete production line.
  - .Blocked [BOOL]
  - .Starved [BOOL]

7.5 Tag details

This section provides a summary listing of the tags and detailed tag descriptions. Table 7, Table 8, and Table 9 list the Command, Status, and Admin PackTags. Table 10 and Table 11 capture the minimum set of tags to be consistent with the technical report.

Table 7 – Command tags (complete listing)

MIN REQ						TAG NAME	DATA TYPE
UnitName						UnitName	PACKMLv2022
	Command					UnitName.Command	PMLc
x		UnitMode				UnitName.Command.UnitMode	DINT
x		UnitModeChangeRequest				UnitName.Command.UnitModeChangeRequest	BOOL
x		MachSpeed				UnitName.Command.MachSpeed	REAL
		MaterialInterlock				UnitName.Command.MaterialInterlock	DWORD
x		CntrlCmd				UnitName.Command.CntrlCmd	DINT
x		CmdChangeRequest				UnitName.Command.CmdChangeRequest	BOOL
		Parameter_REAL[#]				UnitName.Command.Parameter_REAL[#]	PARAMETER_REAL Array
			ID			UnitName.Command.Parameter_REAL[#].ID	DINT
			Name			UnitName.Command.Parameter_REAL[#].Name	STRING
			Unit			UnitName.Command.Parameter_REAL[#].Unit	STRING(6)
			Value			UnitName.Command.Parameter_REAL[#].Value	REAL
		Parameter_STRING[#]				UnitName.Command.Parameter_STRING[#]	PARAMETER_STRING Array
			ID			UnitName.Command.Parameter_STRING[#].ID	DINT
			Name			UnitName.Command.Parameter_STRING[#].Name	STRING
			Unit			UnitName.Command.Parameter_STRING[#].Unit	STRING(6)
			Value			UnitName.Command.Parameter_STRING[#].Value	STRING
		Parameter_LREAL[#]				UnitName.Command.Parameter_LREAL[#]	PARAMETER _LREAL Array
			ID			UnitName.Command.Parameter_LREAL[#].ID	DINT
			Name			UnitName.Command.Parameter_LREAL[#].Name	STRING
			Unit			UnitName.Command.Parameter_LREAL[#].Unit	STRING(6)
			Value			UnitName.Command.Parameter_LREAL[#].Value	LREAL
		Parameter_DINT[#]				UnitName.Command.Parameter_DINT[#]	PARAMETER _DINT Array
			ID			UnitName.Command.Parameter_DINT[#].ID	DINT

MIN REQ						TAG NAME	DATA TYPE
			Name			UnitName.Command.Parameter_DINT[#].Name	STRING
			Unit			UnitName.Command.Parameter_DINT[#].Unit	STRING(6)
			Value			UnitName.Command.Parameter_DINT[#].Value	DINT
		SelectedRecipe				UnitName.Command.SelectedRecipe	DINT
		RecipeChangeRequest				UnitName.Command.RecipeChangeRequest	BOOL
		Recipe[#]				UnitName.Command.Recipe[#]	RECIPE Array
			ID			UnitName.Command.Recipe[#].ID	DINT
			Name			UnitName.Command.Recipe[#].Name	STRING
			Unit			UnitName.Command.Recipe[#].Unit	STRING(6)
			PrimaryQty			UnitName.Command.Recipe[#].PrimaryQty	REAL
			ProcessVariables			UnitName.Command.Recipe[#].ProcessVariables	PROCESS_VARIABLES
				Parameter_ REAL[#]		UnitName.Command.Recipe[#].ProcessVariables.Parameter _REAL[#]	PARAMETER_REAL Array
					ID	UnitName.Command.Recipe[#].ProcessVariables.Parameter _REAL[#].ID	DINT
					Name	UnitName.Command.Recipe[#].ProcessVariables.Parameter _REAL[#].Name	STRING
					Unit	UnitName.Command.Recipe[#].ProcessVariables.Parameter _REAL[#].Unit	STRING(6)
					Value	UnitName.Command.Recipe[#].ProcessVariables.Parameter _REAL[#].Value	REAL
				Parameter_ STRING[#]		UnitName.Command.Recipe[#]. ProcessVariables.Parameter_STRING[#]	PARAMETER_STRING Array
					ID	UnitName.Command.Recipe[#].ProcessVariables.Parameter _STRING[#].ID	DINT
					Name	UnitName.Command.Recipe[#].ProcessVariables.Parameter _STRING[#].Name	STRING
					Unit	UnitName.Command.Recipe[#].ProcessVariables.Parameter _STRING[#].Unit	STRING(6)
					Value	UnitName.Command.Recipe[#].ProcessVariables.Parameter _STRING[#].Value	STRING

MIN REQ						TAG NAME	DATA TYPE
				Parameter_ LREAL[#]		UnitName.Command.Recipe[#]. ProcessVariables.Parameter_LREAL[#]	PARAMETER_LREAL Array
					ID	UnitName.Command.Recipe[#].ProcessVariables.Parameter _LREAL[#].ID	DINT
					Name	UnitName.Command.Recipe[#].ProcessVariables.Parameter _LREAL[#].Name	STRING
					Unit	UnitName.Command.Recipe[#].ProcessVariables.Parameter _LREAL[#].Unit	STRING(6)
					Value	UnitName.Command.Recipe[#].ProcessVariables.Parameter _LREAL[#].Value	LREAL
				Parameter_ DINT[#]		UnitName.Command.Recipe[#].ProcessVariables.Parameter _DINT[#]	PARAMETER_DINT Array
					ID	UnitName.Command.Recipe[#].ProcessVariables.Parameter _DINT[#].ID	DINT
					Name	UnitName.Command.Recipe[#].ProcessVariables.Parameter _DINT[#].Name	STRING
					Unit	UnitName.Command.Recipe[#].ProcessVariables.Parameter _DINT[#].Unit	STRING(6)
					Value	UnitName.Command.Recipe[#].ProcessVariables.Parameter _DINT[#].Value	DINT
			Ingredients			UnitName.Command.Recipe[#].Ingredients	INGREDIENTS
				Parameter_ REAL[#]		UnitName.Command.Recipe[#].Ingredients.Parameter_REAL [#]	PARAMETER_REAL Array
					ID	UnitName.Command.Recipe[#].Ingredients.Parameter_REAL [#].ID	DINT
					Name	UnitName.Command.Recipe[#].Ingredients.Parameter_REAL [#].Name	STRING
					Unit	UnitName.Command.Recipe[#].Ingredients.Parameter_REAL [#].Unit	STRING(6)
					Value	UnitName.Command.Recipe[#].Ingredients.Parameter_REAL [#].Value	REAL
				Parameter_ STRING[#]		UnitName.Command.Recipe [#].Ingredients.Parameter_STRING[#]	PARAMETER_STRING Array
					ID	UnitName.Command.Recipe[#].Ingredients.Parameter_STRI NG[#].ID	DINT

MIN REQ						TAG NAME	DATA TYPE
					Name	UnitName.Command.Recipe[#].Ingredients.Parameter_STRING[#].Name	STRING
					Unit	UnitName.Command.Recipe[#].Ingredients.Parameter_STRING[#].Unit	STRING(6)
					Value	UnitName.Command.Recipe[#].Ingredients.Parameter_STRING[#].Value	STRING
				Parameter_LREAL[#]		UnitName.Command.Recipe[#].Ingredients.Parameter_LREAL[#]	PARAMETER_LREAL Array
					ID	UnitName.Command.Recipe[#].Ingredients.Parameter_LREAL[#].ID	DINT
					Name	UnitName.Command.Recipe[#].Ingredients.Parameter_LREAL[#].Name	STRING
					Unit	UnitName.Command.Recipe[#].Ingredients.Parameter_LREAL[#].Unit	STRING(6)
					Value	UnitName.Command.Recipe[#].Ingredients.Parameter_LREAL[#].Value	LREAL
				Parameter_DINT[#]		UnitName.Command.Recipe[#].Ingredients.Parameter_DINT[#]	PARAMETER_DINT Array
					ID	UnitName.Command.Recipe[#].Ingredients.Parameter_DINT[#].ID	DINT
					Name	UnitName.Command.Recipe[#].Ingredients.Parameter_DINT[#].Name	STRING
					Unit	UnitName.Command.Recipe[#].Ingredients.Parameter_DINT[#].Unit	STRING(6)
					Value	UnitName.Command.Recipe[#].Ingredients.Parameter_DINT[#].Value	DINT

Table 8 – Status tags (complete listing)

MIN REQ						TAGNAME	DATATYPE
UnitName						UnitName	PACKMLv2022
	Status					UnitName.Status	PMLs
x		UnitModeCurrent				UnitName.Status.UnitModeCurrent	DINT
		UnitModeRequested				UnitName.Status.UnitModeRequested	BOOL
		UnitModeChangeInProcess				UnitName.Status.UnitModeChangeInProcess	BOOL
x		StateCurrent				UnitName.Status.StateCurrent	DINT
		StateRequested				UnitName.Status.StateRequested	DINT
		StateChangeInProcess				UnitName.Status.StateChangeInProcess	BOOL
x		MachSpeed				UnitName.Status.MachSpeed	REAL
x		CurMachSpeed				UnitName.Status.CurMachSpeed	REAL
		MaterialInterlock				UnitName.Status.MaterialInterlock	DWORD
		EquipmentInterlock				UnitName.Status.EquipmentInterlock	EQUIPMENT_INTERLOCK
			Blocked			UnitName.Status.EquipmentInterlock.Blocked	BOOL
			Starved			UnitName.Status.EquipmentInterlock.Starved	BOOL
		Parameter_REAL [#]				UnitName.Status.Parameter_REAL[#]	PARAMETER_REAL Array
			ID			UnitName.Status.Parameter_REAL[#].ID	DINT
			Name			UnitName.Status.Parameter_REAL[#].Name	STRING
			Unit			UnitName.Status.Parameter_REAL[#].Unit	STRING(6)
			Value			UnitName.Status.Parameter_REAL[#].Value	REAL
		Parameter_STRING[#]				UnitName. Status.Parameter_STRING[#]	PARAMETER_STRING Array
			ID			UnitName. Status.Parameter_STRING[#].ID	DINT
			Name			UnitName. Status.Parameter_STRING[#].Name	STRING
			Unit			UnitName. Status.Parameter_STRING[#].Unit	STRING(6)
			Value			UnitName. Status.Parameter_STRING[#].Value	STRING
		Parameter_LREAL[#]				UnitName. Status.Parameter_LREAL[#]	PARAMETER_LREAL Array
			ID			UnitName. Status.Parameter_LREAL[#].ID	DINT

MIN REQ						TAGNAME	DATATYPE
			Name			UnitName. Status.Parameter_LREAL[#].Name	STRING
			Unit			UnitName. Status.Parameter_LREAL[#].Unit	STRING(6)
			Value			UnitName. Status.Parameter_LREAL[#].Value	LREAL
		Parameter_DINT[#]				UnitName. Status.Parameter_DINT[#]	PARAMETER_DINT Array
			ID			UnitName. Status.Parameter_DINT[#].ID	DINT
			Name			UnitName. Status.Parameter_DINT[#].Name	STRING
			Unit			UnitName. Status.Parameter_DINT[#].Unit	STRING(6)
			Value			UnitName. Status.Parameter_DINT[#].Value	DINT
		RecipeCurrent				UnitName.Status.RecipeCurrent	DINT
		RecipeRequested				UnitName.Status.RecipeRequested	DINT
		RecipeChangeInProcess				UnitName.Status.RecipeChangeInProcess	BOOL
		Recipe[#]				UnitName.Status.Recipe[#]	RECIPE Array
			ID			UnitName.Status.Recipe[#].ID	DINT
			Name			UnitName.Status.Recipe[#].Name	STRING
			Unit			UnitName.Status.Recipe[#].Unit	STRING(6)
			PrimaryQty			UnitName.Status.Recipe[#].PrimaryQty	REAL
			ProcessVariables			UnitName.Status.Recipe[#].ProcessVariables	PROCESS_VARIABLES
				Parameter_REAL[#]		UnitName.Status.Recipe[#].ProcessVariables.Parameter_REAL[#]	PARAMETER_REAL Array
					ID	UnitName.Status.Recipe[#].ProcessVariables.Parameter_REAL[#].ID	DINT
					Name	UnitName.Status.Recipe[#].ProcessVariables.Parameter_REAL[#].Name	STRING
					Unit	UnitName.Status.Recipe[#].ProcessVariables.Parameter_REAL[#].Unit	STRING(6)
					Value	UnitName.Status.Recipe[#].ProcessVariables.Parameter_REAL[#].Value	REAL
				Parameter_STRING[#]		UnitName.Status.Recipe[#]. ProcessVariables.Parameter_STRING[#]	PARAMETER_STRING Array
					ID	UnitName.Status.Recipe[#].ProcessVariables.Parameter_S	DINT

MIN REQ						TAGNAME	DATATYPE
						TRING[#].ID	
					Name	UnitName.Status.Recipe[#].ProcessVariables.Parameter_STRING[#].Name	STRING
					Unit	UnitName.Status.Recipe[#].ProcessVariables.Parameter_STRING[#].Unit	STRING(6)
					Value	UnitName.Status.Recipe[#].ProcessVariables.Parameter_STRING[#].Value	STRING
				Parameter_LREAL[#]		UnitName.Status.Recipe[#].ProcessVariables.Parameter_LREAL[#]	PARAMETER_LREAL Array
					ID	UnitName.Status.Recipe[#].ProcessVariables.Parameter_LREAL[#].ID	DINT
					Name	UnitName.Status.Recipe[#].ProcessVariables.Parameter_LREAL[#].Name	STRING
					Unit	UnitName.Status.Recipe[#].ProcessVariables.Parameter_LREAL[#].Unit	STRING(6)
					Value	UnitName.Status.Recipe[#].ProcessVariables.Parameter_LREAL[#].Value	LREAL
				Parameter_DINT[#]		UnitName.Status.Recipe[#].ProcessVariables.Parameter_DINT[#]	PARAMETER_DINT Array
					ID	UnitName.Status.Recipe[#].ProcessVariables.Parameter_DINT[#].ID	DINT
					Name	UnitName.Status.Recipe[#].ProcessVariables.Parameter_DINT[#].Name	STRING
					Unit	UnitName.Status.Recipe[#].ProcessVariables.Parameter_DINT[#].Unit	STRING(6)
					Value	UnitName.Status.Recipe[#].ProcessVariables.Parameter_DINT[#].Value	DINT
			Ingredients			UnitName.Status.Recipe[#].Ingredients	INGREDIENTS
				Parameter_REAL[#]		UnitName.Status.Recipe[#].Ingredients.Parameter_REAL[#]	PARAMETER_REAL Array
					ID	UnitName.Status.Recipe[#].Ingredients.Parameter_REAL[#].ID	DINT
					Name	UnitName.Status.Recipe[#].Ingredients.Parameter_REAL[#].Name	STRING
					Unit	UnitName.Status.Recipe[#].Ingredients.Parameter_REAL[#	STRING(6)

MIN REQ						TAGNAME	DATATYPE
						].Unit	
					Value	UnitName.Status.Recipe[#].Ingredients.Parameter_REAL[#].Value	REAL
				Parameter_STRING[#]		UnitName.Status.Recipe[#].Ingredients.Parameter_STRING[#]	PARAMETER_STRING Array
					ID	UnitName.Status.Recipe[#].Ingredients.Parameter_STRING[#].ID	DINT
					Name	UnitName.Status.Recipe[#].Ingredients.Parameter_STRING[#].Name	STRING
					Unit	UnitName.Status.Recipe[#].Ingredients.Parameter_STRING[#].Unit	STRING(6)
					Value	UnitName.Status.Recipe[#].Ingredients.Parameter_STRING[#].Value	STRING
				Parameter_LREAL[#]		UnitName.Status.Recipe[#].Ingredients.Parameter_LREAL[#]	PARAMETER_LREAL Array
					ID	UnitName.Status.Recipe[#].Ingredients.Parameter_LREAL[#].ID	DINT
					Name	UnitName.Status.Recipe[#].Ingredients.Parameter_LREAL[#].Name	STRING
					Unit	UnitName.Status.Recipe[#].Ingredients.Parameter_LREAL[#].Unit	STRING(6)
					Value	UnitName.Status.Recipe[#].Ingredients.Parameter_LREAL[#].Value	LREAL
				Parameter_DINT[#]		UnitName.Status.Recipe[#].Ingredients.Parameter_DINT[#]	PARAMETER_DINT Array
					ID	UnitName.Status.Recipe[#].Ingredients.Parameter_DINT[#].ID	DINT
					Name	UnitName.Status.Recipe[#].Ingredients.Parameter_DINT[#].Name	STRING
					Unit	UnitName.Status.Recipe[#].Ingredients.Parameter_DINT[#].Unit	STRING(6)
					Value	UnitName.Status.Recipe[#].Ingredients.Parameter_DINT[#].Value	DINT
		Stacklight[#]				UnitName.Status.Stacklight[#]	DWORD Array

Table 9 – Admin tags (complete listing)

MIN REQ						TAGNAME	DATATYPE
UnitName						UnitName	PACKMLv2022
	Admin					UnitName.Admin	PMLa
		Parameter_REAL [#]				UnitName.Admin.Parameter_REAL[#]	PARAMETER_REAL Array
			ID			UnitName.Admin.Parameter_REAL[#].ID	DINT
			Name			UnitName.Admin.Parameter_REAL[#].Name	STRING
			Unit			UnitName.Admin.Parameter_REAL[#].Unit	STRING(6)
			Value			UnitName.Admin.Parameter_REAL[#].Value	REAL
		Parameter_STRING[#]				UnitName. Admin.Parameter_STRING[#]	PARAMETER_STRING Array
			ID			UnitName. Admin.Parameter_STRING[#].ID	DINT
			Name			UnitName. Admin.Parameter_STRING[#].Name	STRING
			Unit			UnitName. Admin.Parameter_STRING[#].Unit	STRING(6)
			Value			UnitName. Admin.Parameter_STRING[#].Value	STRING
		Parameter_LREAL[#]				UnitName. Admin.Parameter_LREAL[#]	PARAMETER_LREAL Array
			ID			UnitName. Admin.Parameter_LREAL[#].ID	DINT
			Name			UnitName. Admin.Parameter_LREAL[#].Name	STRING
			Unit			UnitName. Admin.Parameter_LREAL[#].Unit	STRING(6)
			Value			UnitName. Admin.Parameter_LREAL[#].Value	LREAL
		Parameter_DINT[#]				UnitName. Admin.Parameter_DINT[#]	PARAMETER _DINT Array
			ID			UnitName. Admin.Parameter_DINT[#].ID	DINT
			Name			UnitName. Admin.Parameter_DINT[#].Name	STRING
			Unit			UnitName. Admin.Parameter_DINT[#].Unit	STRING(6)
			Value			UnitName. Admin.Parameter_DINT [#].Value	DINT
		Alarm[#]				UnitName.Admin.Alarm[#]	EVENT Array
			Trigger			UnitName.Admin.Alarm[#].Trigger	BOOL
			ID			UnitName.Admin.Alarm[#].ID	DINT
			Value			UnitName.Admin.Alarm[#].Value	DINT

MIN REQ						TAGNAME	DATATYPE
			Message			UnitName.Admin.Alarm[#].Message	STRING
			Category			UnitName.Admin.Alarm[#].Category (Event Grouping)	DINT
			DateTime			UnitName.Admin.Alarm[#].DateTime	DATE_TIME
				Year		UnitName.Admin.Alarm[#].DateTime.Year	User-defined Numeric
				Month		UnitName.Admin.Alarm[#].DateTime.Month	User-defined Numeric
				Day		UnitName.Admin.Alarm[#].DateTime.Day	User-defined Numeric
				Hour		UnitName.Admin.Alarm[#].DateTime.Hour	User-defined Numeric
				Min		UnitName.Admin.Alarm[#].DateTime.Minute	User-defined Numeric
				Sec		UnitName.Admin.Alarm[#].DateTime.Second	User-defined Numeric
			AckDateTime			UnitName.Admin.Alarm[#].AckDateTime	DATE_TIME
				Year		UnitName.Admin.Alarm[#].AckDateTime.Year	User-defined Numeric
				Month		UnitName.Admin.Alarm[#].AckDateTime.Month	User-defined Numeric
				Day		UnitName.Admin.Alarm[#].AckDateTime.Day	User-defined Numeric
				Hour		UnitName.Admin.Alarm[#].AckDateTime.Hour	User-defined Numeric
				Min		UnitName.Admin.Alarm[#].AckDateTime.Minute	User-defined Numeric
				Sec		UnitName.Admin.Alarm[#].AckDateTime.Second	User-defined Numeric
		AlarmExtent				UnitName.Admin.AlarmExtent	DINT
		AlarmHistory[#]				UnitName.Admin.AlarmHistory[#]	EVENT Array
			Trigger			UnitName.Admin.AlarmHistory[#].Trigger	BOOL
			ID			UnitName.Admin.AlarmHistory[#].ID	DINT
			Value			UnitName.Admin.AlarmHistory[#].Value	DINT
			Message			UnitName.Admin.AlarmHistory[#].Message	STRING
			Category			UnitName.Admin.AlarmHistory[#].Category (Event Grouping)	DINT
			DateTime			UnitName.Admin.AlarmHistory[#].DateTime	DATE_TIME
				Year		UnitName.Admin.AlarmHistory[#].DateTime.Year	User-defined Numeric
				Month		UnitName.Admin.AlarmHistory[#].DateTime.Month	User-defined Numeric
				Day		UnitName.Admin.AlarmHistory[#].DateTime.Day	User-defined Numeric
				Hour		UnitName.Admin.AlarmHistory[#].DateTime.Hour	User-defined Numeric

MIN REQ						TAGNAME	DATATYPE
				Min		UnitName.Admin.AlarmHistory[#].DateTime.Minute	User-defined Numeric
				Sec		UnitName.Admin.AlarmHistory[#].DateTime.Second	User-defined Numeric
			AckDateTime			UnitName.Admin.AlarmHistory[#].AckDateTime	DATE_TIME
				Year		UnitName.Admin.AlarmHistory[#].AckDateTime.Year	User-defined Numeric
				Month		UnitName.Admin.AlarmHistory[#].AckDateTime.Month	User-defined Numeric
				Day		UnitName.Admin.AlarmHistory[#].AckDateTime.Day	User-defined Numeric
				Hour		UnitName.Admin.AlarmHistory[#].AckDateTime.Hour	User-defined Numeric
				Min		UnitName.Admin.AlarmHistory[#].AckDateTime.Minute	User-defined Numeric
				Sec		UnitName.Admin.AlarmHistory[#].AckDateTime.Second	User-defined Numeric
		AlarmHistoryExtent				UnitName.Admin.AlarmHistoryExtent	DINT
		StopReason				UnitName.Admin.StopReason	EVENT
			Trigger			UnitName.Admin.StopReason.Trigger	BOOL
x			ID			UnitName.Admin.StopReason.ID	DINT
			Value			UnitName.Admin.StopReason.Value	DINT
			Message			UnitName.Admin.StopReason.Message	STRING
			Category			UnitName.Admin.StopReason.Category	DINT
			DateTime			UnitName.Admin.StopReason.DateTime	DATE_TIME
				Year		UnitName.Admin.StopReason.DateTime.Year	User-defined Numeric
				Month		UnitName.Admin.StopReason.DateTime.Month	User-defined Numeric
				Day		UnitName.Admin.StopReason.DateTime.Day	User-defined Numeric
				Hour		UnitName.Admin.StopReason.DateTime.Hour	User-defined Numeric
				Min		UnitName.Admin.StopReason.DateTime.Minute	User-defined Numeric
				Sec		UnitName.Admin.StopReason.DateTime.Second	User-defined Numeric
			AckDateTime			UnitName.Admin.StopReason.AckDateTime	DATE_TIME
				Year		UnitName.Admin.StopReason.AckDateTime.Year	User-defined Numeric
				Month		UnitName.Admin.StopReason.AckDateTime.Month	User-defined Numeric
				Day		UnitName.Admin.StopReason.AckDateTime.Day	User-defined Numeric
				Hour		UnitName.Admin.StopReason.AckDateTime.Hour	User-defined Numeric

MIN REQ						TAGNAME	DATATYPE
				Min		UnitName.Admin.StopReason.AckDateTime.Minute	User-defined Numeric
				Sec		UnitName.Admin.StopReason.AckDateTime.Second	User-defined Numeric
		Warning[#]				UnitName.Admin.Warning[#]	EVENT Array
			Trigger			UnitName.Admin.Warning[#].Trigger	BOOL
			ID			UnitName.Admin.Warning[#].ID	DINT
			Value			UnitName.Admin.Warning[#].Value	DINT
			Message			UnitName.Admin.Warning[#].Message	STRING
			Category			UnitName.Admin.Warning[#].Category	DINT
			DateTime			UnitName.Admin.Warning[#].DateTime	DATE_TIME
				Year		UnitName.Admin.Warning[#].DateTime.Year	User-defined Numeric
				Month		UnitName.Admin.Warning [#].DateTime.Month	User-defined Numeric
				Day		UnitName.Admin.Warning [#].DateTime.Day	User-defined Numeric
				Hour		UnitName.Admin.Warning [#].DateTime.Hour	User-defined Numeric
				Min		UnitName.Admin.Warning [#].DateTime.Minute	User-defined Numeric
				Sec		UnitName.Admin.Warning [#].DateTime.Second	User-defined Numeric
			AckDateTime			UnitName.Admin.Warning[#].AckDateTime	DATE_TIME
				Year		UnitName.Admin.Warning[#].AckDateTime.Year	User-defined Numeric
				Month		UnitName.Admin.Warning[#].AckDateTime.Month	User-defined Numeric
				Day		UnitName.Admin.Warning[#].AckDateTime,Day	User-defined Numeric
				Hour		UnitName.Admin.Warning[#].AckDateTime,Hour	User-defined Numeric
				Min		UnitName.Admin.Warning[#].AckDateTime.Min	User-defined Numeric
				Sec		UnitName.Admin.Warning[#].AckDateTime.Sec	User-defined Numeric
		WarningExtent				UnitName.Admin.WarningExtent	DINT
		ModeTimeCurrent				UnitName.Admin.ModeTimeCurrent	DINT
		StateTimeCurrent				UnitName.Admin.StateTimeCurrent	DINT
		CumulativeTimes[#]				UnitName.Admin.CumulativeTimes[#]	CUMULATIVE_TIMES Array
			AccTimeSinceReset			UnitName.Admin.CumulativeTimes[#].AccTimeSinceReset	DINT

MIN REQ						TAGNAME	DATATYPE
			ModeStateTimes[#]			UnitName.Admin.CumulativeTimes[#].ModeStateTimes[#]	MODESTATE_TIMES Array
				Mode		UnitName.Admin.CumulativeTimes[#].ModeStateTimes[#].Mode	DINT
				State[#]		UnitName.Admin.CumulativeTimes[#].ModeStateTimes[#].State[#]	DINT Array
		ProductData[#]					PRODUCT_DATA Array
			ID			UnitName.Admin.ProductData[#].ID	DINT
			Name			UnitName.Admin.ProductData[#].Name	STRING
			Unit			UnitName.Admin.ProductData[#].Unit	STRING(6)
			PrimaryQty			UnitName.Admin.ProductData[#].PrimaryQty	REAL
			ConsumedCount			UnitName.Admin.ProductData[#].ConsumedCount	DINT
x			ProcessedCount			UnitName.Admin.ProductData[#].ProcessedCount	DINT
x			DefectiveCount			UnitName.Admin.ProductData[#].DefectiveCount	DINT
			AccConsumedCount			UnitName.Admin.ProductData[#].AccConsumedCount	DINT
			AccProcessedCount			UnitName.Admin.ProductData[#].AccProcessedCount	DINT
			AccDefectiveCount			UnitName.Admin.ProductData[#].AccDefectiveCount	DINT
		MachDesignSpeed				UnitName.Admin.MachDesignSpeed	REAL
		DisabledStatesCfg[#]				UnitName.Admin.DisabledStatesCfg[#]	DWORD Array
		CurDisabledStates				UnitName.Admin.CurDisabledStates	DWORD
		EnabledModesCfg				UnitName.Admin.EnabledModesCfg	DWORD
		ModeTransitionCfg[#]				UnitName.Admin.ModeTransitionCfg[#]	DWORD Array
		PLCDateTime				UnitName.Admin.PLCDateTime	DATE_TIME
				Year		UnitName.Admin.PLCDateTime.Year	User-defined Numeric
				Month		UnitName.Admin.PLCDateTime.Month	User-defined Numeric
				Day		UnitName.Admin.PLCDateTime.Day	User-defined Numeric
				Hour		UnitName.Admin.PLCDateTime.Hour	User-defined Numeric
				Min		UnitName.Admin.PLCDateTime.Minute	User-defined Numeric
				Sec		UnitName.Admin.PLCDateTime.Sec	User-defined Numeric

Table 10 – PackTags: Minimum required for information/machine monitoring

STATUS TAGS						**TAGNAME**	**DATATYPE**
UnitName						UnitName	PACKMLv2022
	Status					UnitName.Status	PMLs
		UnitModeCurrent				UnitName.Status.UnitModeCurrent	DINT
		StateCurrent				UnitName.Status.StateCurrent	DINT
		MachSpeed				UnitName.Status.MachSpeed	REAL
		CurMachSpeed				UnitName.Status.CurMachSpeed	REAL
ADMIN TAGS						**TAGNAME**	**DATATYPE**
UnitName						UnitName	PACKMLv2022
	Admin					UnitName.Admin	PMLa
		ProductData[#].ProcessedCount	DINT			UnitName.Admin.ProductData[#].ProcessedCount	DINT
		ProductData[#].DefectiveCount	DINT			UnitName.Admin.ProductData[#].DefectiveCount	DINT
		StopReason	ID			UnitName.Admin.StopReason.ID	DINT

Table 11 – PackTags: Minimum required for supervisory control

COMMAND TAGS						TAGNAME	DATATYPE
UnitName						UnitName	PACKMLv2022
	Command					UnitName.Command	PMLc
		UnitMode				UnitName.Command.UnitMode	DINT
		UnitModeChangeRequest				UnitName.Command.UnitModeChangeRequest	BOOL
		MachSpeed				UnitName.Command.MachSpeed	REAL
		CntrlCmd				UnitName.Command.CntrlCmd	DINT
		CmdChangeRequest				UnitName.Command.CmdChangeRequest	BOOL

7.5.1 Command tags

Command tags are used to control the operation of the unit/machine. Command tags include unit state commands, which control the state transitions in the base state model. The command tags also include parameters and process variables, which control how the machine operates. Command tags generally originate from the machine user or a remote system. The originator of the command in this report is defined as the “requestor” or “information sender.” The unit/machine in this report is known as the “execution system.”

7.5.1.1 Command.UnitMode

Data Type: DINT
Tag Descriptor: Unit Mode Target

The UnitMode tag is a numerical representation of the commanded mode as defined by the user or machine builder. There can be up to 31 unit modes, and for each unit mode there is an accompanying state model. Mode 0 is invalid, and the first three unit modes are reserved, leaving up to 28 user-defined modes. Example unit modes are Production, Maintenance, Manual, Clean Out, Dry Run, Setup, etc.

0	Invalid
1	Production
2	Maintenance
3	Manual
4 – 31	User Definable

7.5.1.2 Command.UnitModeChangeRequest

Data Type: BOOL
Tag Descriptor: Request Unit Mode Change

When a unit mode change request takes place, a numerical value must be present in the Command.UnitMode tag to change the unit mode. Local processing and conditioning of the requested mode change is necessary in order to accept, reject, or condition the timing of the change request.

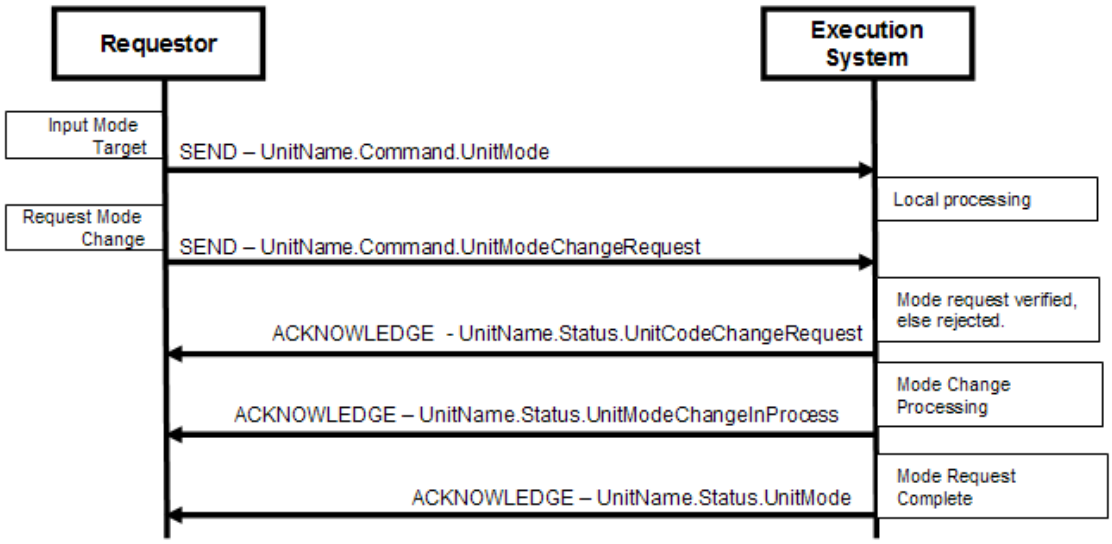

Figure 9 – Unit mode change example sequence

7.5.1.3 Command.MachSpeed

Data Type: REAL
Tag Descriptor: Current Machine Speed Setpoint
Unit of Measure: Primary Packages/Minute

This defines the set point for the current speed of the machine in primary packages per minute. Keeping speed in a primary package unit of measure (UOM) allows for easier control integration. The primary package UOM is the normalized rate for the machine, normalized to a value chosen on the line. The following example is for a bottle line running at balance line speed of 1000 packages/minute. The UOM chosen is equivalent to be the actual count of the filler, or labeler.

Machine	Actual pack counts	Primary packages (UOM)
Bulk Depalletizer	41.6666 (24 pack equiv)	1,000
Filler	1,000	1,000
Labeler	1,000	1,000
Packer	66.666 (15 packs)	1,000

7.5.1.4 Command.MaterialInterlock

Data Type: DWORD
Tag Descriptor: Materials Ready

Indicates materials are ready for processing. It is comprised of a series of bits with 1 equaling ready or not low, 0 equaling not ready, or low. Each bit represents a different user material. Materials are defined as all consumables such as product, cartons, labels, utilities, and glue. The 32-bit word contains bits that indicate when a critical material or process parameter is ready for use. It can also be used for production, and/or indication of low condition. This information may be sent to the unit/machine at any time as the interlock information changes.

Unused bits should be forced to a value of 1 or TRUE so that the entire MaterialInterlock tag may be easily evaluated as completely ready or not ready.

MATERIALINTERLOCK EXAMPLE	Raw Material #1 – Not Low	Raw Material #1 – Ready	Air Pressure – Ready	Compressed Air – Ready	Lubrication Water – Ready	Container Caps – Not Low	Container Caps – Ready	Undefined / Unused	Undefined / Unused	Undefined / Unused
Value	1	1	1	1	1	1	1	1	1	1
MaterialInterlock.[Bit #]	0	1	2	3	4	5	6	..	30	31

7.5.1.5 Command.CntrlCmd

Data Type: DINT
Tag Descriptor: Control Command
The tag holds the value of the command that provides the state command to drive a state change in the base state model, this tag is typically manipulated locally. Local processing of this tag, local machine conditions, and local commands drive changes between machine states. This tag can be set by a local or remote source. All values in the table below are reserved.

0	Undefined
1	Reset
2	Start
3	Stop
4	Hold
5	Unhold
6	Suspend
7	Unsuspend
8	Abort
9	Clear
10	Complete

7.5.1.6 Command.CmdChangeRequest

Data Type: BOOL
Tag Descriptor: State Change Request

This CmdChangeRequest bit will command the machine to proceed to change the state to the target state. The tag can be used to condition when a change of state can occur. The target state will be one of the states in the base state model.

7.5.1.7 Command.Parameter_REAL [#]

Data Type: PARAMETER_REAL Array
Tag Descriptor: Structured Array of Unit/Machine Parameter information for values with REAL data type

The Parameter tags are associated to the end user supervisory interface and are typically used for command parameters that are given to the unit by the end user's supervisory HMI. The parameters are typically needed for running the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. These parameters are typically limited to machine parameters that do not vary based on the product being produced, in contrast to the process and ingredient parameters described in later tags.

7.5.1.7.1 Command.Parameter_REAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.7.2 Command.Parameter_REAL [#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.1.7.3 Command.Parameter_REAL [#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.1.7.4 Command.Parameter_REAL [#].Value

Data Type: REAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the Parameter_REAL[#].ID, Parameter_REAL[#].Name, and the unit of measure is described by the Parameter_REAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit process variable:

Command.Parameter_REAL[1].Name = BEARING_1_OVERTEMP
Command.Parameter_REAL[1].Unit = DegC
Command.Parameter_REAL[1].Value = 350.00

This defines the temperature of a Bearing Overtemp alarm of the #1 bearing and is to be set at 350.0 degrees C for all products.

7.5.1.8 Command.Parameter_STRING[#]

Data Type: PARAMETER_STRING Array
Tag Descriptor: Structured Array of Unit/Machine Parameter information for values with STRING data type

The Parameter tags are associated to the end user supervisory interface and are typically used for command parameters that are given to the unit by the end user's supervisory HMI. The parameters are typically needed for running the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. These parameters are typically limited to machine parameters that do not vary based on the product being produced, in contrast to the process and ingredient parameters described in later tags.

7.5.1.8.1 Command.Parameter_STRING[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.8.2 Command.Parameter_STRING[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.1.8.3 Command.Parameter_STRING[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is typically not used for string values.

7.5.1.8.4 Command.Parameter_STRING[#].Value

Data Type: STRING
Tag Descriptor: Value of Parameter

This is the string value of the parameter. The value is described by the Parameter_STRING[#].ID and Parameter_STRING[#].Name commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit string parameter:

Command.Parameter_STRING[1].Name = MACHINE NAME
Command.Parameter_STRING[1].Value = 'Line 5 Case Packer'

7.5.1.9 Command.Parameter_LREAL[#]

Data Type: PARAMETER_LREAL Array
Tag Descriptor: Structured Array of Unit/Machine Parameter information for values with LREAL data type

The Parameter tags are associated to the end user supervisory interface and are typically used for command parameters that are given to the unit by the end user's supervisory HMI. The parameters are typically needed for running the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. These parameters are typically limited to machine parameters that do not vary based on the product being produced, in contrast to the process and ingredient parameters described in later tags.

7.5.1.9.1 Command.Parameter_LREAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.9.2 Command.Parameter_LREAL[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.1.9.3 Command.Parameter_LREAL[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter
The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.1.9.4 Command.Parameter_LREAL[#].Value

Data Type: LREAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the Parameter_LREAL[#].ID, Parameter_LREAL[#].Name, and the unit of measure is described by the Parameter_LREAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit process variable:

Command.Parameter_LREAL[1].Name = HOME OFFSET 1
Command.Parameter_LREAL[1].Unit = MM
Command.Parameter_LREAL[1].Value = 25.0449578

This defines the calibrated position offset for an axis on the machine in units of millimeters.

7.5.1.10 Command.Parameter_DINT[#]

Data Type: PARAMETER _DINT Array
Tag Descriptor: Structured Array of Unit/Machine Parameter Information for values with DINT data type

The Parameter tags are associated to the end user supervisory interface and are typically used for command parameters that are given to the unit by the end user's supervisory HMI. The parameters are typically needed for running the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. These parameters are typically limited to machine parameters that do not vary based on the product being produced, in contrast to the process and ingredient parameters described in later tags.

7.5.1.10.1 Command.Parameter_DINT[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.10.2 Command.Parameter_DINT[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.1.10.3 Command.Parameter_DINT[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.1.10.4 Command.Parameter_DINT[#].Value

Data Type: DINT
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the Parameter_DINT[#].ID, Parameter_DINT [#].Name, and the unit of measure is described by the Parameter_DINT [#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit process variable:

Command.Parameter_DINT [1].Name = MACHINE MODULUS
Command.Parameter_DINT [1].Unit = MM
Command.Parameter_DINT [1].Value = 848856

This defines machine cycle modulus (i.e., rollover, modulo) of a long tray conveyor system in millimeters.

7.5.1.11 Command.SelectedRecipe

Data Type: DINT
Tag Descriptor: Recipe Selection

This tag is used to designate which recipe should be run on the machine to produce the primary output product, according to a user-design recipe handling procedure. It is designed to correspond directly to the listing in the Recipe Array. A recipe changeover process can be created using the Command.RecipeChangeRequest tag.

7.5.1.12 Command.RecipeChangeRequest

Data Type: BOOL
Tag Descriptor: Recipe change request

This tag is used to trigger a user-defined recipe changeover process that will configure the machine to produce the primary product designated in Command.SelectedRecipe. The changeover process can be interlocked in the same manner as the mode change and state change processes.

7.5.1.13 Command.Recipe[#]

Data Type: RECIPE Array
Tag Descriptor: Structured Array of Recipe Information

The recipe data type can be used for defining product ingredient and product processing parameter variables. The command tags can come from either a local HMI or remote system and are used to process the product on the unit/machine. The array is typically needed for machines that run multiple products consisting of multiple ingredients.

7.5.1.13.1 Command.Recipe[#].ID

Data Type: DINT
Tag Descriptor: ID Value of the Recipe

The ID is used to indicate to the machine which recipe is being produced. The array can be used for machines that run multiple products.

7.5.1.13.2 Command.Recipe[#].Name

Data Type: STRING
Tag Descriptor: Name of Recipe

The Recipe Name is used to further identify the product being produced, especially when alphanumeric characters are used (i.e., SKU or UPC).

7.5.1.13.3 Command.Recipe[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Recipe

The Recipe Unit is used to describe the unit of production for the product being produced. An example product unit of measure may be Pc, Crtn, Case, etc.

7.5.1.13.4 Command.Recipe[#].PrimaryQty

Data Type: REAL
Tag Descriptor: Primary Quantity Value of Recipe

PrimaryQty is used to provide a common reference for manufacturing throughput across all the machines on a production line. This element describes the number of Primary Products per recipe Unit for the product being produced by this unit/machine. It can be used as a conversion factor for a local machine reference speed since Command.MachSpeed is given in units of PrimaryProducts/Min.

For example, a cartoner machine is able to make 6-pack, 12-pack, and 18-pack products. The PrimaryQty element indicates how many primary products (cans) fit in an output product unit.

Command.Recipe[1].Name = ‘6-Pack’
Command.Recipe[2].Name = ’12-Pack’
Command.Recipe[3].Name = ’18-Pack’

Command.Recipe[1].Unit = ‘Crtn’
Command.Recipe[2].Unit = ‘Crtn’
Command.Recipe[3].Unit = ‘Crtn’

Command.Recipe[1].PrimaryQty = 6.0
Command.Recipe[2].PrimaryQty = 12.0
Command.Recipe[3].PrimaryQty = 18.0

7.5.1.13.5 Command.Recipe[#].ProcessVariables

Data Type: PROCESS_VARIABLES
Tag Descriptor: Process Variables for each Recipe

The ProcessVariables structure can be used to hold particular set points needed by the unit/machine for the processing of a specific recipe. Process variables may include such things as speed and time set points, limits, and quality parameters. The extents of the included parameter arrays should be set by the user to contain the maximum number of needed process variables for any particular recipe defined on the unit/machine.

7.5.1.13.5.1 Command.Recipe[#].ProcessVariables.Parameter_REAL [#]

Data Type: PARAMETER_REAL Array
Tag Descriptor: Structured Array of Recipe Process Variable information for values with REAL data type

The Process Variable Parameter tags typically contain those machine parameters that vary based on the product being produced. The parameter value may be anything from process limit parameters to temperatures and counter presets.

7.5.1.13.5.1.1 Command.Recipe[#].ProcessVariables.Parameter_REAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.13.5.1.2 Command.Recipe[#].ProcessVariables.Parameter_REAL [#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.1.13.5.1.3 Command.Recipe[#].ProcessVariables.Parameter_REAL [#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.1.13.5.1.4 Command.Recipe[#].ProcessVariables.Parameter_REAL [#].Value

Data Type: REAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_REAL[#].ID, .Parameter_REAL[#].Name, and the unit of measure is described by the .Parameter_REAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable REAL parameter:

Command.Recipe[#].ProcessVariables.Parameter_REAL[1].Name = STROKE LENGTH
Command.Recipe[#].ProcessVariables.Parameter_REAL[1].Unit = mm
Command.Recipe[#].ProcessVariables.Parameter_REAL[1].Value = 350.0

This defines the stroke length for a particular recipe to be 350.0 mm.

7.5.1.13.5.2 Command.Recipe[#].ProcessVariables.Parameter_STRING[#]

Data Type: PARAMETER_STRING Array
Tag Descriptor: Structured Array of Recipe Process Variable information for values with STRING data type

The Process Variable Parameter tags typically contain those machine parameters that vary based on the product being produced. The parameter value may be anything from process limit parameters to temperatures and counter presets.

7.5.1.13.5.2.1 Command.Recipe[#].ProcessVariables.Parameter_STRING[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.13.5.2.2 Command.Recipe[#].ProcessVariables.Parameter_STRING[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.1.13.5.2.3 Command.Recipe[#].ProcessVariables.Parameter_STRING[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter Unit is typically not used for string values.

7.5.1.13.5.2.4 Command.Recipe[#].ProcessVariables.Parameter_STRING[#].Value

Data Type: STRING
Tag Descriptor: Value of Parameter

This is the string value of the parameter. The value is described by the Parameter_STRING[#].ID and Parameter_STRING[#].Name commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable STRING parameter:

Command.Recipe[#].ProcessVariables.Parameter_STRING[1].Name = Label SKU
Command.Recipe[#].ProcessVariables.Parameter_STRING[1].Value = ‘XK3342IJ’

7.5.1.13.5.3 Command.Recipe[#].ProcessVariables.Parameter_LREAL[#]

Data Type: PARAMETER_LREAL Array
Tag Descriptor: Structured Array of Recipe Process Variable information for values with LREAL data type

The Process Variable Parameter tags typically contain those machine parameters that vary based on the product being produced. The parameter value may be anything from process limit parameters to temperatures and counter presets.

7.5.1.13.5.3.1 Command.Recipe[#].ProcessVariables.Parameter_LREAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.13.5.3.2 Command.Recipe[#].ProcessVariables.Parameter_LREAL[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.1.13.5.3.3 Command.Recipe[#].ProcessVariables.Parameter_LREAL[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.1.13.5.3.4 Command.Recipe[#].ProcessVariables.Parameter_LREAL[#].Value

Data Type: LREAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_LREAL[#].ID, .Parameter_LREAL[#].Name, and the unit of measure is described by the .Parameter_LREAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable LREAL parameter:

Command.Recipe[#].ProcessVariables.Parameter_LREAL[1].Name = START POSITION
Command.Recipe[#].ProcessVariables.Parameter_LREAL[1].Unit = mm
Command.Recipe[#].ProcessVariables.Parameter_LREAL[1].Value = 25.0449578

This defines the starting position for an axis on the machine in units of millimeters.

7.5.1.13.5.4 Command.Recipe[#].ProcessVariables.Parameter_DINT[#]

Data Type: PARAMETER _DINT Array
Tag Descriptor: Structured Array of Recipe Process Variable Information for values with DINT data type

The Process Variable Parameter tags typically contain those machine parameters that vary based on the product being produced. The parameter value may be anything from process limit parameters to temperatures and counter presets.

7.5.1.13.5.4.1 Command.Recipe[#].ProcessVariables.Parameter_DINT[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.13.5.4.2 Command.Recipe[#].ProcessVariables.Parameter_DINT[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.1.13.5.4.3 Command.Recipe[#].ProcessVariables.Parameter_DINT[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.1.13.5.4.4 Command.Recipe[#].ProcessVariables.Parameter_DINT[#].Value

Data Type: DINT
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_DINT[#].ID, .Parameter_DINT [#].Name, and the unit of measure is described by the .Parameter_DINT [#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable DINT parameter:

Command.Recipe[#].ProcessVariables.Parameter_DINT [1].Name = MACHINE MODULUS
Command.Recipe[#].ProcessVariables.Parameter_DINT [1].Unit = mm
Command.Recipe[#].ProcessVariables.Parameter_DINT [1].Value = 848856

This defines machine cycle modulus (i.e., rollover, modulo) of a long tray conveyor system in millimeters.

7.5.1.13.6 Command.Recipe[#].Ingredients

Data Type: INGREDIENTS
Tag Descriptor: Ingredient Information for each Recipe

The Ingredients structure can be used to hold information for the raw materials that are used by the unit/machine in the processing of a particular product or recipe. Each element in one of the parameter arrays is considered to be an ingredient. The extents of the included parameter arrays should be set by the user to contain the maximum number of needed ingredients for any particular product defined on the unit/machine.

7.5.1.13.6.1 Command.Recipe[#].Ingredients.Parameter_REAL [#]

Data Type: PARAMETER_REAL Array
Tag Descriptor: Structured Array of Ingredient information for values with REAL data type

The Ingredient Parameter tags typically contain information relating to the quantities of each raw material used in the production of each resulting output product.

7.5.1.13.6.1.1 Command.Recipe[#].Ingredients.Parameter_REAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.13.6.1.2 Command.Recipe[#].Ingredients.Parameter_REAL [#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be LABEL, CHOCOLATE, other Product Name, etc.

7.5.1.13.6.1.3 Command.Recipe[#].Ingredients.Parameter_REAL [#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.1.13.6.1.4 Command.Recipe[#].Ingredients.Parameter_REAL [#].Value

Data Type: REAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_REAL[#].ID, .Parameter_REAL[#].Name, and the unit of measure is described by the .Parameter_REAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of an ingredient REAL parameter:

Command.Recipe[#].Ingredients.Parameter_REAL[1].Name = CHOCOLATE
Command.Recipe[#].Ingredients.Parameter_REAL[1].Unit = g
Command.Recipe[#].Ingredients.Parameter_REAL[1].Value = 10.0

This defines the amount of chocolate used in a particular recipe to be 10.0 g.

7.5.1.13.6.2 Command.Recipe[#].Ingredients.Parameter_STRING[#]

Data Type: PARAMETER_STRING Array
Tag Descriptor: Structured Array of Ingredient information for values with STRING data type

The Ingredient Parameter tags typically contain information relating to the quantities of each raw material used in the production of each resulting output product.

7.5.1.13.6.2.1 Command.Recipe[#].Ingredients.Parameter_STRING[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.13.6.2.2 Command.Recipe[#].Ingredients.Parameter_STRING[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.1.13.6.2.3 Command.Recipe[#].Ingredients.Parameter_STRING[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is typically not used for string values.

7.5.1.13.6.2.4 Command.Recipe[#].Ingredients.Parameter_STRING[#].Value

Data Type: STRING
Tag Descriptor: Value of Parameter

This is the string value of the parameter. The value is described by the Parameter_STRING[#].ID and Parameter_STRING[#].Name commanded by the local interface, or local processor, as a command to the unit/machine.

An example of an ingredient STRING parameter:

Command.Recipe[#].Ingredients.Parameter_STRING[1].Name = SKU
Command.Recipe[#].Ingredients.Parameter_STRING[1].Value = 'D645w455t7'

7.5.1.13.6.3 Command.Recipe[#].Ingredients.Parameter_LREAL[#]

Data Type: PARAMETER_LREAL Array
Tag Descriptor: Structured Array of Ingredient information for values with LREAL data type

The Ingredient Parameter tags typically contain information relating to the quantities of each raw material used in the production of each resulting output product.

7.5.1.13.6.3.1 Command.Recipe[#].Ingredients.Parameter_LREAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.13.6.3.2 Command.Recipe[#].Ingredients.Parameter_LREAL[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.1.13.6.3.3 Command.Recipe[#].Ingredients.Parameter_LREAL[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.1.13.6.3.4 Command.Recipe[#].Ingredients.Parameter_LREAL[#].Value

Data Type: LREAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_LREAL[#].ID, .Parameter_LREAL[#].Name, and the unit of measure is described by the .Parameter_LREAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of an ingredient LREAL parameter:

Command.Recipe[#].Ingredients.Parameter_LREAL[1].Name = GOLD LEAF WT
Command.Recipe[#].Ingredients.Parameter_LREAL[1].Unit = oz
Command.Recipe[#].Ingredients.Parameter_LREAL[1].Value = 0.00268332

This defines the amount of gold leaf to be used in the recipe as 0.00268332 oz per finished product.

7.5.1.13.6.4 Command.Recipe[#].Ingredients.Parameter_DINT[#]

Data Type: PARAMETER _DINT Array
Tag Descriptor: Structured Array of Ingredient Information for values with DINT data type

The Ingredient Parameter tags typically contain information relating to the quantities of each raw material used in the production of each resulting output product.

7.5.1.13.6.4.1 Command.Recipe[#].Ingredients.Parameter_DINT[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.1.13.6.4.2 Command.Recipe[#].Ingredients.Parameter_DINT[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.1.13.6.4.3 Command.Recipe[#].Ingredients.Parameter_DINT[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.1.13.6.4.4 Command.Recipe[#].Ingredients.Parameter_DINT[#].Value

Data Type: DINT
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_DINT[#].ID, .Parameter_DINT [#].Name, and the unit of measure is described by the .Parameter_DINT [#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable DINT parameter:

Command.Recipe[#].Ingredients.Parameter_DINT [1].Name = PKG COUNT
Command.Recipe[#].Ingredients.Parameter_DINT [1].Unit = Btls
Command.Recipe[#].Ingredients.Parameter_DINT [1].Value = 6

This defines that the finished output product should contain 6 bottles.

7.5.2 Status tags

Status tags are used to describe the operation of the unit/machine. Status tags include state commands, which describe the state transitions in the base state model. The status tags also include parameters and process variables, which describe how the machine operates. Status tags generally originate from the unit/machine user and can be used on a remote system. The originator of the status tags in this report is defined as the “execution system.”

7.5.2.1 Status.UnitModeCurrent

Data Type: DINT
Tag Descriptor: Current Unit Mode Number

This value is predefined by the user/OEM of the available unit modes of the machine allowing a possible different set of states for the base state model and could provide completely different functionality in the same machinery such as Cleanout, Production, etc.

0	Invalid
1	Production
2	Maintenance
3	Manual
4 – 31	User Definable

7.5.2.2 Status.UnitModeRequested

Data Type: BOOL
Tag Descriptor: Requested Unit Mode Change

When a unit mode request takes place, a numerical value must be present in the unit mode target to change the unit mode. Local processing and conditioning of the requested mode change is necessary in order to accept, reject, or condition the timing of the change request.

7.5.2.3 Status.UnitModeChangeInProcess

Data Type: BOOL
Tag Descriptor: Requested Unit Mode Change In Process

When a unit mode request takes place, this tag reflects the status of the state model. If the state of the machine required time to change mode, this bit would track the request and reset when the change was completed.

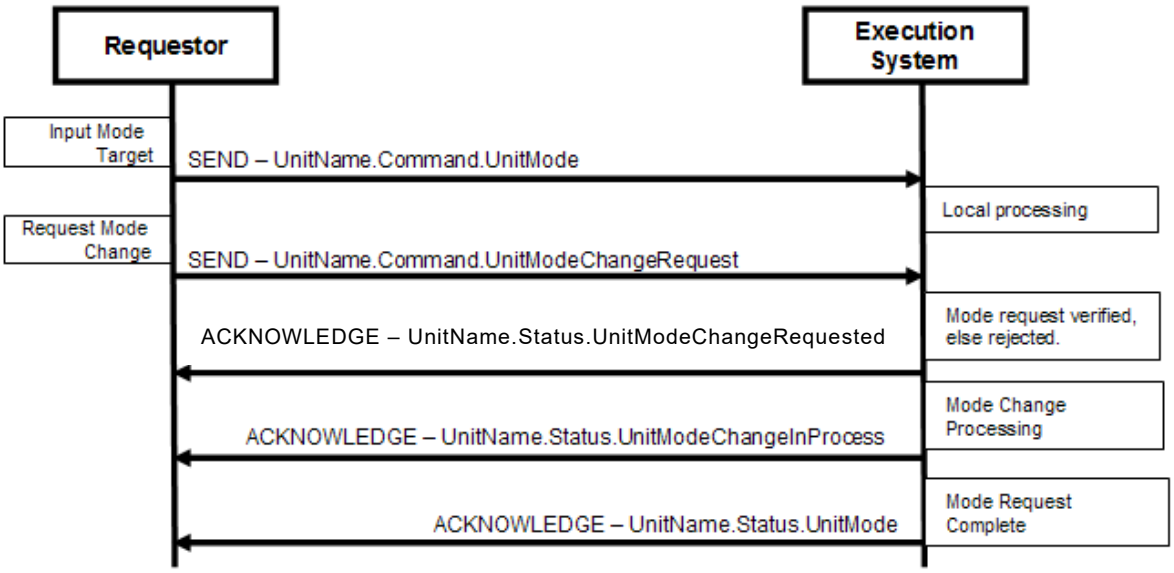

Figure 10 – Unit mode change example sequence

7.5.2.4 Status.StateCurrent

Data Type: DINT
Tag Descriptor: Current State Number

The StateCurrent status tag specifies the current state in the current unit mode of the unit/machine. The numerical values are reserved by this technical report.

0	undefined
1	CLEARING
2	STOPPED
3	STARTING
4	IDLE
5	SUSPENDED
6	EXECUTE
7	STOPPING
8	ABORTING
9	ABORTED
10	HOLDING
11	HELD
12	UNHOLDING
13	SUSPENDING
14	UNSUSPENDING
15	RESETTING
16	COMPLETING
17	COMPLETED

7.5.2.5 Status.StateRequested

Data Type: DINT
Tag Descriptor: Target State

This value is used for state transition checking, to ensure that transition to a target state can be achieved. The target state, StateRequested, is a numerical value corresponding to a state in the base state model (shown above).

7.5.2.6 Status.StateChangeInProcess

Data Type: BOOL
Tag Descriptor: State Change in Process

This bit indicates that a change in state is in progress following a state change request command.

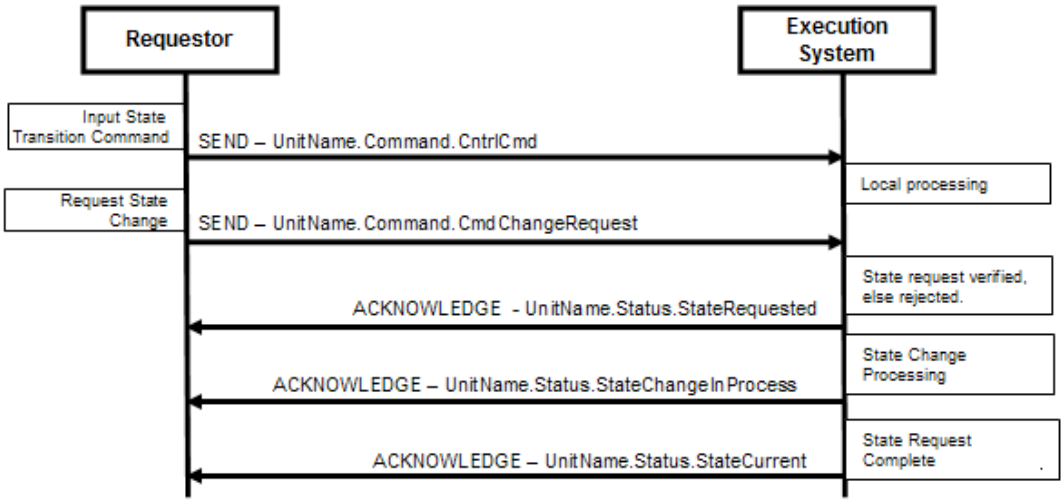

Figure 11 – State change example sequence

7.5.2.7 Status.MachSpeed

Data Type: REAL
Tag Descriptor: Current Machine Speed Setpoint
Units: Primary Packages/Minute

This describes the set point for the current speed of the machine in primary packages per minute. Keeping speed in a primary package unit of measure (UOM) allows for easier control integration. The primary package UOM is the normalized rate for the machine, normalized to a value chosen on the line. The following example is for a bottle line running at a balanced line speed of 1000 bottles/minute, where bottles is the primary package. The UOM chosen is equivalent to be the actual count of the filler, or labeler.

Machine	Actual Pack Counts	Primary packages (UOM)
Bulk Depalletizer	41.6666 (24 pack equiv)	1,000
Filler	1,000	1,000
Labeler	1,000	1,000
Packer	66.666 (15 packs)	1,000

7.5.2.8 Status.CurMachSpeed

Data Type: REAL
Tag Descriptor: Current Actual Machine Speed
Units: Primary Packages/Minute

This the actual value of the machine speed. Keeping units in primary package unit of measure (UOM), allows for easier control integration. The primary package UOM is the normalized rate for the machine, normalized to a value chosen on the line. Pack counts are parameters stored in the administration tags or downloaded parameters stored in command tags parameters.

7.5.2.9 Status.MaterialInterlock

Data Type: DWORD
Tag Descriptor: Materials Ready

MaterialInterlock describes the status of the materials that are ready for processing. It is comprised of a series of bits with 1 equaling ready or not low, 0 equaling not ready, or low. Each bit represents a different user material. Materials are defined as all consumables such as product, cartons, labels, utilities, and glue. The word contains bits that indicate when a critical material or process parameter is ready for use; it can also be used for production, and/or indication of low condition. This information is set by the unit/machine at any time as the interlock information changes.

Unused bits should be forced to a value of 1 or TRUE so that the entire MaterialInterlock tag may be easily evaluated as completely ready or not ready.

MATERIALINTERLOCK EXAMPLE	Raw Material #1 – Not Low	Raw Material #1 – Ready	Air Pressure – Ready	Compressed Air – Ready	Lubrication Water – Ready	Container Caps – Not Low	Container Caps – Ready	Undefined / Unused	Undefined / Unused	Undefined / Unused
Value	1	1	1	1	1	1	1	1	1	1
MaterialInterlock.[Bit #]	0	1	2	3	4	5	6	7	8	31

7.5.2.10 Status.EquipmentInterlock.Blocked

Data Type: BOOL
Tag Descriptor: Indicator for Unit/Machine Blocked

This bit, when set to 1, indicates that a downstream system is not able to accept product. In this condition, the equipment is capable of producing product but is in a suspended state due to a downstream system. This tag is necessary for external equipment monitoring so that the reason for the machine being in a suspended state can be identified.

7.5.2.11 Status.EquipmentInterlock.Starved

Data Type: BOOL
Tag Descriptor: Indicator for Unit/Machine Starved

This bit, when set to 1, indicates that an upstream system is not able to supply product. In this condition, the equipment is capable of producing product but is in a suspended state due to an upstream system. This tag is necessary for external equipment monitoring so that the reason for the machine being in a suspended state can be identified.

7.5.2.12 Status.Parameter_REAL[#]

Data Type: PARAMETER_REAL Array
Tag Descriptor: Structured Array of Unit/Machine Parameter Information for values with REAL data type

The Parameter tags are associated to the end user supervisory interface and are typically used for status parameters that are sent to the end user's supervisory HMI by the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. These parameters are typically limited to machine parameters that do not vary based on the product being produced, in contrast to the process and ingredient parameters described in later tags.

7.5.2.12.1 Status.Parameter_REAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.12.2 Status.Parameter_REAL[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc. This also could be displayed on HMI screens.

7.5.2.12.3 Status.Parameter_REAL[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc. This also could be displayed on HMI screens.

7.5.2.12.4 Status.Parameter_REAL[#].Value

Data Type: REAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the Parameter_REAL[#].ID, Parameter_REAL[#].Name, and the unit of measure is described by the Parameter_REAL[#].Unit on the local machine.

An example of a unit machine parameter variable:

Status.Parameter_REAL[1].Name = BEARING_1_OVERTEMP
Status.Parameter_REAL[1].Unit = DegC
Status.Parameter_REAL[1].Value = 350.00

This confirms the temperature of a bearing overtemp alarm of the #1 bearing is currently set at 350.0 degrees C for all products.

7.5.2.13 Status.Parameter_STRING[#]

Data Type: PARAMETER _STRING Array
Tag Descriptor: Structured Array of Unit/Machine Parameter Information for Values with STRING data type

The Parameter tags are associated to the end user supervisory interface and are typically used for status parameters that are sent to the end user's supervisory HMI by the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. These parameters are typically limited to machine parameters that do not vary based on the product being produced, in contrast to the process and ingredient parameters described in later tags.

7.5.2.13.1 Status.Parameter_STRING[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.13.2 Status.Parameter_STRING[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.2.13.3 Status.Parameter_STRING[#].Unit

Data Type: STRING(6)

Tag Descriptor: Unit of Measure of Parameter

The parameter unit is typically not used for string values.

7.5.2.13.4 Status.Parameter_STRING[#].Value

Data Type: STRING
Tag Descriptor: Value of Parameter

This is the string value of the parameter. The value is described by the Parameter_STRING[#].ID and Parameter_STRING[#].Name commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit string parameter:

Status.Parameter_STRING[1].Name = MACHINE NAME
Status.Parameter_STRING[1].Value = 'Line 5 Case Packer'

7.5.2.14 Status.Parameter_LREAL[#]

Data Type: PARAMETER_LREAL Array
Tag Descriptor: Structured Array of Unit/Machine Parameter information for Values with LREAL data type

The Parameter tags are associated to the end user supervisory interface and are typically used for status parameters that are sent to the end user's supervisory HMI by the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. These parameters are typically limited to machine parameters that do not vary based on the product being produced, in contrast to the process and ingredient parameters described in later tags.

7.5.2.14.1 Status.Parameter_LREAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.14.2 Status.Parameter_LREAL[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.2.14.3 Status.Parameter_LREAL[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.2.14.4 Status.Parameter_LREAL[#].Value

Data Type: LREAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the Parameter_LREAL[#].ID, Parameter_LREAL[#].Name, and the unit of measure is described by the Parameter_LREAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit parameter variable:

Status.Parameter_LREAL[1].Name = HOME OFFSET 1
Status.Parameter_LREAL[1].Unit = MM
Status.Parameter_LREAL[1].Value = 25.0449578

This defines the calibrated position offset for an axis on the machine in units of millimeters.

7.5.2.15 Status.Parameter_DINT[#]

Data Type: PARAMETER _DINT Array
Tag Descriptor: Structured Array of Unit/Machine Parameter information for values with DINT data type

The Parameter tags are associated to the end user supervisory interface and are typically used for status parameters that are sent to the end user's supervisory HMI by the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. These parameters are typically limited to machine parameters that do not vary based on the product being produced, in contrast to the process and ingredient parameters described in later tags.

7.5.2.15.1 Status.Parameter_DINT[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.15.2 Status.Parameter_DINT[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.2.15.3 Status.Parameter_DINT[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.2.15.4 Status.Parameter_DINT[#].Value

Data Type: DINT
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the Parameter_DINT[#].ID, Parameter_DINT[#].Name, and the unit of measure is described by the Parameter_DINT[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit parameter variable:

Status.Parameter_DINT[1].Name = MACHINE MODULUS
Status.Parameter_DINT[1].Unit = MM
Status.Parameter_DINT[1].Value = 848856

This defines machine cycle modulus of a long tray conveyor system in millimeters.

7.5.2.16 Status.RecipeCurrent

Data Type: DINT
Tag Descriptor: The recipe currently running in production

This tag is used to confirm which recipe is currently being produced as the primary output product. It is designed to correspond directly to the listing in the Recipe Array. A user-defined changeover process can be created using the Status.RecipeRequested and Status.RecipeChangeInProcess tags.

7.5.2.17 Status.RecipeRequested

Data Type: DINT
Tag Descriptor: A reflection of the recipe currently selected for production

This tag is used to reflect the value of Command.SelectedRecipe as part of a changeover process to a new recipe to be produced as the primary output product. It is designed to correspond directly to the listing in the Recipe Array.

7.5.2.18 Status.RecipeChangeInProcess

Data Type: BOOL
Tag Descriptor: An indicator for the user-defined recipe changeover process

This tag is used to confirm that a change to a new recipe to be produced as the primary output product is in process.

7.5.2.19 Status.Recipe[#]

Data Type: RECIPE Array
Tag Descriptor: Structured Array of Recipe Information

The recipe data type can be used for defining product ingredient and product processing parameter variables. The command tags can come from either a local HMI or remote system and are used to process the product on the unit/machine. The array is typically needed for machines that run multiple products consisting of multiple ingredients.

7.5.2.19.1 Status.Recipe[#].ID

Data Type: DINT
Tag Descriptor: ID Value of the Recipe

The ID is used to indicate to the machine which recipe is being produced. The array can be used for machines that run multiple products.

7.5.2.19.2 Status.Recipe[#].Name

Data Type: STRING
Tag Descriptor: Name of Recipe

The Recipe Name is used to further identify the product being produced, especially when alphanumeric characters are used (i.e., SKU or UPC).

7.5.2.19.3 Status.Recipe[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Recipe

The Recipe Unit is used to describe the unit of production for the product being produced. An example product unit of measure may be Pc, Crtn, Case, etc.

7.5.2.19.4 Status.Recipe[#].PrimaryQty

Data Type: REAL
Tag Descriptor: Primary Quantity Value of Recipe

PrimaryQty is used to provide a common reference for manufacturing throughput across all the machines on a production line. This element describes the number of Primary Products per recipe Unit for the product being produced by this unit/machine. It can be used as a conversion factor for a local machine reference speed since Status.MachSpeed is given in units of PrimaryProducts/Min.

For example, a cartoner machine is able to make 6-pack, 12-pack, and 18-pack products. The PrimaryQty element indicates how many primary products (cans) fit in an output product unit.

Status.Recipe[1].Name = ‘6-Pack’
Status.Recipe[2].Name = ’12-Pack’
Status.Recipe[3].Name = ’18-Pack’

Status.Recipe[1].Unit = ‘Crtn’
Status.Recipe[2].Unit = ‘Crtn’
Status.Recipe[3].Unit = ‘Crtn’

Status.Recipe[1].PrimaryQty = 6.0
Status.Recipe[2].PrimaryQty = 12.0
Status.Recipe[3].PrimaryQty = 18.0

7.5.2.19.5 Status.Recipe[#].ProcessVariables

Data Type: PROCESS_VARIABLES
Tag Descriptor: Process Variables for each Recipe

The ProcessVariables structure can be used to reflect and confirm particular set points needed by the unit/machine for the processing of a specific recipe or to report production statistics. Process variables may include such things as speed and time set points, limits, and quality metrics. The extents of the included parameter arrays should be set by the user to contain the maximum number of needed process reporting variables for any particular recipe defined on the unit/machine.

7.5.2.19.5.1 Status.Recipe[#].ProcessVariables.Parameter_REAL [#]

Data Type: PARAMETER_REAL Array
Tag Descriptor: Structured Array of Recipe Process Variable information for values with REAL data type

The Process Variable Parameter tags typically contain those machine parameters that vary based on the product being produced. The parameter value may be anything from process limit parameters to temperatures and counter values.

7.5.2.19.5.1.1 Status.Recipe[#].ProcessVariables.Parameter_REAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.19.5.1.2 Status.Recipe[#].ProcessVariables.Parameter_REAL [#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.2.19.5.1.3 Status.Recipe[#].ProcessVariables.Parameter_REAL [#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.2.19.5.1.4 Status.Recipe[#].ProcessVariables.Parameter_REAL [#].Value

Data Type: REAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_REAL[#].ID, .Parameter_REAL[#].Name, and the unit of measure is described by the .Parameter_REAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable REAL parameter:

Status.Recipe[#].ProcessVariables.Parameter_REAL[1].Name = AVG MEAS LENGTH
Status.Recipe[#].ProcessVariables.Parameter_REAL[1].Unit = mm

Status.Recipe[#].ProcessVariables.Parameter_REAL[1].Value = 351.52

This reports the average measured length for products made by a particular recipe to be 351.52 mm.

7.5.2.19.5.2 Status.Recipe[#].ProcessVariables.Parameter_STRING[#]

Data Type: PARAMETER_STRING Array
Tag Descriptor: Structured Array of Recipe Process Variable information for values with STRING data type

The Process Variable Parameter tags typically contain those machine parameters that vary based on the product being produced. The parameter value may be anything from process limit parameters to temperatures and counter values.

7.5.2.19.5.2.1 Status.Recipe[#].ProcessVariables.Parameter_STRING[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.19.5.2.2 Status.Recipe[#].ProcessVariables.Parameter_STRING[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.2.19.5.2.3 Status.Recipe[#].ProcessVariables.Parameter_STRING[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter Unit is typically not used for string values.

7.5.2.19.5.2.4 Status.Recipe[#].ProcessVariables.Parameter_STRING[#].Value

Data Type: STRING
Tag Descriptor: Value of Parameter

This is the string value of the parameter. The value is described by the Parameter_STRING[#].ID and Parameter_STRING[#].Name commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable STRING parameter:

Status.Recipe[#].ProcessVariables.Parameter_STRING[1].Name = Label SKU
Status.Recipe[#].ProcessVariables.Parameter_STRING[1].Value = ‘XK3342IJ’

7.5.2.19.5.3 Status.Recipe[#].ProcessVariables.Parameter_LREAL[#]

Data Type: PARAMETER_LREAL Array
Tag Descriptor: Structured Array of Recipe Process Variable information for values with LREAL data type

The Process Variable Parameter tags typically contain those machine parameters that vary based on the product being produced. The parameter value may be anything from process limit parameters to temperatures and counter values.

7.5.2.19.5.3.1 Status.Recipe[#].ProcessVariables.Parameter_LREAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.19.5.3.2 Status.Recipe[#].ProcessVariables.Parameter_LREAL[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.2.19.5.3.3 Status.Recipe[#].ProcessVariables.Parameter_LREAL[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.2.19.5.3.4 Status.Recipe[#].ProcessVariables.Parameter_LREAL[#].Value

Data Type: LREAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_LREAL[#].ID, .Parameter_LREAL[#].Name, and the unit of measure is described by the .Parameter_LREAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable LREAL parameter:

Status.Recipe[#].ProcessVariables.Parameter_LREAL[1].Name = START POSITION
Status.Recipe[#].ProcessVariables.Parameter_LREAL[1].Unit = mm
Status.Recipe[#].ProcessVariables.Parameter_LREAL[1].Value = 25.0449578

This defines the starting position for an axis on the machine in units of millimeters.

7.5.2.19.5.4 Status.Recipe[#].ProcessVariables.Parameter_DINT[#]

Data Type: PARAMETER _DINT Array
Tag Descriptor: Structured Array of Recipe Process Variable Information for values with DINT data type

The Process Variable Parameter tags typically contain those machine parameters that vary based on the product being produced. The parameter value may be anything from process limit parameters to temperatures and counter values.

7.5.2.19.5.4.1 Status.Recipe[#].ProcessVariables.Parameter_DINT[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.19.5.4.2 Status.Recipe[#].ProcessVariables.Parameter_DINT[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.2.19.5.4.3 Status.Recipe[#].ProcessVariables.Parameter_DINT[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.2.19.5.4.4 Status.Recipe[#].ProcessVariables.Parameter_DINT[#].Value

Data Type: DINT
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_DINT[#].ID, .Parameter_DINT [#].Name, and the unit of measure is described by the .Parameter_DINT [#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable DINT parameter:

Status.Recipe[#].ProcessVariables.Parameter_DINT [1].Name = MACHINE MODULUS
Status.Recipe[#].ProcessVariables.Parameter_DINT [1].Unit = mm
Status.Recipe[#].ProcessVariables.Parameter_DINT [1].Value = 848856

This defines machine cycle modulus (i.e., rollover, modulo) of a long tray conveyor system in millimeters.

7.5.2.19.6 Status.Recipe[#].Ingredients

Data Type: INGREDIENTS
Tag Descriptor: Ingredient Information for each Recipe

The Ingredients structure can be used to hold information for the raw materials that are used by the unit/machine in the processing of a particular product or recipe. Each element in one of the parameter arrays is considered to be an ingredient. The extents of the included parameter arrays should be set by the user to contain the maximum number of needed ingredients for any particular product defined on the unit/machine.

7.5.2.19.6.1 Status.Recipe[#].Ingredients.Parameter_REAL [#]

Data Type: PARAMETER_REAL Array
Tag Descriptor: Structured Array of Ingredient information for values with REAL data type

The Ingredient Parameter tags typically contain information relating to the quantities of each raw material used in the production of each resulting output product.

7.5.2.19.6.1.1 Status.Recipe[#].Ingredients.Parameter_REAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.19.6.1.2 Status.Recipe[#].Ingredients.Parameter_REAL [#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be LABEL, CHOCOLATE, other Product Name, etc.

7.5.2.19.6.1.3 Status.Recipe[#].Ingredients.Parameter_REAL [#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.2.19.6.1.4 Status.Recipe[#].Ingredients.Parameter_REAL [#].Value

Data Type: REAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_REAL[#].ID, .Parameter_REAL[#].Name, and the unit of measure is described by the .Parameter_REAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of an ingredient REAL parameter:

Status.Recipe[#].Ingredients.Parameter_REAL[1].Name = CHOCOLATE
Status.Recipe[#].Ingredients.Parameter_REAL[1].Unit = g
Status.Recipe[#].Ingredients.Parameter_REAL[1].Value = 10.0

This defines the amount of chocolate used in a particular recipe to be 10.0 g.

7.5.2.19.6.2 Status.Recipe[#].Ingredients.Parameter_STRING[#]

Data Type: PARAMETER_STRING Array
Tag Descriptor: Structured Array of Ingredient information for values with STRING data type

The Ingredient Parameter tags typically contain information relating to the quantities of each raw material used in the production of each resulting output product.

7.5.2.19.6.2.1 Status.Recipe[#].Ingredients.Parameter_STRING[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.19.6.2.2 Status.Recipe[#].Ingredients.Parameter_STRING[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.2.19.6.2.3 Status.Recipe[#].Ingredients.Parameter_STRING[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is typically not used for string values.

7.5.2.19.6.2.4 Status.Recipe[#].Ingredients.Parameter_STRING[#].Value

Data Type: STRING
Tag Descriptor: Value of Parameter

This is the string value of the parameter. The value is described by the Parameter_STRING[#].ID and Parameter_STRING[#].Name commanded by the local interface, or local processor, as a command to the unit/machine.

An example of an ingredient STRING parameter:

Status.Recipe[#].Ingredients.Parameter_STRING[1].Name = SKU
Status.Recipe[#].Ingredients.Parameter_STRING[1].Value = ‘D645w455t7’

7.5.2.19.6.3 Status.Recipe[#].Ingredients.Parameter_LREAL[#]

Data Type: PARAMETER_LREAL Array
Tag Descriptor: Structured Array of Ingredient information for values with LREAL data type

The Ingredient Parameter tags typically contain information relating to the quantities of each raw material used in the production of each resulting output product.

7.5.2.19.6.3.1 Status.Recipe[#].Ingredients.Parameter_LREAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.19.6.3.2 Status.Recipe[#].Ingredients.Parameter_LREAL[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.2.19.6.3.3 Status.Recipe[#].Ingredients.Parameter_LREAL[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.2.19.6.3.4 Status.Recipe[#].Ingredients.Parameter_LREAL[#].Value

Data Type: LREAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_LREAL[#].ID, .Parameter_LREAL[#].Name, and the unit of measure is described by the .Parameter_LREAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of an ingredient LREAL parameter:

Status.Recipe[#].Ingredients.Parameter_LREAL[1].Name = GOLD LEAF WT
Status.Recipe[#].Ingredients.Parameter_LREAL[1].Unit = oz
Status.Recipe[#].Ingredients.Parameter_LREAL[1].Value = 0.00268332

This defines the amount of gold leaf to be used in the recipe as 0.00268332 oz per finished product.

7.5.2.19.6.4 Status.Recipe[#].Ingredients.Parameter_DINT[#]

Data Type: PARAMETER _DINT Array
Tag Descriptor: Structured Array of Ingredient Information for values with DINT data type

The Ingredient Parameter tags typically contain information relating to the quantities of each raw material used in the production of each resulting output product.

7.5.2.19.6.4.1 Status.Recipe[#].Ingredients.Parameter_DINT[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.2.19.6.4.2 Status.Recipe[#].Ingredients.Parameter_DINT[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, SOAK TIME, STROKE LENGTH, etc.

7.5.2.19.6.4.3 Status.Recipe[#].Ingredients.Parameter_DINT[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.2.19.6.4.4 Status.Recipe[#].Ingredients.Parameter_DINT[#].Value

Data Type: DINT
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the .Parameter_DINT[#].ID, .Parameter_DINT [#].Name, and the unit of measure is described by the .Parameter_DINT [#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a recipe process variable DINT parameter:

Status.Recipe[#].Ingredients.Parameter_DINT [1].Name = PKG COUNT
Status.Recipe[#].Ingredients.Parameter_DINT [1].Unit = Btls
Status.Recipe[#].Ingredients.Parameter_DINT [1].Value = 6

This defines that the finished output product should contain 6 bottles.

7.5.2.20 Status.Stacklight[#]

Data Type: DINT Array, User-defined array size, minimum 1.
Tag Descriptor: Indicator for the current stacklight status.

This tag can be used simultaneously for reporting stacklight conditions and as control bits for physical outputs. The status of a light in the stack is associated to a particular bit location within the register and the user has the ability to define more than one stacklight. Certain bits are reserved as follows in accordance with IEC 60073 and the companion OMAC guideline for HMI and stacklight design.

Table 12 – Reserved bits for stacklight status

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

Shown in Figure 12 is an example of how the stacklight can be made to correspond to certain machine conditions and PackML States.

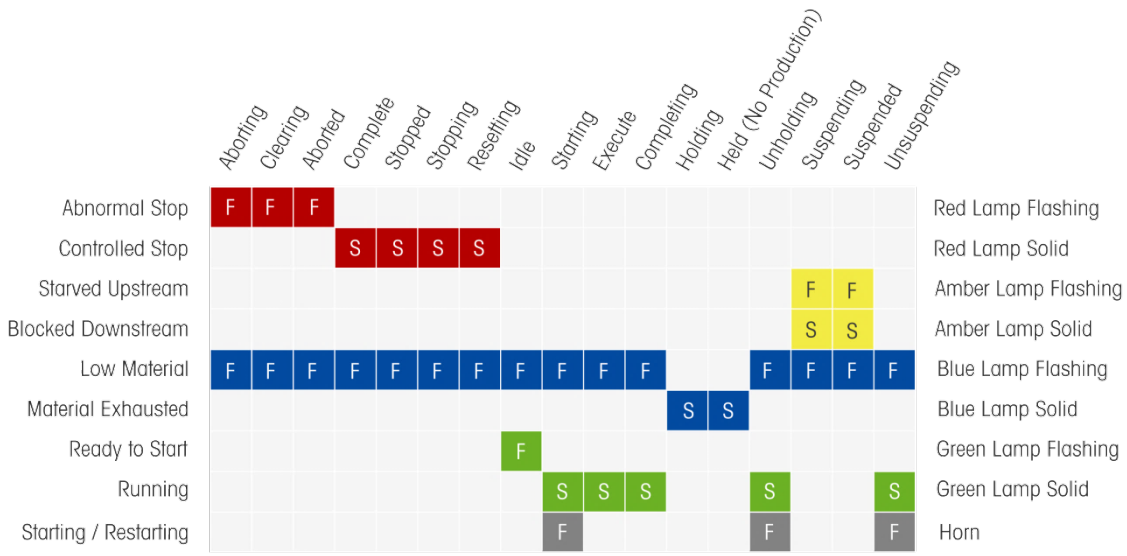

Figure 12 – Example of stacklight relationship to PackML states

7.5.3 Administration tags

Administration tags are used to describe the quality and alarm information of the unit/machine. Administration tags include alarm parameters, which describe the conditions within the base state model typically for production data acquisition (PDA) systems. The administration tags also include parameters that can describe how well the machine operates, or specific information on the product quality produced by the machine. Administration tags generally originate from the unit/machine and can be used on the HMI or a remote system.

Some administration tags support transfer of data for OEE calculations. Refer to ISO 22400 for information concerning measurement of key performance indicators (KPIs), including OEE.

7.5.3.1 Admin.Parameter_REAL[#]

Data Type: PARAMETER_REAL Array
Tag Descriptor: Structured Array of Unit/Machine Parameter Information for Values with REAL data type

The parameter tags associated to the local interface are typically used for parameters that are displayed by the end user's supervisory HMI. These parameters can be used to display any quality, alarm, or machine downtime parameter. The parameters are typically limited to

parameters related the unit. The extent of the array is the maximum number of parameters needed.

7.5.3.1.1 Admin.Parameter_REAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.3.1.2 Admin.Parameter_REAL[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The parameter name is used to describe the parameter number, and its associated value. An example parameter name may be CASES MADE, OPERATOR SHIFT, REJECTED PRODUCTS, etc. This also could be displayed on HMI screens. The array is typically needed for machines that have quality reporting or PDA (Production Data Acquisition) needs.

7.5.3.1.3 Admin.Parameter_REAL[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The administration parameter unit of measure is used to display or confirm the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be CASES, PROD, PPM, etc. This also could be displayed on HMI screens.

7.5.3.1.4 Admin.Parameter_REAL[#].Value

Data Type: REAL
Tag Descriptor: Value of Parameter

The parameter value is used to display or confirm a unit/machine variable for use in the PDA or to be sent to an “information receiver.” This also could be displayed on HMI screens.

An example of a unit machine parameter variable:

Admin.Parameter_REAL[1].Name = TOTAL PRODUCTION
Admin.Parameter_REAL[1].Unit = STAT
Admin.Parameter_REAL[1].Value = 50010.0

7.5.3.2 Admin.Parameter_STRING[#]

Data Type: PARAMETER _STRING Array
Tag Descriptor: Structured Array of Unit/Machine Parameter Information for Values with STRING data type

The Parameter tags are associated to the end user supervisory interface and are typically used for parameters that are displayed by the end user’s supervisory HMI. The parameters are typically needed for running the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. The parameters are typically limited to machine parameters as product and process parameters are described in later tags.

7.5.3.2.1 Admin.Parameter_STRING[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.3.2.2 Admin.Parameter_STRING[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.3.2.3 Admin.Parameter_STRING[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is typically not used for string values.

7.5.3.2.4 Admin.Parameter_STRING[#].Value

Data Type: STRING
Tag Descriptor: Value of Parameter

This is the string value of the parameter. The value is described by the Parameter_STRING[#].ID and Parameter_STRING[#].Name commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit string parameter:

Admin.Parameter_STRING[1].Name = MACHINE NAME
Admin.Parameter_STRING[1].Value = ‘Line 5 Case Packer’

7.5.3.3 Admin.Parameter_LREAL[#]

Data Type: PARAMETER_LREAL Array
Tag Descriptor: Structured Array of Unit/Machine Parameter information for values with LREAL data type

The Parameter tags are associated to the end user supervisory interface and are typically used for parameters that are displayed by the end user’s supervisory HMI. The parameters are typically needed for running the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. The parameters are typically limited to machine parameters as product and process parameters are described in later tags.

7.5.3.3.1 Admin.Parameter_LREAL[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.3.3.2 Admin.Parameter_LREAL[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.3.3.3 Admin.Parameter_LREAL[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.3.3.4 Admin.Parameter_LREAL[#].Value

Data Type: LREAL
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the Parameter_LREAL[#].ID, Parameter_LREAL[#].Name, and the unit of measure is described by the Parameter_LREAL[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit parameter variable:

Admin.Parameter_LREAL[1].Name = HOME OFFSET 1
Admin.Parameter_LREAL[1].Unit = MM
Admin.Parameter_LREAL[1].Value = 25.0449578

This defines the calibrated position offset for an axis on the machine in units of millimeters.

7.5.3.4 Admin.Parameter_DINT[#]

Data Type: PARAMETER _DINT Array
Tag Descriptor: Structured Array of Unit/Machine Parameter information for values with DINT data type

The Parameter tags are associated to the end user supervisory interface and are typically used for parameters that are displayed by the end user's supervisory HMI. The parameters are typically needed for running the unit/machine. The parameter value may be anything from machine limit parameters to temperatures and counter presets. The parameters are typically limited to machine parameters as product and process parameters are described in later tags.

7.5.3.4.1 Admin.Parameter_DINT[#].ID

Data Type: DINT
Tag Descriptor: ID Value of Parameter

This is the arbitrary (user defined) ID value of the parameter. This is a nondescript value that can be used for any user tag requirements.

7.5.3.4.2 Admin.Parameter_DINT[#].Name

Data Type: STRING
Tag Descriptor: Name of Parameter

The literal parameter name is used to describe the parameter number, and its associated value. An example parameter name may be GLUE TEMP, BEARING TEMP, OVERLOAD TIME, etc.

7.5.3.4.3 Admin.Parameter_DINT[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of Parameter

The parameter unit is used to describe the unit of measure of the parameter, and its associated value. An example parameter unit of measure may be DegF, secs, PPM, revs, mm, etc.

7.5.3.4.4 Admin.Parameter_DINT[#].Value

Data Type: DINT
Tag Descriptor: Value of Parameter

This is the numeric value of the parameter. The value is described by the Parameter_DINT[#].ID, Parameter_DINT[#].Name, and the unit of measure is described by the Parameter_DINT[#].Unit commanded by the local interface, or local processor, as a command to the unit/machine.

An example of a machine unit parameter variable:

Admin.Parameter_DINT[1].Name = MACHINE MODULUS
Admin.Parameter_DINT[1].Unit = MM
Admin.Parameter_DINT[1].Value = 848856

This defines machine cycle modulus of a long tray conveyor system in millimeters.

7.5.3.5 Admin.Alarm[#]

Data Type: EVENT Array
Descriptor: Array of Given Size for Machine Alarms

The alarm tags associated to the local interface are typically used as parameters that are displayed or used on the unit locally, for example from an HMI. These alarm parameters can be used to display any alarm, or machine downtime cause, that is currently occurring in the system. The alarms are typically limited to the machine unit. The extent of the array is the maximum number of alarms needed to be enunciated.

7.5.3.5.1 Admin.Alarm[#].Trigger

Data Type: BOOL
Tag Descriptor: Alarm Message Trigger

Alarm trigger should be turned on only when the alarm is currently active.

7.5.3.5.2 Admin.Alarm[#].ID

Data Type: DINT
Tag Descriptor: Alarm Message Identification Number

The alarm ID number is a unique value assigned to each alarm.

7.5.3.5.3 Admin.Alarm[#].Value

Data Type: DINT
Tag Descriptor: Alarm Message Number

The alarm message number is a value that is associated to the alarm allowing for user-specific detail or to break down the Alarm.ID to greater detail.

7.5.3.5.4 Admin.Alarm[#].Message

Data Type: STRING
Tag Descriptor: Alarm Message

The alarm message is the actual text of the alarm for those machines capable of providing string information.

7.5.3.5.5 Admin.Alarm[#].Category

Data Type: DINT
Tag Descriptor: Alarm Category

Alarm category is used to group alarms into response levels that can be used to issue state commands, display information on an HMI, or perform other actions. For example, alarms in Category 0 may be HMI Display only, alarms in Category 1 may trigger an Abort command, etc.

7.5.3.5.6 Admin.Alarm[#].DateTime

Data Type: DATE_TIME
Tag Descriptor: Date and Time the Alarm Occurred

The timestamp for when the alarm was first triggered.

7.5.3.5.7 Admin.Alarm[#].AckDateTime

Data Type: DATE_TIME
Tag Descriptor: Date and Time the Alarm was Acknowledged

The timestamp for when the alarm was acknowledged by the operator or cleared.

7.5.3.6 Admin.AlarmExtent

Data Type: DINT
Tag Descriptor: Extent of Alarm Array

The alarm extent is associated with the maximum number of alarms needed for the machine annunciation or reporting. This tag can be used by a remote machine to understand the extent of the alarm array, or locally to manage the use of the array.

7.5.3.7 Admin.AlarmHistory[#]

Data Type: EVENT Array
Tag Descriptor: Array of Given Size for Machine Fault Number and Messaging History

The AlarmHistory array is reserved for alarms that have occurred on the unit/machine and can be sorted in chronological order with the most recently occurring alarmed indexed as Admin.AlarmHistory[0]. The extent of the array is the maximum number of historical alarms needed to be retained for later viewing.

Any unused elements should be set to zero (null) values

7.5.3.7.1 Admin.AlarmHistory[#].Trigger

Data Type: BOOL
Tag Descriptor: Alarm History Message Trigger

Alarm trigger should be turned on only when the alarm is currently active. Tags such as the trigger bit are not typically used for history functionality.

7.5.3.7.2 Admin.AlarmHistory[#].ID

Data Type: DINT
Tag Descriptor: Alarm Message Identification Number

The alarm ID number is a unique value assigned to each alarm.

7.5.3.7.3 Admin.AlarmHistory[#].Value

Data Type: DINT
Tag Descriptor: Alarm Message Number

The Alarm message number is a value that is associated to the alarm allowing for user-specific detail or to break down the Alarm.ID to greater detail.

7.5.3.7.4 Admin.AlarmHistory[#].Message

Data Type: STRING
Tag Descriptor: Alarm Message

The alarm message is the actual text of the alarm for those machines capable of providing string information.

7.5.3.7.5 Admin.AlarmHistory[#].Category

Data Type: DINT
Tag Descriptor: Alarm Category

Alarm category is used to group alarms into response levels that can be used to issue state commands, display information on an HMI, or perform other actions. For example, alarms in Category 0 may be HMI Display only, alarms in Category 1 may trigger an Abort command, etc.

7.5.3.7.6 Admin.AlarmHistory[#].DateTime

Data Type: DATE_TIME
Tag Descriptor: Date and Time the Alarm Occurred

The timestamp for when the alarm was first triggered.

7.5.3.7.7 Admin.AlarmHistory[#].AckDateTime

Data Type: DATE_TIME
Tag Descriptor: Date and Time the Alarm was Acknowledged

The timestamp for when the alarm was acknowledged by the operator or cleared.

7.5.3.8 Admin.AlarmHistoryExtent

Data Type: DINT
Tag Descriptor: Extent of Alarm History Array

The alarm history extent is associated with the maximum number of alarms needed to be archived or tagged as alarm history for the machine. This tag can be used by a remote machine to understand the extent of the alarm array, or locally to manage the array.

7.5.3.9 Admin.StopReason

Data Type: EVENT
Tag Descriptor: Machine Stop Reason is typically used for “First Out Fault” Reporting and Other Stoppage Events. The stop reason is the first event captured during an abort, held, suspended, or stop event.

7.5.3.9.1 Admin.StopReason.Trigger

Data Type: BOOL
Tag Descriptor: Stop Reason Message Trigger

The stop reason trigger should be turned on only when the stop reason is currently active.

7.5.3.9.2 Admin.StopReason.ID

Data Type: DINT
Tag Descriptor: Stop Reason Message Identification Number

The stop reason ID number is a unique value assigned to each stop reason.

7.5.3.9.3 Admin.StopReason.Value

Data Type: DINT
Tag Descriptor: Stop Reason Value

The stop reason value number is a value that is associated to the stop reason allowing for user-specific detail or to break down the StopReason.ID to greater detail.

7.5.3.9.4 Admin.StopReason.Message

Data Type: STRING
Tag Descriptor: Stop Reason Message

The stop reason message is the actual text of the stop reason for those machines capable of providing string information.

7.5.3.9.5 Admin.StopReason.Category

Data Type: DINT

Tag Descriptor: Stop Reason Category

Stop Reason category is used to report the response level of the event that caused production to cease.

7.5.3.9.6 Admin.StopReason.DateTime

Data Type: DATE_TIME
Tag Descriptor: Date and Time the Stop Reason Occurred

The timestamp for when the stop reason was first triggered.

7.5.3.9.7 Admin. StopReason.AckDateTime

Data Type: DATE_TIME
Tag Descriptor: Date and Time the Stop Reason was Acknowledged

The timestamp for when the stop reason was acknowledged by the operator or cleared.

7.5.3.10 Admin.Warning[#]

Data Type: EVENT Array

Tag Descriptor: Machine warnings are for general events that do not cause the machine to stop, but may require operator action as a stoppage may be imminent. Warnings are not typically stored in history.

7.5.3.10.1 Admin.Warning[#].Trigger

Data Type: BOOL
Tag Descriptor: Warning Message Trigger

Warning trigger should be turned on only when the warning is currently active.

7.5.3.10.2 Admin.Warning[#].ID

Data Type: DINT
Tag Descriptor: Warning Message Identification Number

The warning ID number is a unique value assigned to each warning.

7.5.3.10.3 Admin.Warning[#].Value

Data Type: DINT
Tag Descriptor: Warning Value

The warning value number is a value that is associated to the warning allowing for user-specific detail or to break down the Warning.ID to greater detail.

7.5.3.10.4 Admin.Warning[#].Message

Data Type: STRING
Tag Descriptor: Warning Message

The warning message is the actual text of the warning for those machines capable of providing string information.

7.5.3.10.5 Admin.Warning[#].Category

Data Type: DINT
Tag Descriptor: Warning Category

Warning category is used to group warnings into response levels that can be used to issue state commands, display information on an HMI, or perform other actions. For example, warnings in Category 0 may be HMI Display only, warnings in Category 1 may trigger an Abort command, etc.

7.5.3.10.6 Admin.Warning[#].DateTime

Data Type: DATE_TIME
Tag Descriptor: Date and Time the Warning Occurred

The timestamp for when the warning was first triggered.

7.5.3.10.7 Admin.Warning[#].AckDateTime

Data Type: DATE_TIME
Tag Descriptor: Date and Time the Warning was Acknowledged

The timestamp for when the warning was acknowledged by the operator or cleared.

7.5.3.11 Admin.WarningExtent

Data Type: DINT
Tag Descriptor: Extent of Warning Array

The warning extent is associated with the maximum number of warnings needed to be archived or tagged as warnings for the machine. This tag can be used by a remote machine to understand the extent of the warning array, or locally to manage the array.

7.5.3.12 Admin.ModeTimeCurrent

Data Type: DINT
Tag Descriptor: Current Mode Time
Unit of Measure: Sec

This tag represents the current amount of time (in seconds) spent in the current Mode as indicated by Status.UnitModeCurrent. The value will start from 0 every time the Mode is changed. The values roll over to 0 after 2,147,483,647.

7.5.3.13 Admin.StateTimeCurrent

Data Type: DINT
Tag Descriptor: Current State Time
Unit of Measure: sec

This tag represents the current amount of time (in seconds) spent in the current State as indicated by Status.StateCurrent. The value will start from 0 every time the State is changed. The values roll over to 0 after 2,147,483,647.

7.5.3.14 Admin.CumulativeTimes[#]

Data Type: CUMULATIVE_TIMES Array, User-defined array size, minimum 1
Tag Descriptor: Structured Array of Timer Values

This tag represents a collection of accumulated time (in seconds) spent in any defined state of any defined mode. The user may define a variable number of time-tracking collections indicated by the array index and may also define when the time values in each collection are reset. The minimum array extent is 1 (Admin.CumulativeTimes[0]).

7.5.3.15 Admin.CumulativeTimes[#].AccTimeSinceReset

Data Type: DINT
Tag Descriptor: Accumulated Time Since Last Reset
Unit of Measure: Sec

The tag represents the amount of time since the reset has been triggered for a particular collection of cumulative times. When a reset for a particular collection is triggered, all times inside the structure for that collection are reset.

This value rolls over to 0 after 2,147,483,647. The tag can be used for simple OEE calculations as the definition of “scheduled production time.” The simple OEE calculation is total amount of good products divided by total amount of good products that can be produced with the unit time, with the unit of time being scheduled production time.

7.5.3.16 Admin.CumulativeTimes[#].ModeStateTimes[#]

Data Type: MODESTATE_TIMES Array
Tag Descriptor: Structured Array of Mode and State Time Values for each Cumulative Time Tracker

This tag is a collection of times for modes and states since the last timer reset was executed. The user is able to decide the extent of the array based on how many modes are defined for the system. The minimum array extent is 4 (Admin.CumulativeTimes[#].ModeStateTimes[0..3]), based on the minimum reserved modes defined in Figure 4.

7.5.3.17 Admin.CumulativeTimes[#].ModeStateTimes[#].Mode

Data Type: DINT
Tag Descriptor: Mode Time Values for each Mode
Unit of Measure: Sec

This tag represents the cumulative amount of time (in seconds) spent in each Mode since the last timer and counter reset was executed. The values roll over to 0 after 2,147,483,647.

7.5.3.18 Admin.CumulativeTimes[#].ModeStateTimes[#].State[#]

Data Type: DINT Array
Tag Descriptor: State Time Values for each State in each Mode
Unit of Measure: Sec

This tag represents the cumulative amount of time (in seconds) spent in each state of a particular Mode since the last timer and counter reset was executed. The array index represents the State number as defined in 4.3 and the extent of the array is fixed at 18 [0..17]. The values roll over to 0 after 2,147,483,647.

7.5.3.19 Admin.ProductData[#]

Data Type: PRODUCT_DATA Array
Tag Descriptor: Structured array of Product Stream Data

This tag represents a collection of information about each unique input or output product stream of the machine unit. The extent of the array is typically limited to the maximum number of input and output product streams. The array index of # = 0 can typically be reserved for data regarding the primary product stream (i.e., cans, bottles) flowing through the machine unit. In such a case, Admin.RecipeData[0].PrimaryQty would equal 1.0.

7.5.3.19.1 Admin.ProductData[#].ID

Data Type: DINT
Tag Descriptor: ID Number for each Product Stream

The product IDs are used to report product identity for each product stream. This information can also be displayed on all HMI screens. The array can be used for machines that run or consume multiple products.

7.5.3.19.2 Admin.ProductData[#].Name

Data Type: STRING
Tag Descriptor: Name of Product Stream

The Name element is used to further report the identity of the product within the product stream, especially when alphanumeric characters are used (i.e., SKU or UPC).

7.5.3.19.3 Admin.ProductData[#].Unit

Data Type: STRING(6)
Tag Descriptor: Unit of Measure of each Product Stream

The Unit element is used to report the unit of production for the product within the product stream. An example product unit of measure may be Pc, Crtn, Case, etc.

7.5.3.19.4 Admin.ProductData[#].PrimaryQty

Data Type: REAL
Tag Descriptor: Product Primary Quantity Value of each Product Stream

The PrimaryQty element is used to provide a common reference for manufacturing throughput across all the machines on a production line. This element relates the number of Primary Products per Unit for a particular product stream. It can be used as a conversion factor for a local machine reference speed since Command.MachSpeed is given in units of PrimaryProducts/Min.

For example, a cartoner machine is able to make 6-pack, 12-pack, and 18-pack products. The PrimaryQty element indicates how many primary products (cans) fit in an output product unit.

Admin.ProductData[1].Name = ‘6-Pack’
Admin.ProductData[2].Name = ’12-Pack’
Admin.ProductData[3].Name = ’18-Pack’

Admin.ProductData[1].Unit = ‘Crtn’
Admin.ProductData[2].Unit = ‘Crtn’
Admin.ProductData[3].Unit = ‘Crtn’

Admin.ProductData[1].PrimaryQty = 6.0
Admin.ProductData[2].PrimaryQty = 12.0
Admin.ProductData[3].PrimaryQty = 18.0

7.5.3.19.5 Admin.ProductData[#].ConsumedCount

Data Type: DINT
Tag Descriptor: Consumed Count of each Product Stream

This tag represents the material used/consumed in the production machine for a particular input or output product stream. An example of tag usage would be the number of bags consumed in a filler, or bagger packaging machine, or the amount of linear length used, or the number caps used. This tag can be used locally or remotely if needed.

7.5.3.19.6 Admin.ProductData[#].ProcessedCount

Data Type: DINT
Tag Descriptor: Processed Count of each Product Stream

This tag represents the number of products from a particular input or output product stream processed by the production machine. An example of tag usage would be the number of products that were made, *including* all good and defective products. This tag can be used locally or remotely if needed. The number of products processed minus the number of defective products is the number of products actually made by the machine.

7.5.3.19.7 Admin.ProductData[#].DefectiveCount

Data Type: DINT
Tag Descriptor: Defective Count of each Product Stream

This tag represents the number of products from a particular input or output product stream that is marked as defective in the production machine, to be used if applicable. An example of tag usage would be the number of products rejected or products that are termed defective. This tag can be used locally or remotely if needed. When this tag is used with Admin.ProductData[#].ProcessedCount, the number of good products/well-formed cycles made by the machine can be calculated.

7.5.3.19.8 Admin.ProductData[#].AccConsumedCount

Data Type: DINT
Tag Descriptor: Accumulated Consumption Count of each Product Stream Consumption Since Last Reset

This tag represents the total accumulated material used/consumed in the production machine for a particular input or output product stream. An example of tag usage would be the number of bags consumed in a filler, or bagger packaging machine, or the amount of linear length used, or

the number caps used. This tag can be used locally or remotely if needed. The counter rolls over to 0 after 2,147,483,647.

7.5.3.19.9 Admin.ProductData[#].AccProcessedCount

Data Type: DINT
Tag Descriptor: Accumulated Processed Count of each Product Stream Since Last Reset

This tag represents the total accumulated number of products from a particular input or output product stream processed by the production machine. An example of tag usage would be the number of products that were made, *including* all good and defective products. This tag can be used locally or remotely if needed. The number of products processed minus the number of defective products is the number of products actually made by the machine. The counter rolls over to 0 after 2,147,483,647.

7.5.3.19.10 Admin.ProductData[#].AccDefectiveCount

Data Type: DINT
Tag Descriptor: Accumulated Defective Count of each Product Stream Since Last Reset

This tag represents the total accumulated number of products from a particular input or output product stream that is marked as defective in the production machine, to be used if applicable. An example of tag usage would be the number of products rejected or products that are termed defective. This tag can be used locally or remotely if needed. The counter rolls over to 0 after 2,147,483,647.

7.5.3.20 Admin.MachDesignSpeed

Data Type: REAL
Tag Descriptor: Machine Design Speed
Unit of Measure: Primary Packages/Minute

This tag represents the maximum design speed of the machine in primary packages per minute for the package configuration being run. This speed is NOT the maximum speed as specified by the manufacturer, but rather the speed of the machine is designed to run in its installed environment. Note that in practice the maximum speed of the machine as used for efficiency calculations will be a function of how it is set up and what products it is producing.

7.5.3.21 Admin.DisabledStatesCfg[#]

Data Type: DWORD Array
Tag Descriptor: An array reflecting the Disabled States Configuration for all defined Modes.

The array index represents the Mode number. Bit locations within the DWORD value represent State numbers. A value of 1 in a bit location indicates that the corresponding state number (see table in 4.3) is disabled. State 0 is an Undefined State, therefore bit 0 is unused. Ex: If the SUSPENDED, SUSPENDING and UNSUSPENDING states were to be disabled for Mode 2, a 1 would be placed in bit locations 13, 5 and 14 of array element 2. Admin.DisabledStatesCfg[2] = 0110_0000_0010_0000 (binary), = 6020 (hex), = 24608 (decimal). See section 4.3 for Minimum Required States that should not be disabled. Note that some state transition rules may over-ride the disable bits. For example, if HOLDING is not disabled, then HELD cannot be disabled.

7.5.3.22 Admin.CurDisabledStates

Data Type: DWORD
Tag Descriptor: Reflects the disabled states for the current Mode (Status.UnitModeCurrent).

State numbers are represented by bit location. A value of 1 in a certain bit location indicates that the corresponding state number (see table in 4.3) is disabled. State 0 is an Undefined State, therefore bit 0 is unused. Ex: If the SUSPENDING, SUSPENDED and UNSUSPENDING states were disabled for the current mode, a 1 would be placed in bit locations 13, 5 and 14. Admin.CurDisabledStates = 0110_0000_0010_0000 (binary), = 6020 (hex), = 24608 (decimal). Note that some state transition rules may over-ride the disable bits set in Admin.DisabledStatesCfg[#]. For example, if HOLDING is not disabled, then HELD cannot be disabled.

7.5.3.23 Admin.EnabledModesCfg

Data Type: DWORD
Tag Descriptor: An element reflecting the Enabled Modes Configuration for the unit/machine.

Bit locations within the DWORD represent Mode numbers. A value of 1 in a bit location [0-31] indicates that the corresponding Mode number is enabled. Mode 0 is an Invalid Mode, therefore bit 0 is unused. Ex: If Modes 1, 2 and 3 are used in the Unit/machine, a 1 would be placed in bit locations 1, 2 and 3. Admin.EnabledModesCfg = 0000_0000_0000_1110 (binary), = 000E (hex), = 14 (decimal).

7.5.3.24 Admin.ModeTransitionCfg[#]

Data Type: DWORD Array
Tag Descriptor: An array reflecting the Mode Transition Configuration for all modes of the unit/machine.

The array index represents the Mode number. Bit locations within the DWORD value represent State numbers. A value of 1 in a bit location indicates that a mode transition from the corresponding state number (see table in 4.3) is allowed. Ex: For Mode 2, if transitions are to be allowed while in the Stopped, Aborted or Idle states, a 1 would be placed in bit locations 2, 9 and 4. Admin.ModeTransitionCfg[2] = 2# 0000 0010 0001 0100, = 16#214, = DINT#532.

Both modes involved in a mode transition attempt from a particular state must be configured to allow a transition to/from that state.

7.5.3.25 Admin.PLCDateTime

Data Type: DATE_TIME
Tag Descriptor: Current Date and Time of the Programmable Logic Controller

Annex A (informative) — Overview of 2022 technical report changes

#	Section/Page	Reason	Description
1	pp. 1–2	Update	The document title references were changed to ISA-TR88.00.02-2022.
2	Clause 2	Update	The Weihenstephan reference version and hyperlink were updated.
3	Foreword	Update	Added a paragraph summarizing the updates in this new version.
4	Subclause 4.4.1 c	Improve	The reference to “Suspended” has been removed. The improved state model now supports a direct transition from SUSPENDED to HOLDING.
5	Subclause 4.4.1, Figure 2	Improve	Replaced the existing state model diagram with an improved version. Added a transition from SUSPENDED to HOLDING via Hold command. Denoted by a transition boundary around EXECUTE and SUSPENDED. Replaced EXECUTE_SC transition with a new command to Complete. Added a transition from HELD, EXECUTE or SUSPENDED to COMPLETING via Complete command. Denoted by a transition boundary around HELD, EXECUTE and SUSPENDED. Renamed the COMPLETE state to COMPLETED to distinguish it from the Complete command. Updated the key for improved state backgrounds. Updated the footnotes for improved description of transition boundaries.
6	Table 3	Improve	Added a new column for the Complete command and removed the reference for EXECUTE_SC. Added new COMPLETING state references for the Complete command column in the EXECUTE, HELD AND SUSPENDED rows. Added new HOLDING state reference for the Hold command column in the SUSPENDED row. Changed the name of the COMPLETE state to COMPLETED. Reoriented the page to Landscape for improved viewing clarity. Footnote 1: Replaced the State names “Clearing” and “Resetting” with the Command names “Clear” and “Reset.”

#	Section/Page	Reason	Description
7	Subclause 7.5.1.5	Improve	Added a Command “Complete” to the tag ‘Command.CntrlCmd’, integer value 10.
8	Table 1 Table 2 Table 3	Improve	Reoriented the page to Landscape for improved viewing clarity. Changed the name of the COMPLETE state to COMPLETED.
9	Annex A		Removed the previous information regarding Weihenstephan Harmonization and replaced it with this reference table of changes for the 2022 revision.
10	Annex B		Removed the previous Annex B regarding changes for the 2015 revision.
11	Subclause 7.5.1.7.4 Table 7	Clarify	Changed data type for Command.Parameter[x].Value from ‘User-defined’ to ‘Real’.
12	Table 7 Table 8 Table 9	Clarify	Set the data type for all .Parameter[x].Unit elements to STRING(6) to clarify that this is a 6 char string and not a full 80 char string.
13	Table 7 Table 8 Table 9	Improve	Redefined .Parameter[#] datatype to Parameter_REAL[#] to better reflect that the vaule contained is of a REAL data type. Added new elements to the Command, Status and Admin Structures: .Parameter_STRING[#] .ID [DINT] .Name [STRING] .Unit [STRING(6)] .Value [STRING] .Parameter_LREAL[#] .ID [DINT] .Name [STRING] .Unit [STRING(6)] .Value [LREAL] .Parameter_DINT[#] .ID [DINT] .Name [STRING] .Unit [STRING(6)] .Value [DINT]
14	Subclause 7.4.1	Improve	Added new data type structures ‘Parameter_STRING’, ‘Parameter_LREAL’ and ‘Parameter_DINT’ Renamed ‘PackMLTagv30’ to ‘PackMLTagv2022’

#	Section/Page	Reason	Description
15	Subclause 7.5.1.8 Subclause 7.5.2.13 Subclause 7.5.3.2	Improve	Added descriptions for .Parameter_STRING[#] to the Command, Status, and Admin tag structures.
16	Subclause 7.5.1.9 Subclause 7.5.2.14 Subclause 7.5.3.3	Improve	Added descriptions for .Parameter_LREAL[#] to the Command, Status, and Admin tag structures.
17	Subclause 7.5.1.10 Subclause 7.5.2.15 Subclause 7.5.3.4	Improve	Added descriptions for .Parameter_DINT[#] to the Command, Status, and Admin tag structures.
18	Table 8		Added a new tag for Status.Stacklight[#] as an array of DINT.
19	Table 6	Clarify	Added Table 6
20	Table 7 Table 8 Subclause 7.5.1.7 (formerly) Subclause 7.5.2.12 (formerly) Subclause 7.4.1	Improve	Removed all references to the “RemoteInterface” PackTags. Removed the “Interface” structure data type.
21	Table 2	Clarify	EXECUTE state type is changed from “Acting” to “Acting and Wait.” The description is changed to clarify the example given.
22	Subclause 7.4	Clarify	Insert a table to better define the actual datatypes used in PackTags
23	Subclause 7.4.1	Improve	Added a new data type structure ‘PRODUCT_DATA Structure’.
24	Table 7 Table 8 Subclause 7.5.1.11 Subclause 7.5.2.16	Improve	Added new elements to the Command.Recipe[#] and Status.Recipe[#]structures: .Name [STRING] .Unit [STRING(6)] .PrimaryQty [REAL] This helps support the use of Command.MachSpeed in [Primary Products/Min] across all product streams.

#	Section/Page	Reason	Description
25	Table 9 Subclause 7.5.3.19	Improve	Added new tag Admin.Recipe_DATA[#] along with associated structure elements. Set the elements Admin.Recipe_DATA[#].ProcessedCount and Admin.Recipe_DATA[#].DefectiveCount to be part of the minimum set of Admin PackTags. Removed the individual tags for ProductConsumedCount[#], ProdProcessedCount[#] and ProdDefectiveCount[#]. This change was made to better align with the concept of *product streams* and to keep the information for a single product stream together.
26	Table 8 Table 9	Clarify	Status.EquipmentInterlock.Blocked and Status.EquipmentInterlock.Starved were removed from the minimum set of PackTags. Admin.ProdProcessedCount[#].Count and Admin.ProdDefectiveCount[#].Count were removed from the minimum set of PackTags Admin.Recipe[#].ProcessedCount and Admin.Recipe[#].ProcessedCount were added to the minimum set of PackTags
27	Table 9 Subclause 7.5.3.21 Subclause 7.5.3.22 Subclause 7.5.3.23 Subclause 7.5.3.24	Clarify	The tag Admin.StatesDisabled has been renamed to Admin.CurDisabledStates. The data type is changed from DINT to DWORD. The tag Admin.DisabledStatesCfg[#] has been added as a DWORD Array. The tag Admin.ModeTransitionCfg[#] has been added as a DWORD Array. The tag Admin.EnabledModesCfg has been added as a DWORD.
28	Clause 7.4.1	Clarify	Added structure breakouts to all of the structured datatypes.
29	Table 9 Subclause 7.5.3.13 through Subclause 7.5.3.18	Improve	Replaced the tags: - .Admin.StateCurrentTime[#][#] - .Admin.StateCUMULATIVE_TIMES[#][#] - .Admin.ModecurrentTime[#] - .Admin.ModeCumulativeTime[#] With: - .Admin.ModeTimeCurrent - .Admin.StateTimeCurrent - .Admin.CUMULATIVE_TIMES[#] This change was made to eliminate the 2-dimension structured array not supported on some platforms.

#	Section/Page	Reason	Description
30	Subclause 4.5.3	Clarify	Inserted a new section to clarify the rules for State Transitions
31	Subclause 6.4	Clarify	Created a new “Test” Mode as a better example of a user-defined Mode.
32	Clause 6	Clarify	New introduction section to the Mode examples
33	Subclause 6.1 Subclause 6.2 Subclause 6.3	Improve	New examples and state model diagrams were provided for the Production, Maintenance and Manual Modes. The redundant State Function figures were removed.
34	Table 7 Table 8 Subclause 7.5.1.11.6.2	Improve	A Name element was added to the Ingredient structure
35	Table 9 Subclause 7.4.1 Subclause 7.5.3.10 (formerly)	Clarify	Confirmed that the Admin.StopReason is a single structure. Since it is not an array, the tag Admin.StopReasonExtent has been removed.
36	Table 8 Table 10	Clarify	The tags Status.EquipmentInterlock.Blocked and Status.EquipmentInterlock.Starved have been removed from the minimum set of monitoring PackTags.
37	Table 9	Clarify	DateTime element [1] was missing from the listing in Admin.Alarm[#] and Admin.Warning[#]. It has now been included again.
38	Subclause 7.5.3.15	Improve	The .Admin.AccTimeSinceRest tag has been moved inside the new .Admin.CumulativeTImes[#] structure so that there is a total time tracker available for each timer in the array.
39	Subclause 7.4.1	Improve	Expanded this section to include the full element listing of all structured datatypes
40	Subclause 7.4.1	Improve	Redefined the DATE_TIME data type to be *user-definable* with certain minimum requirements and restrictions. This allows the capture and display of timestamp information to be more readily optimized across different control platforms, yet maintains
41	Clause 2	Clarify	Added reference to ISO/IEC9899 (C99) and OPC 30050
42	Introduction	Clarify	Removed the bullet point stating that one of the purposes of the TR is to provide application examples and templates. Those examples are to be provided in separate supporting documents and are outside the scope of this document.
43	Subclause 4.1 Foreword	Clarify	Relocated the 2010 revision notes from 4.1 to the Foreword section. This groups all the revision notes in the same place.

#	Section/Page	Reason	Description
44	Subclause 4.1 and throughout	Improve	Added elements to the Command tag. .SelectedRecipe .RecipeChangeRequest Added elements to the Status Tag .RecipeCurrent .RecipeRequested .RecipeChangeInProcess These additions allow the user a means to create a product changeover process using PackTags similar to the way a mode change or state change is commanded.
45	Subclause 7.4.1 and throughout	Improve	Renamed the PRODUCT data type to RECIPE and subsequently renamed the Command.Product[#] tag to Command.Recipe[#] and the Status.Product[#] tag to Status.Recipe[#]. This change better reflects the industry convention to use the term “Recipe” when selecting what a machine should produce.
46	Subclause 7.4.1 and throughout	Improve	Restructured the RECIPE data type to hold only a single instance of PROCESS_VARIABLES and INGREDIENTS data types. The arrays have been moved further inside the PROCESS_VARIABLES and INGREDIENTS data types.
47	Subclause 7.4.1 and throughout	Improve	Restructured the PROCESS_VARIABLES and INGREDIENTS data types to contain only parameter arrays. This allows process variables and ingredients to contain the full range of REAL, STRING, LREAL, and DINT data type values.

Developing and promulgating sound consensus standards, recommended practices, and technical reports is one of ISA's primary goals. To achieve this goal the Standards and Practices Department relies on the technical expertise and efforts of volunteer committee members, chairpersons, and reviewers.

ISA is an American National Standards Institute (ANSI) accredited organization. ISA administers United States Technical Advisory Groups (USTAGs) and provides secretariat support for International Electrotechnical Commission (IEC) and International Organization for Standardization (ISO) committees that develop process measurement and control standards. To obtain additional information on the Society's standards program, please write:

ISA
Attn: Standards Department
Email: standards@isa.org

ISBN: 978-1-64331-224-8