

Batch Control

Part 5: Implementation Models & Terminology for Modular Equipment Control

Working Draft 04

February 2008

This document is a draft that represents work being done by an ISA Standards Committee possibly leading to the development of an ISA Standard. ISA grants permission to anyone to reproduce and distribute copies of this ISA draft, in whole or in part, but only for the following purposes and only as long as the recipient is not charged any fee for the copy (nor may the copy be included as part of a package with other materials or presentations for which a fee is charged):

1. Review of and comment on the draft standard;
2. Provide to others for review and comment;
3. Promotion of the standard; or
4. Informing and educating others about the standard.

In addition, all copies must reproduce a copyright notice as follows:

© Copyright 2005 ISA. All rights reserved. Reproduced and distributed with permission of ISA.

ISA reserves all other rights to the draft. Any other reproduction or distribution without the prior written consent of ISA is prohibited.

The reader is cautioned that this document has not been approved and cannot be presumed to reflect the position of ISA or any other committee, society, or group. Although every effort has been made to ensure accuracy, neither ISA, members of the S&P Department, nor their employers shall be held liable for errors or limitations.



ISA-88 Draft on Batch production record

Copyright © 2008 by ISA. All rights reserved. Not for resale. Printed in the United States of America

ISA

67 Alexander Drive

P. O. Box 12277

Research Triangle Park, NC 27709 USA

Preface

This document has been prepared as part of the service of ISA - The Instrumentation, Systems, and Automation Society, 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; ISA; 67 Alexander Drive; P. O. Box 12277; Research Triangle Park, NC 27709; Telephone (919) 549-8411; Fax (919) 549-8288; E-mail: standards@isa.org.

The 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. The Department is further aware of the benefits to 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, this Department will endeavor to introduce SI-acceptable metric units in all new and revised standards, recommended practices, and technical reports to the greatest extent possible. *Standard for Use of the International System of Units (SI): The Modern Metric System*, published by the American Society for Testing & Materials as IEEE/ASTM SI 10-97, 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, recommended practices, and technical reports. 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 standard, 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 standard or a license on reasonable terms and conditions that are free from unfair discrimination.

Even if ISA is unaware of any patent covering this Standard, the user is cautioned that implementation of the standard 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 standard. ISA is not responsible for identifying all patents that may require a license before implementation of the standard or for investigating the validity or scope of any patents brought to its attention. The user should carefully investigate relevant patents before using the standard for the user's intended application.

However, ISA asks that anyone reviewing this standard who is aware of any patents that may impact implementation of the standard notify the ISA Standards and Practices Department of the patent and its owner.

Additionally, the use of this standard may involve hazardous materials, operations or equipment. The standard cannot anticipate all possible applications or address all possible safety issues associated with use in hazardous conditions. The user of this standard must exercise sound professional judgment concerning its use and applicability under the user's particular

Feltkode ændret

circumstances. The user must also consider the applicability of any governmental regulatory limitations and established safety and health practices before implementing this standard.

The scope of this Part 4 standard is structured to follow the IEC guidelines. Therefore, the first three clauses discuss the *Scope* of the standard, *Normative References*, and *Definitions*, in that order.

Clause 4 is informative. The intent of this clause is to.

Clause 5 is normative. The intent of this clause is to.

Clause 6 is normative. The intention of this clause is to state the completeness, compliance and conformance requirements for this part of the standard.

Annex A is informative. It defines the

Annex B is informative. It presents a list of frequently asked questions and answers about this standard as it is currently envisioned.

This document is intended for people who are

- responsible for defining manufacturing equipment requirements
- Responsible for the automation of manufacturing equipment

Feltkode ændret

CONTENTS

1	Scope	10
2	Normative references	11
3	Definitions	11
4	Modular equipment control concepts	13
4.1	Introduction	13
4.2	General concepts	13
4.3	Device Module	15
4.4	Control Module	16
4.4.1	Primary Control Module	16
4.4.2	Secondary Control Modules	16
4.5	Personnel Roles in Manufacturing	16
4.6	Control Strategies in Manufacturing	18
4.6.1	Strategies in a manufacturing enterprise	18
4.6.2	Manufacturing control strategies	19
4.6.3	Normal control strategy	19
4.6.4	Intervention control strategy	19
4.6.5	Maintenance control strategy	19
4.7	Process Task Strategy	20
4.8	Spectrum of control	20
4.8.1	Types of control and levels	20
4.8.2	Coordination control	21
4.8.3	Equipment principal control	21
4.8.4	Basic control	21
4.8.5	Equipment phase control	22
4.8.6	Equipment sequenced control	22
4.8.7	Equipment basic control	22
4.8.8	Equipment control	22
4.8.9	Contained and referenced control components	22
4.9	Equipment states and statuses	23
4.10	Control & operational modes	23
5	Control shells	26

Feltkode ændret

5.1	Control shell pattern -----	26
5.1.1	Control Components-----	~~32~~ <u>29</u>
5.2	Automation Module Interface-----	~~33~~ <u>30</u>
5.3	Control Shell Wrapper-----	~~33~~ <u>30</u>
5.3.1	Control Shell Wrapper Implementation Decisions -----	~~33~~ <u>30</u>
5.3.2	Control Shell Wrapper Rules -----	~~34~~ <u>31</u>
5.3.3	Resource Manager Overview-----	~~35~~ <u>32</u>
5.3.4	Supervisory Interface-----	~~36~~ <u>32</u>
5.3.5	Subordinate Interface-----	~~38~~ <u>34</u>
5.3.6	Resource Manager (RM) Supervisory Interface-----	~~42~~ <u>37</u>
5.3.7	Resource Manager (RM) Subordinate Interface -----	~~44~~ <u>39</u>
5.3.8	Resource Manager (RM) Administrative Interface-----	~~46~~ <u>40</u>
5.3.9	Function Manager Overview-----	~~47~~ <u>41</u>
5.3.10	Function Manager (FM) Interface-----	~~48~~ <u>42</u>
5.3.11	Functional Manager (FM) Outgoing Command Outputs -----	~~51~~ <u>44</u>
5.3.12	Function Manager (FM) Administrative Inputs-----	~~52~~ <u>45</u>
5.4	Functional Strategy Overview -----	~~53~~ <u>46</u>
5.5	MEC Status Overview-----	~~53~~ <u>46</u>
5.5.1	MEC Reset Command -----	~~54~~ <u>47</u>
5.6	MEC Control Shell Variable List -----	~~55~~ <u>48</u>
6	Completeness-----	~~58~~ <u>49</u>
6.1	Compliance -----	~~58~~ <u>49</u>
6.2	Conformance-----	~~58~~ <u>49</u>
6.3	Extending the object model -----	~~58~~ <u>49</u>
Annex A	Working Notes – Definitions -----	~~59~~ <u>50</u>
Annex B	Questions and Answers -----	~~66~~ <u>57</u>
Annex C	PID Example -----	~~67~~ <u>58</u>
C.1	PID Implementation-----	~~67~~ <u>58</u>
C.2	MEC PID Control Module -----	~~67~~ <u>58</u>
C.3	PID Control Element with Control Shell -----	~~68~~ <u>59</u>
C.4	MEC Analog Input Functional Strategy for PID Measured Variable Input----	~~70~~ <u>61</u>
C.5	MEC Virtual Input Functional Strategy for PID Manual Output and PID Setpoint Inputs <u>72</u> ~~63~~	

Feltkode ændret

C.6 MEC Analog Output Functional Strategy for PID Output-----	<u>7465</u>
Annex D - Cascaded PID Implementation -----	<u>7768</u>
6.4 MEC Cascaded PID Implementation -----	<u>7768</u>
Annex E– Material Transfer Model -----	<u>7869</u>
Annex ZZ - Bipolar Device Control Shell Wrapper-----	<u>9889</u>
1. Resource Manager (RM) Supervisory Interface-----	<u>9889</u>
2 Resource Manager (RM) Subordinate Interface -----	<u>9990</u>
3 Resource Manager (RM) Administrative Interface-----	<u>Fejl! Bogmærke er ikke defineret.</u>
Resource Manager (RM) Administration-----	<u>Fejl! Bogmærke er ikke defineret.</u>
4 Function Manager Incoming Commands -----	<u>10192</u>
Control Request Inputs (CRI) Commands-----	<u>10192</u>
5 Functional Manager (FM) Outgoing Command Outputs -----	<u>10393</u>
6 MEC Status Overview-----	<u>10595</u>
7 MEC Reset Command -----	<u>10696</u>
8 MEC Control Shell Variable List -----	<u>10797</u>

Feltkode ændret

LIST OF FIGURES

Figure 1 - Equipment entity hierarchy.....	15
Figure 2 Device Module	<u>Fejl! Bogmærke er ikke defineret.</u> 16
Figure 3 – Manufacturing roles.....	17
Figure 4 – Spectrum of control	21
Figure 5 - PID Function Block Example.....	24
Figure 6 - Control shell pattern.....	26
Figure 7 - Resource Manager Activate Sequence Diagram	<u>4035</u>
Figure 8 - Resource Manager Release Sequence Diagram.....	<u>4136</u>
Figure 7 - MEC PID Control Module.....	<u>6758</u>
Figure 8 – PID basic controller in MEC format	<u>6859</u>
Figure 9 - PID Control Element	<u>6960</u>
Figure 10 – PID Functional Strategy	<u>7061</u>
Figure 11 - MEC Analog Input.....	<u>7162</u>
Figure 12 - Analog Input.....	<u>7263</u>
Figure 13 – MEC Virtual Input	<u>7364</u>
Figure 14 - Virtual Input.....	<u>7465</u>
Figure 15 – MEC Analog Output	<u>7566</u>
Figure 16 - Analog Output.....	<u>7667</u>
Figure 17 Material A Storage & HBC U2.....	<u>10898</u>
Figure 18 HBC U2 Heating Control Strategy.....	<u>111401</u>

List of Tables

Table 1	<u>4338</u>
Table 2	<u>4438</u>

- 1) The formal decisions or agreements of the IEC on technical matters, prepared by technical committees on which all the National Committees having a special interest therein are represented, express, as nearly as possible, an international consensus of opinion on the subjects dealt with.
- 2) They have the form of recommendations for international use and they are accepted by the National Committees in that sense.
- 3) In order to promote international unification, the IEC expresses the wish that all National Committees should adopt the text of the IEC recommendation for their national rules in so far as national conditions will permit. Any divergence between the IEC recommendation and the corresponding national rules should, as far as possible, be clearly indicated in the latter.
- 4) The IEC has not laid down any procedure concerning marking as an indication of approval and has no responsibility when an item of equipment is declared to comply with one of its recommendations.

Feltkode ændret

Introduction

ANSI-ISA-88.00.01-1995 entitled *Part 1: Models and Terminology* (referred to as Part 1 throughout this standard) provides abstract models and terminology applicable to batch control. Part 1 Clause 5 *Batch Control Concepts* defines types and hierarchies of control functionality and allocation, modes, and states of equipment and controls. Clauses 6.6 *Unit Supervision* and 6.7 *Process Control* describe the control activities that must be implemented at the equipment control level. Although the Part 1 standard provides some hierarchical models for equipment control, it does not provide sufficient guidance to enable consistent modular implementation of many key common performance requirements cited elsewhere in the document.

ANSI/ISA-88.00.02-2001 entitled *Part 2: Data Structures and Guidelines for Languages* (referred to as Part 2 throughout this standard) specifies visual and data representations of detailed recipe information, the content of which interacts with and determines the scope and reusability of equipment control element design. ANSI/ISA-88.00.03-2003 entitled *Part 3: General and Site Recipe Models and Representation* (referred to as Part 3 throughout this standard) provides some insights on reusability impact, but in relation to recipe mappings.

ANSI/ISA-88.00.04-2006 entitled *Part 4: Batch Production Records* (referred to as Part 4 throughout this standard) defines a reference model for batch production records, the content of which may place certain design requirements on equipment control elements.

This Part 5 standard defines implementation models and terminology for modular equipment control in a consistent modular fashion based on the Part 1 equipment control concepts. The intent of this standard is to provide a clear hierarchical structure for defining and implementing the control strategies that execute the process task strategy that occurs in equipment modules and control modules.

This will provide a consistent framework for a modular and distributed design of these functions with predictable interactions. This part includes an extensible core set of exposed interface functions and protocols to minimize integration issues. The models in this part cover encapsulation of basic control functionality, procedural control functionality (process task strategies), and coordination control among different equipment entities and external systems.

This standard is consistent with the other parts of the ANSI/ISA88 standard, particularly building upon Part 1. Although this series of standards is intended primarily for batch processes, adoption of its models may provide value in other manufacturing domains.

Chairman's Note: The stretch goal is to define methods and public domain specifications for the development of a library of automation components consistent with the ISA 88 Part 1 models that can be supported by automation vendors across all types of manufacturing. The components provide a common base of commonly used automation functions that encourages modularity and defines common methods for components to interact.

1 Scope

This part defines a reference model for module equipment control within equipment modules and control modules based on the equipment control concepts described in Part 1 of this standard.

The reference model applies to equipment that may be used in batch processes and on other types of production and packaging processes. The reference model defines:

1. a template for definition of modular reusable components that defines a standard terminology,
2. a template for command and control functionality,
3. a method to describe and identify each modular component,

Feltkode ændret

4. a method for exchanging component definitions, and
5. a method for intercommunications among components.

The reference model in this part may also apply to higher level equipment entities (process cells and units) but these are not explicitly covered.

Chairman's Note: The "4. method for exchanging component definitions" may be removed from the scope if this turns out to take too much time for the initial release.

2 Normative references

The following normative documents contain provisions, which through reference in this text, constitute provisions of this part of this standard. At the time of publication, the editions indicated were valid. All normative documents are subject to revision, and parties to agreements based on this part of this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below.

- IEC 60902:1987, Industrial-process measurement and control: Terms and definitions
- IEC 61512:1997, Batch control- Part 1: Models and terminology
- IEC 62264-1:2000, Enterprise/control system integration – Part 1: Models and terminology
- ANSI/ISA-88.01-1995, Batch control- Part 1: Models and terminology
- ANSI/ISA-88.00.02-2001, Batch control - Part 2: Data Structures and Guidelines for Languages.
- ANSI/ISA-95.00.01-2000, Enterprise-Control System Integration – Part 1: Models and Terminology
- ANSI/ISA-95.00.02-2001, Enterprise-Control System Integration – Part 2: Object Model Attributes
- ISO/IEC 19501-1, Information technology – Unified Modeling Language (UML) – Part 1: Specification
- IEC 61131-3, Programmable Controllers – Part 3: Programming languages
- IEC 61499-1, Function Blocks – Part 1 Architecture

3 Definitions

For the purposes of this part of this draft, the following definitions apply. Definitions and concepts expressed in the Part 1 standard apply, except where differences are explicitly stated in this part.

3.1

control strategy

...

Note: ...

Feltkode ændret

Feltkode ændret

4 Modular equipment control concepts

4.1 Introduction

This part defines a template (or pattern) to combine function blocks, as defined IEC 61131-3 and IEC 61499, in a way that supports dynamic coordination of multiple controlling entities and control strategies without requiring recoding or reengineering.

This part describes models and terminology used in automated equipment control for equipment modules and control modules to successfully carry out a Process Task Strategy as defined by the Part 1¹.

The equipment hierarchy defined in Part 1 of this standard is made up of process cells which are made up of units, units are made up of equipment modules and control modules, equipment modules can be made up of other equipment modules and control modules, control modules may be made up of multiple control modules and control modules provide the only interface to the physical equipment. Part 1 is a conceptual and abstract model that can support manual operations as well as provide a guide to some possible automation concepts and approaches. As manufacturing disciplines other than batch address the need to create more efficient methods to automate, adapting the concepts put forward by this series of standards to the needs of these disciplines will provide similar benefits.

Part 1 of the S88 series provided guidance on how to separate the complex procedures required in batch control from the control of equipment. Part 1 provides details of “what to do” but leaves much of the “how to do” when delivering automated or hybrid control systems up to the designers. This often leads to inconsistent solutions that are incompatible with other similar systems. The concepts, models and terminology in this part provide guidelines on the “how to do it” component of equipment control so that applications will be consistent and allow easier integration.

4.2 General concepts used in this part

There are general concepts that provide background and understanding for the approaches defined in this part. Many of these concepts are beyond the scope of the standard, but are defined here to place the models in context.

- Multiple strategies exist within an enterprise. The focus on this part is on automation control strategies.
- Functional roles in manufacturing are discussed as a background for understanding the automation environment.
- A spectrum of control is defined, where the spectrum defines a sequence of related control methods that span process cell procedural control through to equipment basic control, as shown in [Figure 3](#)~~Figure 4~~.
- The unique logic or automation code that implements *Equipment Control*² is defined as a Control Component. The control components implement the various types of control

¹ This is defined in the 2007-2008 update to the ANSI/ISA88.01 standard, with an expected release in 2008.

within an equipment entity, including resource management control, functional strategy control, and functional management control.

Note 1 There may be additional Control Components that implement other facets of control, such as process cell recipe execution (recipe interpretation), unit recipe execution, equipment recipe procedural execution, and process cell coordination, but these are not defined in this part. However, these additional components could use the models defined in this part.

- The Control Component is made up of function blocks and a Control Shell.
- An Elemental Component is the lowest level to which something can be reduced and still be functional. The scope of an Elemental Component is defined by the user and may vary between applications.

Example 1 a block valve with feedback, there are the elements of the valve body, proximity sensors, and an actuator. Bringing these elements together create a compound of a block valve.

Example 2 The block valve in Example 1 is treated as a single Elemental component.

- A Compound Component consists of two or more Components that can be commanded by another entity from outside of the Compound Component.

Note 1 It is assumed that a Control Component can have one entity at a time that can provide commands, this is part of coordination control provided by the Mode of Control concepts.

- A Device is a physical item and has no control associated with it. When control is associated with a device it becomes a Control Module.
- Control Components may be compound either through a containment structure (one component encapsulates another) or through a reference structure (one component references another component). Compound Control Components implement the hierarchy of control defined in Part 1.
- Equipment Entities may have multiple Control Modes. A Control Mode defines what functions can be performed on and by the equipment when it is in the mode.

Example 1 Sanitization Control Mode – The equipment entity is only able to execute the sanitization procedure.

Example 2 Production Control Mode – The equipment entity is only able to execute the production procedure.

Example 3 Maintenance Control Mode – All equipment procedural control (implementing recipe procedures) is suspended, all referenced components can be controlled by maintenance personnel, and interlocks may be manually disabled.

Note 2 The Part 1 term *Equipment Entity Mode* (auto, manual, and semi-auto) applies to an individual control component and is different from the Control Mode of the equipment entity.

- State Models for all equipment entities: *Note: This may be removed after the major work is complete and we see if this is still needed.*
- A protocol and transaction definitions are provided for coordination across Control Components.
- Figure 1 illustrates the equipment hierarchy defined in Part 1. The scope of the models this part is shown as the highlighted area.

² As defined in ANSI/ISA88.01 - *The equipment-specific functionality that provides the actual control capability for an equipment entity, including procedural, basic, and coordination control, and that is not part of the recipe.*

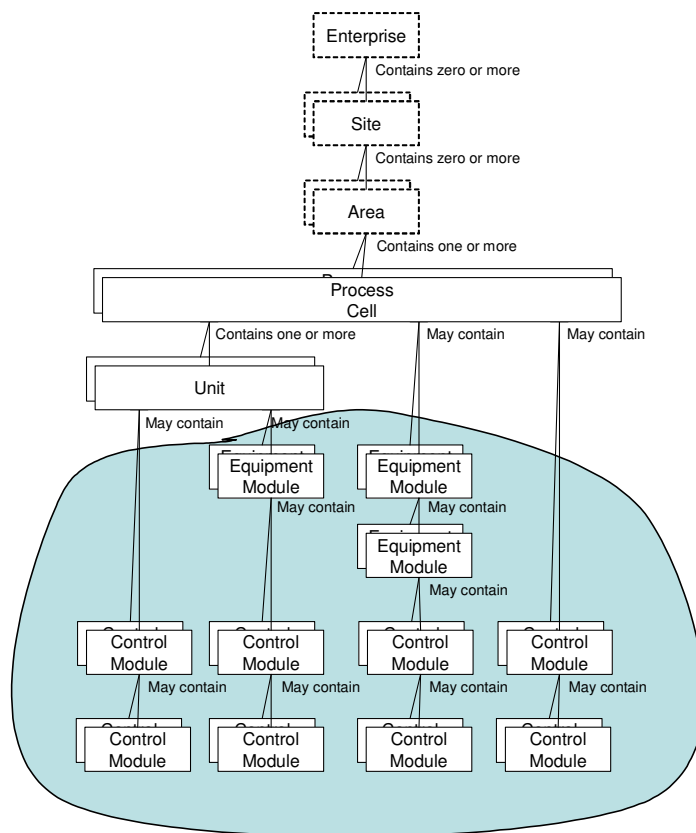


Figure 1 - Equipment entity hierarchy

4.3 Device

The assumption that the final interface with equipment is bounded by the Control Module has led to confusion. To help reduce this area of confusion this part is including the concept of the physical Device.

Devices can be simple or complex. A simple device generally has a single function, such as a proximity sensor, or an actuator. A complex device is generally made up of several simple devices, such as a block valve that has an actuator, a valve body, and possibly a proximity sensor(s) to determine position of the valve.

A Device is a physical thing and contains no control. When basic control is applied to a device it is done through the actions of a Control Module. A device is often constructed of several devices that together enable a processing activity.

Looking at a block valve with feedback, there are the elements of the valve body, proximity sensors, and an actuator. Bringing these elements together we create a compound of a block valve. The valve itself is a compound device made up of the three elemental devices. There is no control yet, so the block valve can't do anything.

Feltkode ændret

4.4 Control Module

Control Modules must ultimately combine with a device. It may do so through other Control Modules, or be the first point of contact between the logical control realm and the physical world. There can be Control Modules that process information from a sensing device, such as a proximity sensor in a block valve, that will apply the control necessary to de-bounce the

4.4.1 Elemental Control Module

4.4.2 Compound Control Module

4.5 Personnel Roles in Manufacturing

In order to emphasize the differences in perspective, content and needed skills for development of different levels of control, 3 different roles of automation developers have been defined, as shown in [Figure 2](#)~~Figure 3~~.

Feltkode ændret

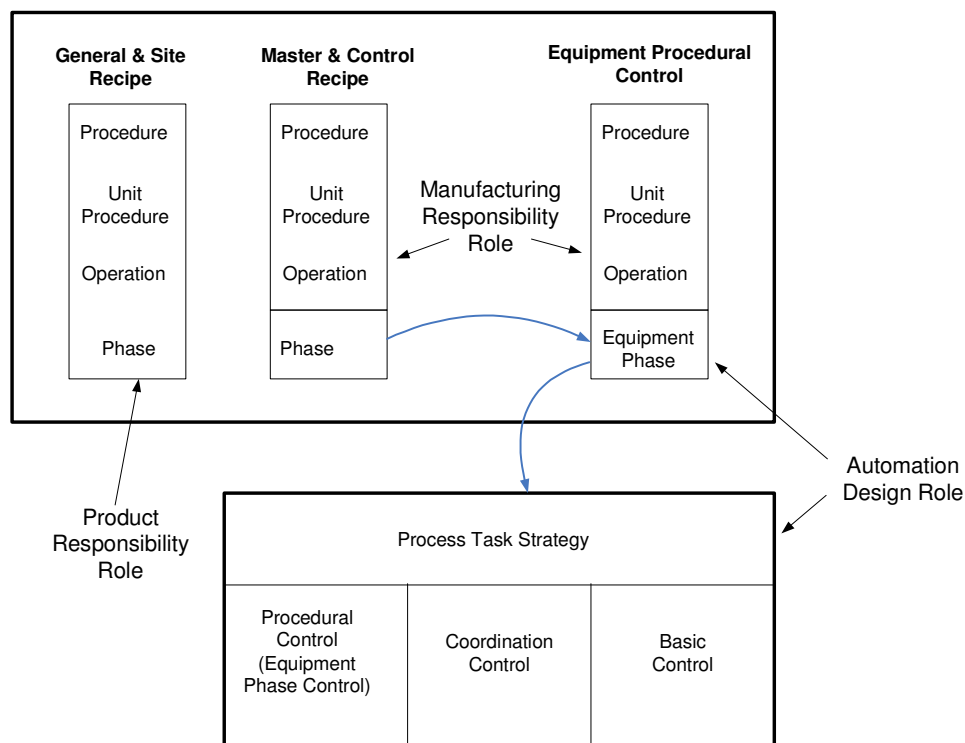


Figure 2 – Manufacturing roles

1. Product Responsibility Role

This role requires detailed knowledge about the product itself and how in general it is produced.

The role requires familiarity with the product BOM's, formulas and general procedures for producing the product, but not necessarily with the actual equipment that is going to be used.

This person will be responsible for General Recipes and Site Recipes.

Example 1 Titles may be Product Specialist or Corporate Brew Master.

2. Manufacturing Responsibility Role

This role requires manufacturing responsibility and requires detailed manufacturing process and manufacturing equipment knowledge.

This role includes translation of the product requirements in the General and Site Recipes into specific manufacturing requirements based on the existing production equipment.

This role involves the specification of the general manufacturing process functions of phases and the detailed order in which they are to be executed, as defined through Master Recipes and Equipment Procedural Control.

Example 2 Titles may be Process Engineer, Production Manager, Plant Brew Master

Feltkode ændret

3. Automation Design Role

This role requires automation design responsibility and requires detailed knowledge about the specific production equipment, its automation devices, and how they interact at the Process Control activity level.

This role includes development of all equipment specific controls, down to individual devices. This includes: Equipment Phase parameterization as well as Basic Control and Coordination Control components of the Process Task Strategy.

In cases where higher level procedural control (Operations or Unit Procedures) are embedded with equipment control, these requirements would be developed by the person with the Manufacturing responsibility role and the content should remain equipment independent.

Example 3 Titles may be Automation Engineer, Automation Specialist, Process Control Engineer.

The primary target of this Part is a person performing the automation design role.

The definition of roles is not meant to specify job titles. All roles may be filled by the same person but it is important to keep separate the different perspectives and to maintain the different details of the product and manufacturing knowledge.

Additional roles could be Production Responsibility and Maintenance Responsibility. These would have the task of running and maintaining the system – not developing or changing it. There may be other roles found in a manufacturing enterprise not listed here.

4.6 Control Strategies in Manufacturing

4.6.1 Strategies in a manufacturing enterprise

All business endeavors have strategies to guide their efforts. This is an often overused term and depending upon the group using it will have significant differences from other groups so it is important to understand who and what the strategy is defining.

An enterprise may have business strategies, product strategies, manufacturing strategies, operational strategies, manufacturing control strategies, and dozens of other strategies at various levels of the enterprise.

Example Strategies may include:

Business Strategy – Develop and deliver products using optimized manufacturing to meet local market needs.

Product Strategy – Product shall be developed by the corporate central offices and shall be described using general recipe concepts with procedure and formula in the form of PFC and all required manufacturing methods identified and the sequences needed to create the final product. Product recipes shall be provided to manufacturing sites which will adapt them to the local market and manufacturing needs and capabilities. Finished products will meet the supplied product specifications.

Manufacturing Strategy – Use batch and/or continuous systems based on production quantities required and locate within a maximum 500 kilometers of any customer requiring 100 Kiloton per year of product. Production rates greater than 1 Kiloton per day should consider continuous systems, less than 1 Kiloton should consider batch systems.

Manufacturing Operating Strategy – Minimize human labor in developed areas through automation of manufacturing. In undeveloped areas automate only those manufacturing operations that are required to maintain quality of the product. In all cases the operations will be conducted the same whether by automation or by human action.

Feltkode ændret

Manufacturing Control Strategy – ANSI/ISA88 principles will be followed. Products specific procedures will be specified using a Master Recipe with Procedure Function Charts (PFC). The PFC will be used to create a Control Recipe that will be followed to direct the equipment to carry out the Process Task Strategy to deliver the expected result of the recipe.

This part focuses only on Manufacturing Control Strategies.

4.6.2 Manufacturing control strategies

There may be multiple manufacturing control strategies within an equipment entity. There is a Normal Control Strategy that is used when production is proceeding in the normal state. When the process enters an undesired state from which the normal control strategy can not recover, then other control strategies are required. These other strategies may enable humans or other automated systems to intercede and deal quickly and effectively with unforeseen upsets or events.

Note Environmental and personal safety are outside the scope of the strategies motioned here and must be covered by the safety organizations. Product and Equipment safety is addressed.

A manufacturing control strategy uses the Control Modes of the Control Components to achieve its purpose.

4.6.3 Normal control strategy

There shall be a normal control strategy that defines how the equipment is required to operate for normal production. This is the strategy developed by the equipment, process and automation designers to deal with all known control needs to make the required products. Mixing of human actions and automated actions to carry out the desired control strategy may be required.

The normal control strategy is the strategy that should be active the majority of the time.

4.6.4 Intervention control strategy

There shall be an intervention control strategy that enables operations to suspend the normal strategy and defines how the equipment operates under the direction of production personnel.

Equipment and product protection shall be included with the Intervention Control Strategy.

This strategy is required when the equipment is not responding as expected by the normal control strategy. The Intervention Control Strategy allows operations to direct the equipment in ways not anticipated by the normal control strategy. This enables operations to bring the equipment back into a state in which the normal strategy can resume control.

Note 1 This control strategy is often overlooked because of a focus on the normal control strategy.

Note 2 An often used alternate method of taking direct control of the actual equipment is not the best approach because it does not provide the equipment and product adequate protection and does not provide operations with appropriate overview and guidance that should be included in an Intervention Control Strategy.

Note 3 Company policies may determine which personnel have the authority and rights to start the Intervention Control Strategy.

4.6.5 Maintenance control strategy

There shall be a maintenance control strategy that enables authorized personnel to suspend the other strategies for purposes of maintaining equipment. In the maintenance control strategy all equipment procedural control (implementing recipe procedures) is suspended, all referenced

Feltkode ændret

components can be controlled by maintenance personnel, and interlocks may be manually disabled.

The maintenance control strategy should allow direct personnel access to Control Components. In this strategy, equipment and product protection may be bypassed and it will be the responsibility of authorized personnel to ensure that equipment and product are protected as needed.

Note Company policies may determine which personnel have the authority and rights to start the Maintenance Control Strategy.

4.7 Process Task Strategy

Editor's note: Because this uses the word "strategy", there may be the need for explanation of how this relates to the control strategies defined above.

4.8 Spectrum of control

4.8.1 Types of control and levels

Part 1 of this standard describes procedural, basic and coordination as the three types of control required to manage a batch process. These types of control are implemented in various levels of the equipment hierarchy. In addition different types of control are implemented at different levels.

The combination of these can be represented as a spectrum of control that combines the control types and equipment hierarchy levels, as illustrated in [Figure 3](#)~~Figure 4~~. Different roles have responsibility for different parts of the spectrum of control.

Feltkode ændret

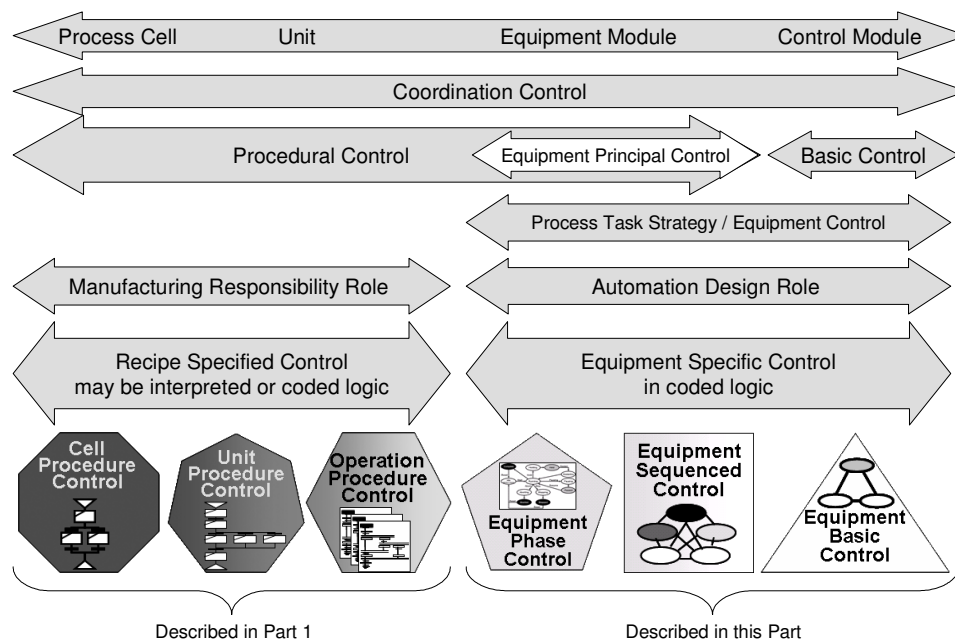


Figure 3 – Spectrum of control

4.8.2 Coordination control

Coordination control is the type of control in equipment modules and control modules that handles the:

1. coordination of a module's component parts;
2. propagation of modes to contained and referenced control components;
3. propagation of Control Modes to contained and referenced control components; and
4. arbitrating requests for usage.

4.8.3 Equipment principal control

Equipment principal control is the type of procedural control in equipment modules that implements a sequence of control steps and does not use the Part 1 procedural model structure (procedure, unit procedure, operation and phase). Equipment principal control is implemented in *Equipment Phase Control* and *Equipment Sequenced Control*.

4.8.4 Basic control

Basic control is the control that is dedicated to establishing and maintaining a specific state of equipment or process condition. It may include regulatory control, interlocking, monitoring, exception handling, and discrete or sequential control. Basic control is implemented in the Functional Strategy within a Control Module.

Feltkode ændret

4.8.5 Equipment phase control

Equipment Phase Control is a type of control component that handles the exposing of a phase interface and the interfacing of the phase commands and controls to other control components.

This control component is primarily coordination control and provides the interface to the Procedure Phase or Operation, depending on how these layers are implemented.

Note IEC 61131-3 documents methods of defining sequenced control, such as Sequential Function Charts (SFCs), this is commonly used to implement equipment phase control.

The state model for equipment phase control is the procedure state model defined in Part 1.

4.8.6 Equipment sequenced control

Equipment Sequenced Control is a type of control component that is responsible for the sequential set of equipment oriented actions required to implement equipment principal control.

Equipment Sequenced Control is sequential in nature and has quiescent state(s) which typically require an external command to become active or to move from state to state.

Note IEC 61131-3 documents methods of defining sequenced control, such as Sequential Function Charts (SFCs), this is commonly used to implement equipment sequenced control.

The state model for equipment sequenced control is function specific and is generally different from the procedure state model defined in Part 1.

4.8.7 Equipment basic control

Equipment Basic Control is a type of control component that is responsible for implementing basic control, any associated coordination control, and interfacing to the physical equipment (device(s)).

Equipment Basic Control is persistent and always active and performs its function based on inputs from the equipment/process, higher levels of control or humans.

4.8.8 Equipment control

Equipment control is the logic or code that implements the *Process Task Strategy* (as defined in Part 1).

There are three types of Control Components that make up equipment control; Equipment Phase Control, Equipment Sequence Control, and Equipment Basic Control.

Equipment control can be very simple, consisting only of a single Equipment Phase Control entity that manages a single Basic Control Entity, to quite complex structures with many interactive control components using all types of control.

4.8.9 Contained and referenced control components

Control components may use the services of other control components to accomplish its functional strategy. The use of the services may be through a containment structure or a reference structure.

Example An equipment module may use the services of multiple control modules.

Feltkode ændret

A containment structure is when a control component encapsulates one or more control components, and normal access to the encapsulated component is through the parent component.

A reference structure is when a control component does not encapsulate another control component but uses the services of other control components, and access to all of the control components is the same.

An implementation may use containment, reference, or a combination. The template defined in this Part makes no assumptions on the structure used and is equally valid for all choices.

Note An implementation of containment or referenced control components may be dependent on the functionality of the controller and programming environment.

4.9 Equipment states and statuses

Editor's note: Dan Seger has responsibility for this section.

4.10 Control & operational modes

Part 5 defines two classes of "mode" that can be applied to the Equipment Phase Control, Equipment Sequence Control and Equipment Basic Control, which are used to carry out a S88 Process Task Strategy.

Class 1 Mode of Action

The first class is how the internal activity of a single Control Component will operate and the algorithm's relation to the outputs it directly manages. This is the Mode of Action. Each Control Component will have at least one Mode of Action.

The internal activity of any control component will generally have at least the two classic modes described by S88 of Automatic and Manual. In the example of a PID function block, shown in [Figure 4Figure 5](#), when in a mode of Automatic the PID algorithm will be directly managing the Output. When the PID function block is in Manual the algorithm is no longer managing the Output which is now managed by something outside of the function block, sometimes an operator through some form of Human Machine Interface (HMI). Two examples of an HMI are digital values on a graphic display and a physical rheostat on a panel. Other Control Components might also be able to manipulate the output through this method.

Feltkode ændret

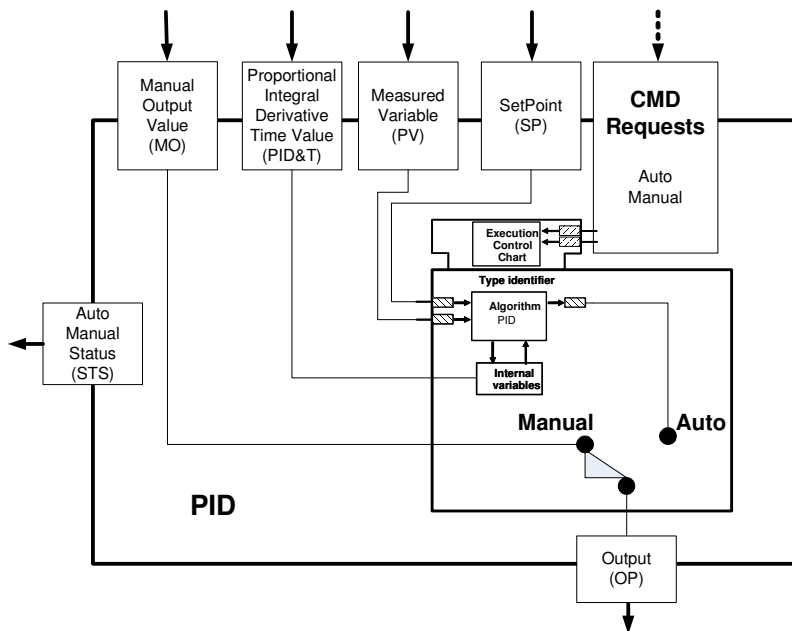


Figure 4 - PID Function Block Example with Auto/Manual

It is possible for a Control Component to have more than one Mode that describes its behavior, these modes can be complementary to one another and can be active at the same time. Besides Auto/Manual some other possible modes for Class 1 modes are: Simulation/Not-Simulation, Active/Inactive and any others that might be required. The Control Components themselves will generally manage this internal mode and might do so based upon the overall Mode of Control that this Control Component is part of.

If the Control Component is sufficiently complex and capable of executing complex sequences or procedures and will be manipulation the inputs of other Control Components being able to manage the internal execution of the sequence or procedure becomes important. In this situation the Mode of Action would follow the S88 Procedural Modes of Automatic, Semi-Automatic and Manual.

Note: need to develop a diagram of a “sequence” and “phase” control component

Class 2 Mode of Control

The second class is how multiple Control Components operate & interact together to carry out a more complex activity than is reasonable in a single control Component. This is the Mode of Control. Each grouping of Control Components will have a Mode of Control that is descriptive of what that grouping is designed to accomplish.

When two or more Control Components are grouped together to carry out a function it is often beneficial to allow these grouping to be represented by the concept of a Mode that describes what it is that grouping is designed to accomplish at any given time. Modes such as: Cascade

Feltkode ændret

Control of PID loops, Operator Control for recovery from process upsets, and Maintenance Control for any unforeseen requirements. When considering larger groupings Modes such as Clean-Out, Sanitization, Production and others are often used.

Having a method to manage and coordinate the different possible modes for a Control Component grouping is a requirement and can be viewed as a separate Basic or Sequence Control Component that manages the group or groupings of other Control Components.

Feltkode ændret

5 Control shells

5.1 Control shell pattern

This standard defines the interface that is used to communicate from one Automation Module to another Automation Module. This standard will define attributes that are common to all Automation Modules. These common attributes are listed below in groups that correspond to the types of inputs and outputs defined in the Automation Module Pattern in section 5.2.

Reset (RST) – Input that causes the automation module to go to a nominal state.

Status (STS) – This is the status that reports back to the supervisory automation module regarding the state that the module is in.

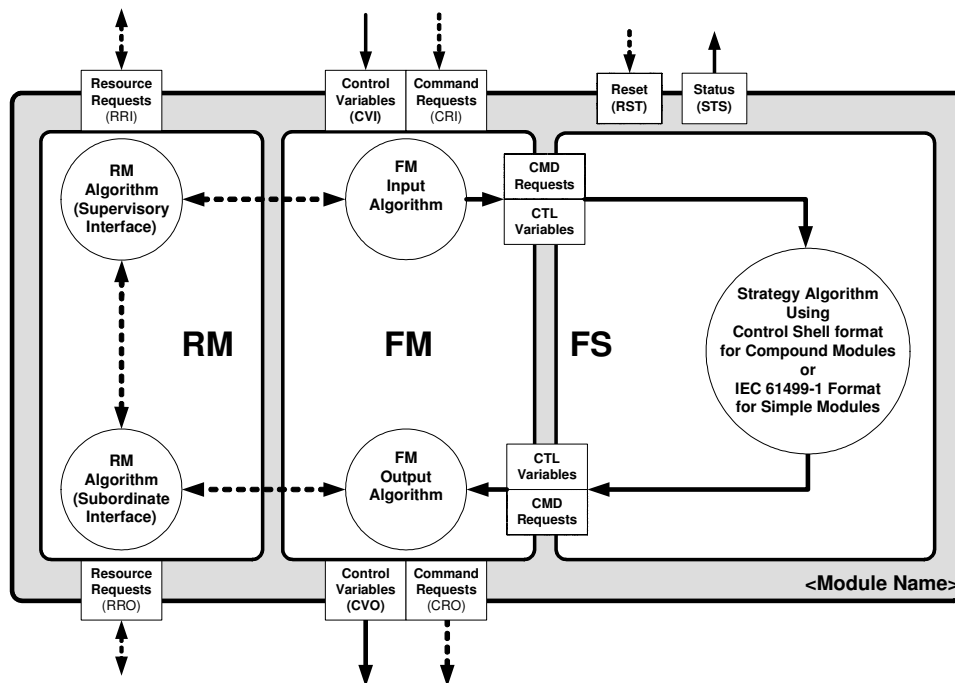


Figure 5 - Control shell pattern

Note: All algorithms may contain internal data that resides only locally.

Feltkode ændret

Resource Manager

The Resource Manager is designed to regulate who is able to direct the Automation Module. Every Automation Module will have a resource (parent) that directs it. Also it could have one or more subordinate Automation Modules that it directs. If there are more than one subordinate Automation Modules, there will be logic in the containing module, usually an Equipment Module, that will combine the multiple attributes of the subordinate Automation Modules in to one attribute for the directing Automation Module. The Resource Manager's job is to take requests from other Automation Modules and change the directing resource of it's Automation Module. It will validate whether it is OK to change the directing resource and determine whether to direct or release subordinate Automation Modules. The interface for the Resource Manager will be divided into two different sections, the Director Interface and the Subordinate Interface. The Director Interface deals with commands and status values that get sent to and from the Directing Automation Module. The Subordinate Interface deals with commands and status values that get sent to and from this module's Subordinate Automation Modules.

S88 Resource Requests (in and out) – There is the possibility of resource requests coming into and out of each automation module. Through this mechanism the automation module can receive a request for ownership from supervisory module and when necessary request ownership of subordinate module.

RM Algorithm (Supervisory interface) – This algorithm determines the current ownership status of the automation module and communicates that status with supervisory modules. The algorithm communicates with the function manager to help determine ownership and ultimately assigning ownership.

RM Algorithm (Subordinate interface) – This algorithm propagates which subordinate modules the module needs to acquire ownership of and makes the appropriate requests. This algorithm communicates with the function manager and the supervisory algorithm to help determine which subordinate modules need to be acquired and when. It then manages and communicates the subordinate module responses.

Formateret: Hold sammen med næste, Hold linjer sammen

Feltkode ændret

Single Request Algorithm Command Processor

All the Algorithms defined in the Control Shell are implemented using code call a Command Processor. This Command Processor takes inputs and uses the double integer command word to execute commands. The command word is divided into two pieces, the upper INT which indicates the “WHO” part of the command. The upper integer defines who is sending the command. The lower INT indicates the “WHAT” part of the command. The lower integer defines the command to execute.

The “WHO” part of the command is implemented using an array. This array can have at least three different formats. The format is defined in the first element of the array. The second element of the array defines the number of entries in the array.

Format 1 – First array element is equal to one (1). This format defines only two elements for the array. If the second element of the array is a zero (0), then the command processor will ignore the upper part of the command or the “WHO”.

Format 2 – First array element is equal to two (2). This format defines that the array will be defined of individual instances of the “WHO”. This standard defines the “WHO” for the Function Manager to be a list of unique ModuleIDs. In this format each element of the array would indicate an individual ModuleID. The list of ModuleIDs would start in the third array element. An example of the array would be 2,4,5,10,15,17. The first element indicates Format 2, the second element indicates there are 4 ModuleIDs defined in the array, the third element indicates Module 5, the fourth element indicates Module 10 and so forth.

Format 3 – This format defines that the array will use pairs of array elements to define the ranges of “WHO”. This standard defines the “WHO” for the Function Manager to be a list of ModuleIDs. In this format each pair of elements of the array would indicate a range. The list of ModuleIDs would start in the third array element. An example of the array would be 3,3,1,10,23,27,54,58. The first element indicates Format 3, the second element indicates there are 3 ranges defined in the array, the third element indicates to start the first range at Module 1 and go to Module 10 which is defined in the fourth array element. Then the fifth element indicates start the next range at Module 23 and end the range at Module 27 which is the sixth element in the array. This continues on until the end of the array.

The “WHAT” part of the command is also implemented using an array. With this part of the command there is no format. Each element of the array will indicate a specific function to execute. The logic for the Command Processor will look at the value of each command and perform the necessary logic to execute that command.

Formateret: Indrykning: Venstre:
1,27 cm

Feltkode ændret

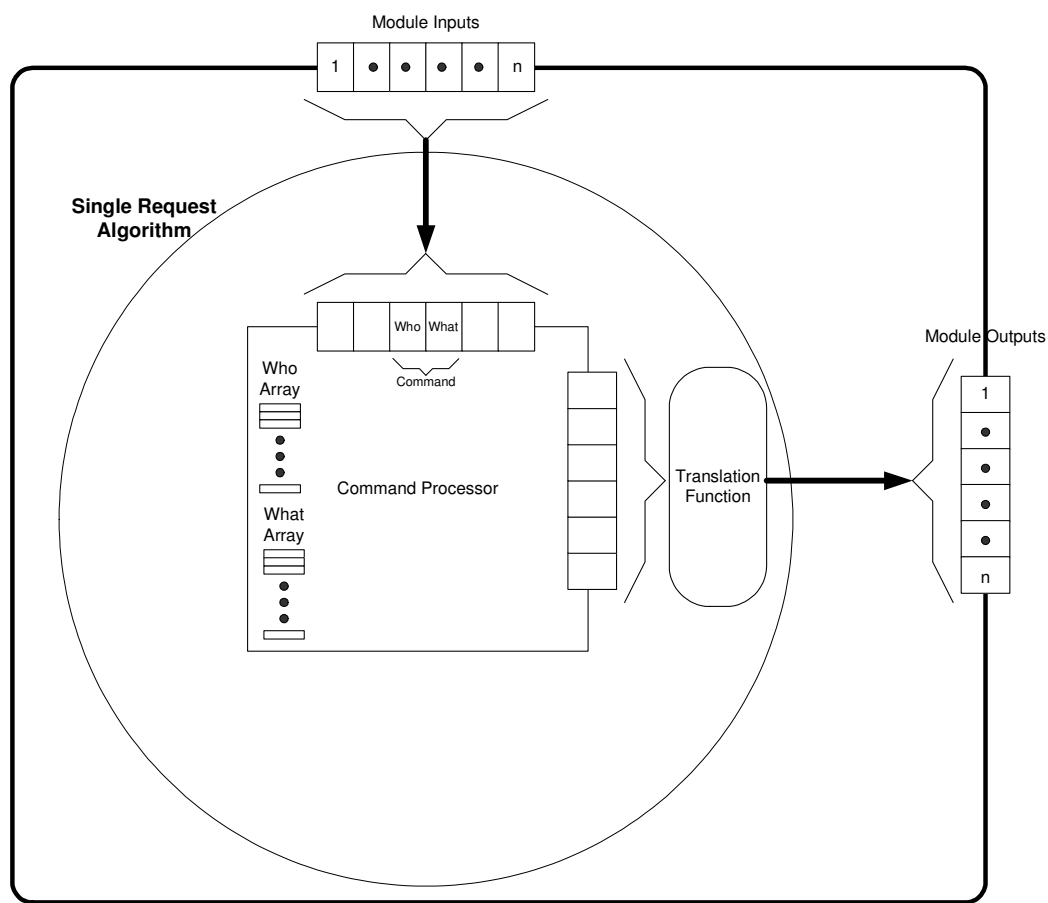


Figure 6: Common Command Processor

Functional Manager:

The Function Manager is designed to accept command and control attributes from a supervisory Automation Module. The supervisory Automation Module is determined by the Resource Manager. Only the supervisory Automation Module specified by the Resource Manager will be responded to by the Function Manager.

The Function Manager will accept command attributes which cause an action to be performed in the Functional Strategy. The common incoming commands for all Automation Modules are listed below. The control attributes are typically used by the Functional Strategy to maintain Basic Control. These control attributes are typically interlock and permissive attributes that are passed to the Functional Strategy. These control attributes are passed through the Function Manager because there may be certain cases or modes where the control signals may not be passed to the Functional Strategy. The control attributes are custom for every application so there are no common attributes defined in this document.

Feltkode ændret

The Function Manager also includes status attributes that are used to pass status of commands and control signals to the directing Automation Module.

The Function Manager is also in charge of determining the status of the Functional Strategy and determining whether the Automation Module is in a safe state to allow the directing module to change in the Resource Manager.

Control Variables/Requests - IN – These inputs are received from pertinent automation modules and represent the necessary data required to enable the automation module to perform its function.

Control Variables/Requests -OUT – These outputs are sent made available to interested automation modules and represent the necessary data required to enable the interested automation modules to perform their function.

FM Input Algorithm – Based on ownership the FM algorithm may determine the appropriate inputs to send to the functional strategy. It then maps the input names to the names utilized by the functional strategy.

FM Output Algorithm – Based on ownership the FM algorithm may determine the appropriate outputs to send to the subordinate modules. It then maps the output names of the functional strategy to the control outputs.

Feltkode ændret

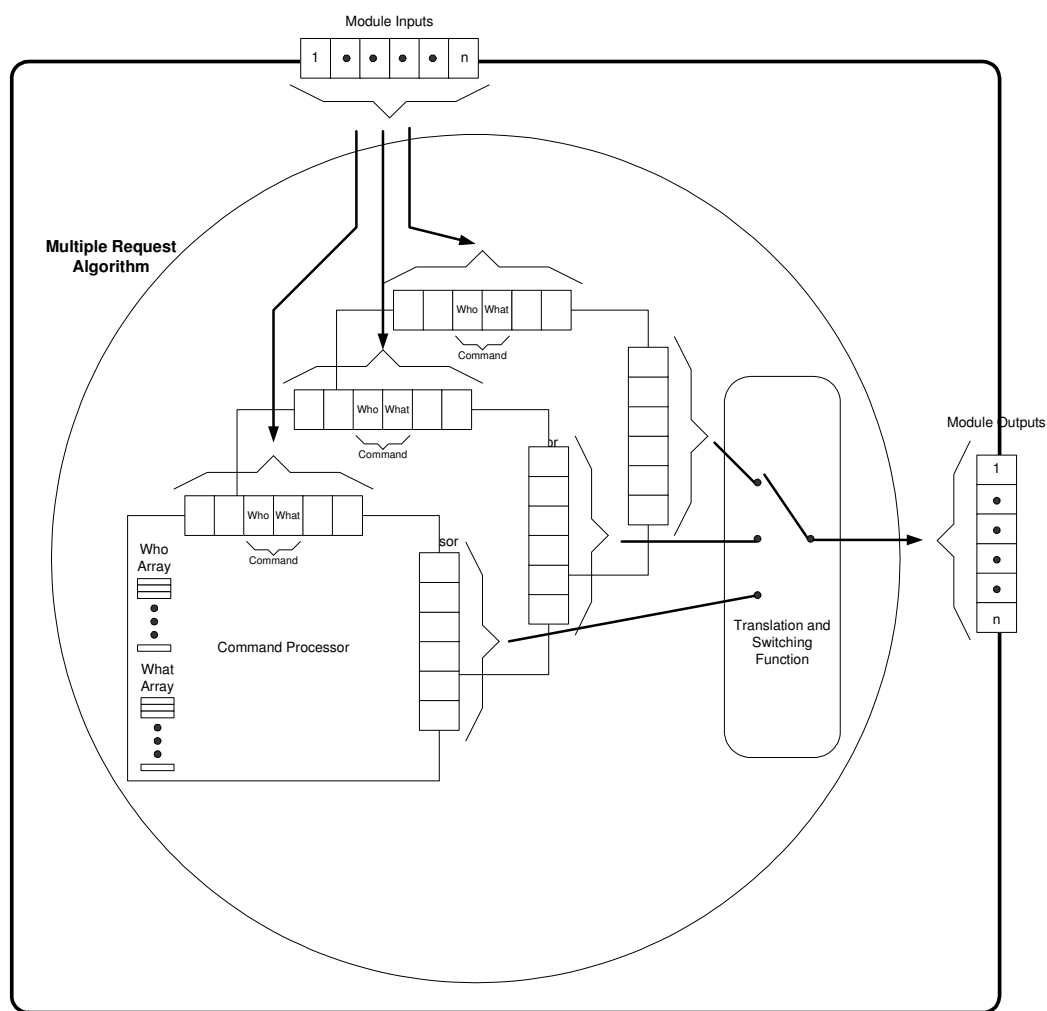


Figure 7: Common Command Processor Supporting Multiple users

Functional Strategy:

Functional Strategy (FS) implements the core Basic Control, Principle Control, or Procedural Control function of the Automation Module, resulting in command and control outputs to subordinate Automation Modules or (if executing Basic Control) physical outputs

It is presumed that the Functional Strategy of an Automation Module can be represented as one or more state diagrams, no matter where it fits in the S88 continuum of basic control, principle control, procedure aware principle control, phase, operation, unit procedure, and procedure.

Some examples of devices that would be implemented in a Functional Strategy are Servos, Material Transfers, Bipolar Devices, Two-way Values, Motors, Pumps, etc. The Functional Strategy implements the control that is necessary to make these devices work. The Functional

Feltkode ændret

Strategy is not limited to physical devices. The Functional Strategy can define a process or complete machine.

The Functional Strategy Status Outputs reflect all necessary aspects of the internal workings of the Functional Strategy and are available for view by any component. The common status attributes for the Functional Strategy are defined below.

Then Functional Strategy also have command and control attributes that are used to command and control subordinate Automation Modules. The command attributes are used to execute actions in the subordinate Automation Modules. The control attributes are used by the subordinate Automation Modules to perform Basic Control. The common command and control attributes for an Automation Module are listed below.

There are many different organizations that have already defined Functional Strategies for devices, processes and machines that are in use in manufacturing.

Strategy Algorithm – This is the core control algorithm that performs the function of the automation module.

CMD/CTL Requests/Variables (In and out) – These are a pass through of the data from and to the function manager.

5.1.1 Control Components

Every strategy is implemented by a controlling entity. In a manufacturing system this is referred to as the Control Component. The Control Component may be a human action, automation, or a combination of applying coordination, procedural and basic control on manufacturing equipment.

To support modular systems the Control Component should be focused on a single task and work with other Control Components to implement the overall strategy. There are Control Components that translate the product manufacturing strategy as specified by a unit recipe's PFD and formulation that interact with the Control Components responsible for Equipment Phase Control. When referencing the automation module pattern the control component can be thought of as the entity that implements the functional strategy.

While it is possible to create a monolith Control Component to carry out a process task strategy, such as equipment sanitization, that contains all directives in a single entity doing so reduces the reusability of that component. As an example, consider a control component that is an operator work instruction, the work instruction document may become cumbersome and difficult to support. In contrast a control component that is an operator work instruction which references other work instructions to carry out the strategy, allows for reuse of these work instructions for other strategies. For example, when it is necessary to provide specific sanitization instructions for each product being made on a line of equipment, a set of distributed control components

Feltkode ændret

would allow for the reuse of the equipment specific portion of an operator work instruction while being referenced from multiple product specific operator work instructions. For automated control components the same benefits for modular approaches apply.

Add Picture.

Logical Hierarchy of control - Modular and shared

Physical hierarchy of control – containment Monolithic and protected

5.2 Automation Module Interface

This standard defines the interface that is used to communicate from one Automation Module to another Automation Module. The definition of an Automation Module is anything in the system that can command another equipment entity. For example an Automation Module might be other Automation Logic, the HMI, the Operator or a Supervisory system. This standard will define attributes that are common to all Automation Modules. These common attributes are listed below in groups that correspond to the types of inputs and outputs defined in the Automation Module Pattern in section X.X.

5.3 Control Shell Wrapper

This Control Shell Wrapper is an interface that is used to communicate from one Automation Module to another Automation Module. This section will define the attributes that are common to all Automation Modules. These common attributes are listed and defined below. The listing is grouped by attributes that correspond to the types of inputs and outputs defined in the Control Shell pattern in section XX.XXX Editors Note: Enter correct section for Control Shell pattern.

5.3.1 Control Shell Wrapper Implementation Decisions

The first decision to make is the software methodology that you will use to implement the interface. Automation Modules will be created as grouped logic. Typically these groups of logic will be implemented as function blocks, objects, program blocks or some other vendor specific grouping. These groupings of logic can expose all internal variables or only a subset of all the variables. This is similar to software programming where you create function calls that have private and public variables. Also when you create these Automation Modules, they may be limited to one function or they may contain references to other Automation Modules. When they only contain one function the Automation Module would be considered shared, meaning that other Automation Modules can call or use the shared Automation Module. If the Automation Module contains references to other Automation Modules, this is called containment. Typically if an Automation Module is shared it will expose all the necessary variables to enable the Automation Module to perform its function. In the case where the Automation Module contains other Automation Modules, typically the containing Automation Module will not expose all the

Feltkode ændret

variables of the contained Automation Modules. There are some cases where a containing Automation Module may exposed all the variables of the contained Automation Modules, but this may lead to spaghetti code and is not recommended as good programming practice.

Once you have chosen your programming methodology, you must determine how to implement the interface. There are many different ways to implement the interface to the Automation Module. This document will give three examples that should cover most applications.

All Automation Modules will have a Resource Manager that is referenced through the interface. This Resource Manager interface will always consist of one set of inputs that are used to supervise the Automation Module. All Automation Modules will also have a Function Manager that is referenced through the interface. The Function Manager interface can vary based on your implementation. The following examples apply to the Function Manager interface used to control the Automation Module.

The first example, referred to as Class 1, -provides for a single set of inputs that will be used by all command sources. It relies on Resource Management to determine who is controlling the Automation Module and how many different input sources are needed to create the interface. The Resource Manager receives resource requests and determines what supervising Automation Module will be controlling the local module. Then when commands are sent to the Function Manager of the local module it will evaluate the command to make sure it came from the supervising Automation Module that is in control. In this example there would only be one Command Processor defined for the Function Manager.

The second example, referred to as Class 2, provides for multiple inputs to create the interface for control. In this example the Resource Manager receives resource requests and will determine what supervising Automation Module will be controlling the local module. The Function Manager will have multiple inputs for each possible supervising Automation Module. In this example there would be multiple Command Processors defined to handle the multiple inputs to the Function Manager. The Function Manager will switch (gate) from one Command Processor to another based on the decision of the Resource Manager.

The third example, referred to as Class3, really is a combination of the first two examples. In this example the Resource Manager will receive resource requests and will determine what supervising Automation Module will be controlling the local module. The Function Manager will perform both functions described above. Some inputs will have to have the commands evaluated to make sure they are from the correct source and other commands will be accepted from any command source. inputs will be switched to determine what values are used.

5.3.2 Control Shell Wrapper Rules

Defined rules can make the interface easier to design and use. There will be some rules that will be different for the examples listed above.

1. The first rule, which is common to all the examples applies to all Classes defined above, is that all Automation Modules will have an unique identifier, which is defined as an integer. This unique identifier will be used to enable the interface to subordinate Automation Modules. Typically this integer will be unique for each module, but there may be some Control Strategies that allow more than one Automation Module to have the same identifier.

Formateret: Opstilling - tal eller bogst. 5

Feltkode ændret

2. Class 1 Rule 2 ~~For the first example above, a second rule would be that every command (except the RequestID command sent to the Resource Manager – see section 5.3.4) would have to will~~ include the ID of the module that sent it. This ID would be stripped out and compared to the ID of the Automation Module that is in control as determined by the Resource Manager.
3. Class 2 Rule 2 ~~For the second example above, a second rule would be that~~ every possible supervisory Automation Module would have a unique set of inputs that it would send commands to. This would allow a supervisory Automation Module to have an exclusive interface to control ~~of~~ the subordinate Automation Module. This guarantees that no other supervisory Automation Module can accidentally send commands to the subordinate Automation Module.
4. Class 3 Rule 2, for this type of implementation there will be some commands that will include the ID of the module that sent it and some inputs will be unique to a particular supervisory Automation Module.

Based on the Control Strategy being implemented, some supervisory Automation Modules may share a set of inputs because these Automation Modules are part of the same system. However this can lead to commands being sent by multiple supervisory Automation Modules that conflict and cause the subordinate Automation Module to have a race condition.

5.3.3 Resource Manager Overview

The Resource Manager provides an incoming (Supervisory) interface and an outgoing (Subordinate) interface. The Supervisory interface is used to tell the Automation Module who is it's owner and who commands can come from. Requests to be the owner are processed by the Resource Manager. Using the Resource Request Input algorithm of the Resource Manager and input from the Function Manager, the Resource Manager will assign the owner. The Requester is the Automation Module that is the owner of the Automation Module. The owner then uses the Supervisory interface to tell Resource Manager who the Automation Module should accept commands from. The Resource is the entity that supplies the commands to the Automation Module. The interface defined in this standard will allow for any resource allocation that is necessary. The Supervisory interface exposed attributes consist of commands and status values that get sent from and to the Supervising Automation Module. This interface is implemented using a Single Request Command Processor.

The Subordinate interface is used to supervise other Automation Modules. The Resource Request Output algorithm of the Resource Manager and input from the Function Manager determines what Subordinate Automation Modules to supervise and when. The Subordinate interface exposed attributes consist of commands and status values that get sent to and from the Supervisory interface of the Subordinate Automation Modules. This interface is implemented using a Single Request Command Processor.

~~The Resource Manager is designed to regulate who is able to supervise the Automation Module. Every Automation Module will have a resource that supervises it. Also it could have one or more subordinate Automation Modules that it supervises. The Resource Manager's job is to take requests from other Automation Modules and change the supervising resource of its Automation~~

Feltkode ændret

~~Module. It will validate whether it is OK to change the supervising resource and determine whether to supervise or release subordinate Automation Modules. The interface for the Resource Manager will be divided into two different sections, the Supervisory Interface and the Subordinate Interface. The Supervisory Interface deals with commands and status values that get sent to and from the Supervising Automation Module. The Subordinate Interface deals with commands and status values that get sent to and from this module's Subordinate Automation Modules. Each of these interfaces are defined as Command Processors.~~

~~The Resource Manager Supervisory interface must support several different cases. First it must take requests and process them to determine the supervising resource. Then once that has been established there must be a way of changing the supervising resource. One way is to have the Supervisory Automation Module release control. Another way is to allow the supervising resource to change at any time. The third case would be the capability to override and force the supervising resource to release or change to something else. Also the Resource Manager must support changing the supervising resource and then restoring control back to the previous resource.~~

~~The basic Control Shell Resource Manager Supervisory interface defined in this standard will support the processing of requests by supervisory Automation Modules and only allow the resource to change when the supervisory Automation Module has released control. From this basic functionality additional implementation specific logic can be added to support the other cases for changing the supervising resource.~~

~~The basic Control Shell Resource Manager Subordinate interface defined in this standard will have one set of output attributes. If the subordinate interface needs to be linked to more than one subordinate Automation Module, there can be implementation specific code added to multiplex the attributes to all the subordinate Automation Modules.~~

The basic Control Shell Resource Manager ~~also has~~ a section for design time attributes that may be exposed to configure the Resource Manager algorithms. This section is called Administration. The developer of this Automation Module will decide whether to expose these attributes or not. Typically these items are set at design time using the programming terminal for the controller being used, but if changes to these values are required at run time the developer can expose these attributes so that the values can be modified without the need of a programming terminal.

Here is a basic overview of how the Resource Manager interfaces will work, ~~for all the examples listed above:~~

5.3.4 Supervisory Interface

This section will describe one way to implement the Single Request Command Processor for the Supervisory Interface. The implementation described uses all the functionality defined in the standard.

Any Automation Module that would like to change the ~~supervisor owner~~ of a subordinate Automation Module will write a value to the RequestID of a subordinate Automation Module. When the subordinate Automation Modules Resource Manager is ready it will process the request. ~~will look for the subordinate Automation Module's Resource Manager to be ready to accept a new supervisor.~~ This will be indicated when the

Feltkode ændret

Resource Manager status attributes, RequestStatus is not two (2 = busy processing a request) and ResourceID is zero (0 = No Resource currently in control). To allow for other types of resource allocation, there is an attribute, AllowChangeNoRelease, that will bypass the requirement for ResourceID to be zero. When AllowChangeNoRelease is set to a one (1), the Resource Manager will process any request when RequestStatus is not two (2 = busy processing a request).

The Resource Manager will wait until the Supervisory Automation Module sets the attribute, RequestID, to the ~~value of the supervisors~~ Module ID of the Supervisory Automation Module. The Resource Manager will check the list of permitted supervisor Module IDs using the array attribute, ValidRequestID[] . The Resource Manager will then validate, from the Functional Manager, whether it is OK to change the supervising resource. (This attribute name is not defined in this standard because it does not exist outside the Automation Module.) If the validation with Function Manager is not required, this value can be force to a True value all the time. If it is OK to change the directing resource, the Resource Manager will write the director's Module ID to the status output RequestID and sets the incoming RequestID to zero (0). It also sets the status output, RequestStatus, to the value of two (2) indicating a request is pending.

The Resource Manager will have a timer that can be enabled to block further execution for a period of time. This timer will be used to facilitate Asynchronous communications with other Supervising Automation Modules. The preset for this timer comes from the attribute, AsyncPreset. For other resource allocation requirements, the AsyncPreset can be set to zero (0) which will allow the other commands to execute immediately.

The Resource Manager will wait until the Asynchronous communications timer, if used, has expired and ~~the supervisor sets~~ will then accept an input to the attribute, ResourceID which is a DINT. This ResourceID has two parts, the Resource ID to be put in control which is put in the lower INT and the Module ID of the sending Automation Module which is put in the upper INT. This upper INT is validated against the status value of RequestID to make sure the sending Automation Module is in control. Depending on the value (1 or 0) of attribute, SubordinatePassThru, when the ResourceID is set from the supervising Automation Module, the Resource Manager may execute the Subordinate Interface logic to acquire supervisor status of the subordinate Automation Modules or it may wait for Function Manager input. This Function Manager input to Resource Manager controls when Resource Manager acquires and releases subordinate modules. The Subordinate Interface is defined below.

If the SubordinatePassThru attribute is on, when the Subordinate Interface completes the acquisition of subordinate modules the Resource Manager will change the status output, RequestStatus, to a value of 1 indicating a successful change of supervising Automation Module and set the outgoing status attribute, ResourceID, to the value of the incoming command attribute, ResourceID . If the Subordinate Interface does not complete within the time period specified by the attribute, TimeOutPreset, then the status output, RequestStatus, is set to a value of minus one (-1) to indicate a failure to acquire supervisor status of the Subordinate Automation Modules.

If the SubordinatePassThru attribute is off, then the Resource Manager will Change the status output, RequestStatus, to a value of 1 indicating a successful change of supervising

Formateret: Ikke Fremhævning

Feltkode ændret

Automation Module and set the outgoing status attribute, ResourceID, to the value of the incoming command attribute, ResourceID.

When the ~~supervising Automation Module~~Requestor is finished with this module, it will set the attribute, ResourceID to zero (0). The Resource Manager will then set the status output attribute, ResourceID, to zero (0). This starts a timer in the Resource Manager that will delay for the preset time defined in attribute, ReleaseDelay. The Requestor can change the ResourceID at anytime without any delays. If the strategy is to release immediately, the ReleaseDelay preset can be set to zero.

If another ~~supervisor~~Requestor does not take control of this module in the delay time and the SubordinatePassThru attribute is on, the Resource Manager will execute the Subordinate Interface logic to release all subordinate Automation Modules. The Subordinate Interface is defined below. When the Subordinate Interface completes the release of all subordinate modules the Resource Manager will set the attribute, RequestStatus, to a one (1) for success. If the Subordinate Interface does not complete within the time period specified by the attribute, TimeOutPreset, then the status output, RequestStatus, is set to a value of minus one (-2) to indicate a failure to release director status of the Subordinate Automation Modules.

If another ~~supervisor~~Requestor does not take control of this module in the delay time and the SubordinatePassThru attribute is off, the Resource Manager will set the attribute, RequestStatus, to a one (1) for success.

5.3.5 Subordinate Interface

This section describes one way to implement the Single Request Command Processor for the Subordinate Interface. The implementation described uses all the functionality defined in the standard.

The Resource Manager Subordinate Interface mirrors the Supervisory Interface so that the two interfaces may be connected together to allow a hierarchy of control. When the ResourceID is set from the supervisor and the SubordinatePassThru attribute is true or when the Function Manager requests, the Resource Manager will execute the Subordinate Interface logic to acquire supervisory status of the subordinate Automation Modules. The Function Manager will send an attribute, Acquire equal to True, that will cause Resource Manager to acquire all the subordinate Automation Modules. As long as the Subordinate Automation Modules are not acquired, incoming RequestStatus not equal to 1 (Success – Acquired), Resource Manager will continue to try to acquire them. If the Acquire attribute is False, Resource Manager will stop trying to acquire the subordinates. The Subordinate interface will have custom code that is used to send commands to the Resource Manager of the Subordinate Automation Modules to get ownership. This custom code will use the Resource Request Output attributes to acquire ownership and report status.

First, this Automation Module will write a value to the RequestID of the subordinate Automation Modules it wants to own. This value is the ModuleID of this Automation Module. When the subordinate Automation Modules Resource Manager is ready it will process the request. This will be indicated when the Resource Manager status attributes,

Feltkode ændret

~~RequestStatus is not two (2 = busy processing a request) and ResourceID is zero (0 = No Resource currently in control). To allow for other types of resource allocation, there is an attribute, AllowChangeNoRelease, that will bypass the requirement for ResourceID to be zero. When AllowChangeNoRelease is set to a one (1), the Resource Manager will process any request when RequestStatus is not two (2 = busy processing a request). If there are multiple subordinate Automation Modules, there can be implementation specific code that will take the attributes described here and multiplex them out to all the subordinate Automation Modules.~~

~~The first step is to check to see if all the subordinate Automation Modules are ready to accept a new supervisor. This is done by monitoring the incoming status attributes, ResourceID and RequestStatus, to see if the ResourceID is equal to zero (0) and the RequestStatus is not equal to two (2). If this condition is true, the Resource Manager will then set the attribute RequestID to the value of the ModuleID.~~

The Resource Manager will wait until the incoming status attribute, RequestID, is equal to the ModuleID. When this occurs the Resource Manager knows it is allowed to change who is supervising the subordinate Automation Modules. Now the Resource Manager will set the outgoing command attribute, ResourceID, to the ModuleID of the local module in the upper integer and to the ModuleID of the Automation Module that should supervise the subordinate Automation Modules in the lower integer. Typically this is the ModuleID of the local module, but it does not have to be. For instance an HMI can tell an Automation object that it is supervised by an upstream piece of equipment rather than ~~its own state model~~ the HMI itself.

Now the Resource Manager starts a timer with the preset equal to the value in attribute TimeoutPreset. If the incoming status attribute RequestStatus is equal to one (1) before the timer times out, the Resource Manager will set the Supervisor Interface outgoing status attribute RequestStatus to a one (1). If the timer reaches its timeout value the Resource Manager will set the Supervisor Interface outgoing status attribute, RequestStatus, to a minus one (-1) to indicate a failure to acquire the subordinate Automation Modules.

When the Resource Manager is done with the subordinate Automation Modules, this is when the supervisor interface sets ResourceID to zero and when the Release Delay timer finishes, the Resource Manager will attempt to release all the subordinate Automation Modules. At this time the Resource Manager will write a value of zero (0) to the outgoing command attribute, ResourceID. When the subordinate incoming request status attribute, RequestStatus, is equal to zero (0), this will start a timer with a preset equal to the value in attribute TimeoutPreset. If the incoming status attribute RequestStatus is equal to one (1) before the timer times out, the Resource Manager will set the Supervisor Interface outgoing status attribute RequestStatus to a one (1). If the timer reaches its timeout value the Resource Manager will set the Supervisor Interface outgoing status attribute, RequestStatus to a minus two (-2) to indicate a failure to release the subordinate Automation Modules.

Feltkode ændret

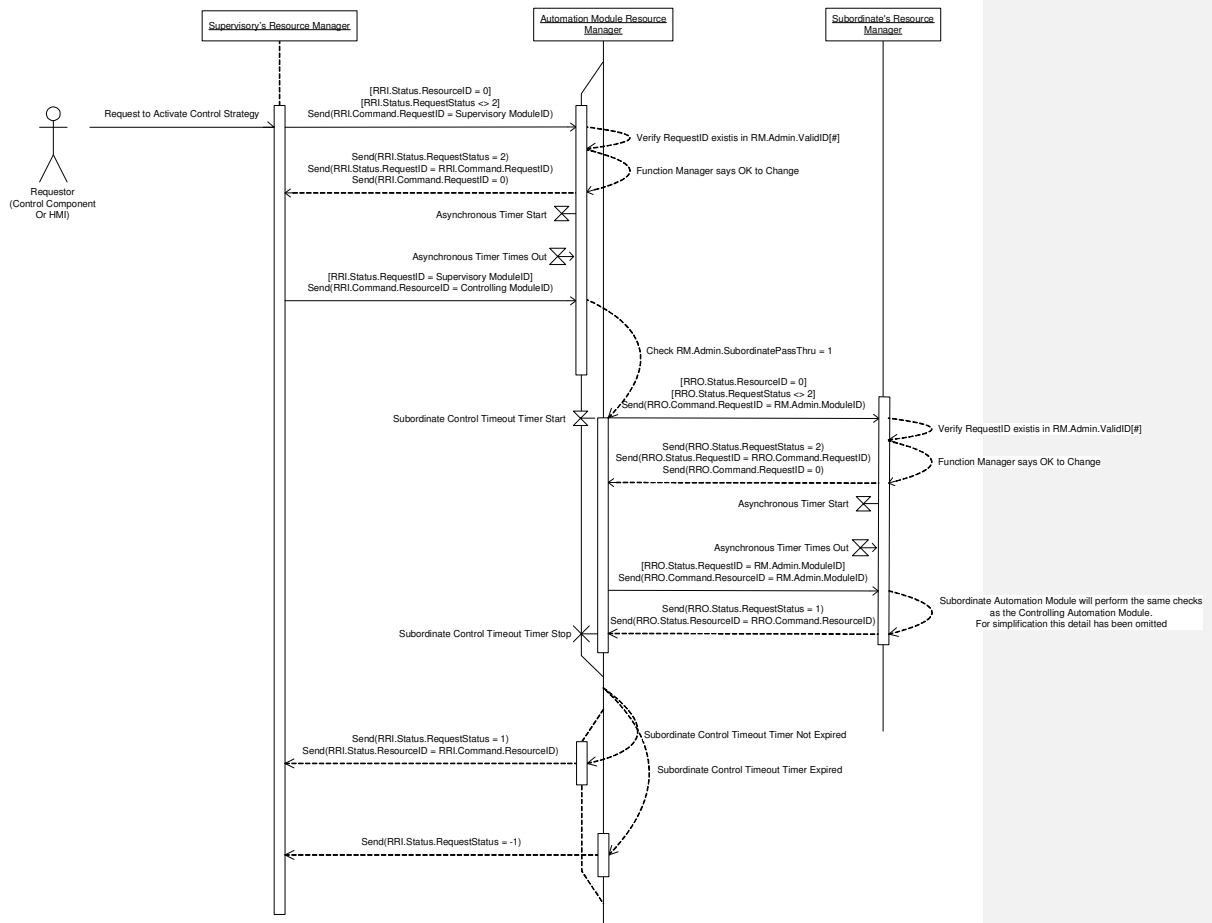


Figure 8 - Resource Manager Activate Sequence Diagram

Feltkode ændret

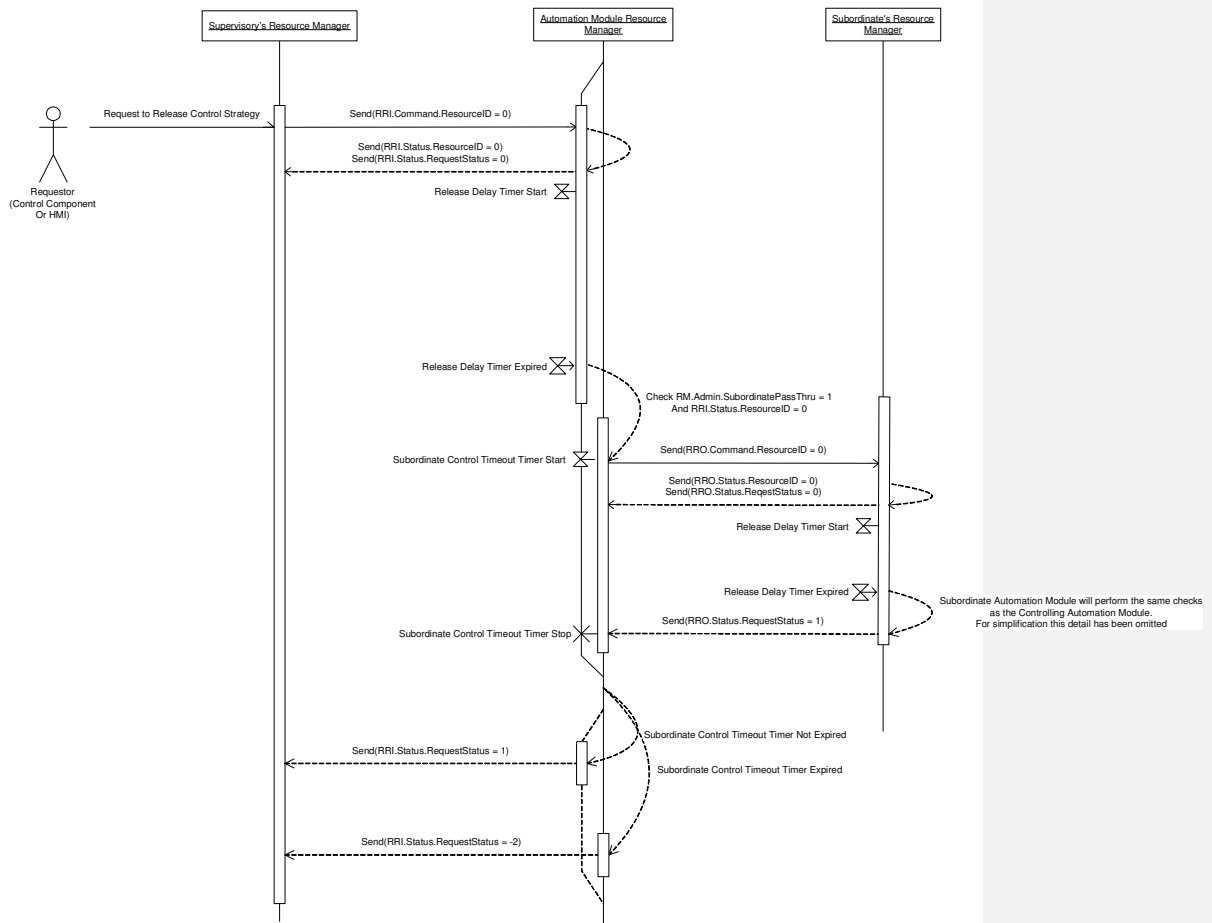


Figure 9 - Resource Manager Release Sequence Diagram

Feltkode ændret

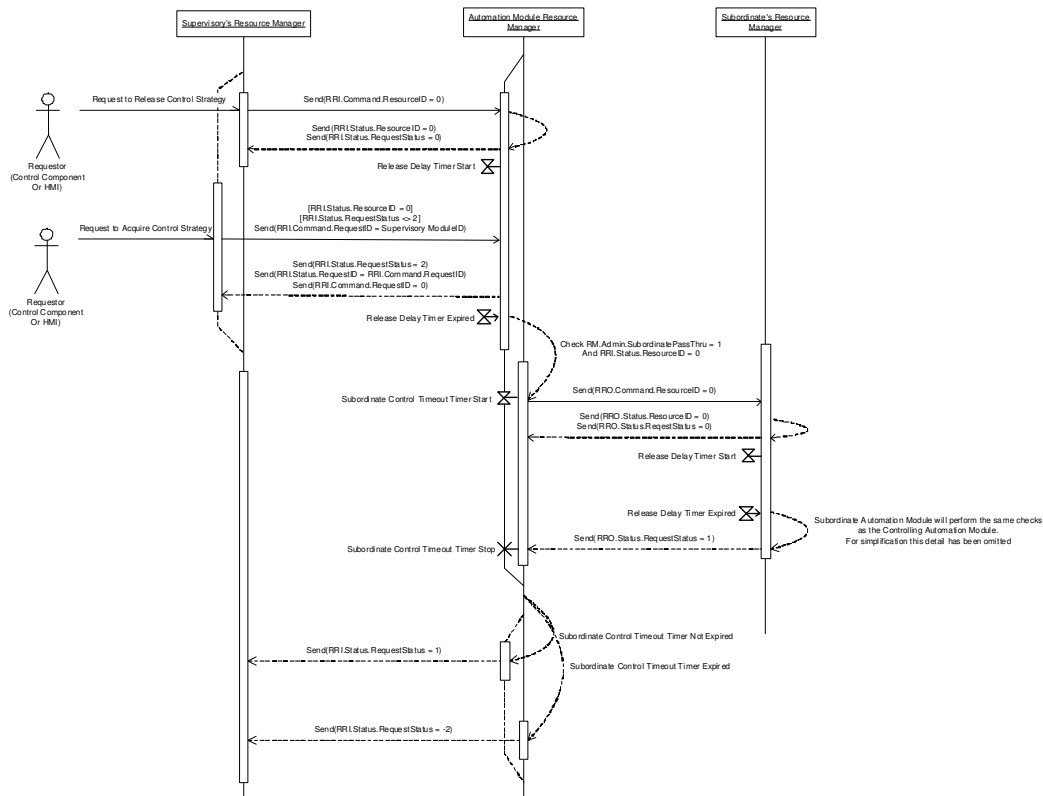


Figure 10 - Resource Manager Release Sequence Diagram with another Activate Request

5.3.6 Resource Manager (RM) Supervisory Interface

Resource Request Inputs (RRI) Commands

MEC.RRI.Command.RequestID – (INT) – This input tells the Resource Manager who is requesting a change in the resource that will be supervising-controlling the Automation Module. This value is verified against the MEC.RM.Admin.ValidRequestID[] to validate the sending Module ID is allowed to send a value. This input is the Module ID of the specific equipment entity that is requesting to change the value of the resource that is supervising-controlling the Automation Module. This input is implemented using an integer value and has a range from 1 to 32767.

MEC.RRI.Command.ResourceID – (DINT) – This input tells the Resource Manager what resource will be supervising-controlling the Automation Module. This input is only set by the equipment entity specified by the Module ID in

Feltkode ændret

MEC.RRI.Command.RequestID, when the supervisor verifies the Automation Module is listening to it. The upper INT is the Module ID of the Automation Module sending the value. ~~The value of this attribute does not have to, and must~~ be the same as the value of the MEC.RRI.Command.RequestID. ~~There can be one Automation Module that changes the supervising resource of a subordinate Automation Module for another Automation Module. This value is verified against the~~

~~MEC.RM.Admin.ValidRequestID[] to validate the sending Module ID is allowed to send a value.~~ The lower INT is the Module ID of the specific equipment entity that will be ~~supervising-controlling~~ the Automation Module. The value for this Resource Module ID is 1 to 32767. However there is a special value, -1, that will reset the Resource Manager back to a default state. See the ~~Fejl! Henvissningskilde ikke fundet, Table 1~~ below. This input is implemented using a double integer value.

Table 1- MEC.~~RM.Admin.ValidRequestID~~RRI.Command.ResourceID (Lower INT)

-1	Resource Manager Reset Request
0	No Module in Control
1	Module ID 1
•	
•	
•	
32767	Module ID 32767

Resource Request Inputs (RRI) Status

MEC.RRI.Status.RequestID – (INT) – This status value indicates the Module ID of the ~~supervising~~ equipment entity that has been acknowledged by the Resource Manager to change the ~~supervising-controlling~~ resource of the Automation Module. Once this value is set and read by the supervisor, the supervisor can then send the value for the resource it wants to ~~direct-control~~ this Automation Module. This input is implemented using an integer value.

MEC.RRI.Status.ResourceID – (INT) – This status value indicates the Module ID of the equipment entity that has been allowed to ~~direct-control~~ the Automation Module. This input is implemented using an integer value.

MEC.RRI.Status.RequestStatus – (INT) – This status value indicates the status of the request to change the resource ~~supervising-controlling~~ the Automation Module. This input has many different values that indicate the specific states of the request. At a minimum there will be five (5) values as shown in Table 2:

Feltkode ændret

Table 2 - MEC.RRI.Status.RequestStatus

-2	Failed to Release
-1	Failed to Control
0	Initialized
1	Success
2	Pending Request

MEC.RRI.Status.LastResourceID – (DINT) – This status value is the last ResourceID command that was sent. This input is a copy of the incoming command and is implemented using a double integer value. This value is updated everytime a new ResourceID is received.

MEC.RRI.Status.LastRequestStatus – (INT) – This status value is the returned status for the last ResourceID command that was sent. This input is a copy of the status, MEC.RRI.Status.RequestStatus, when the last command was executed. This value is updated everytime a new ResourceID is received.

5.3.7 Resource Manager (RM) Subordinate Interface

Resource Request Outputs (RRO) Status

MEC.RRO.Status.RequestID – (INT) – This input tells the Resource Manager who is allowed to request a change in the resource ~~supervising~~controlling the subordinate Automation Module. If this input matches the Automation Module's ID, then the Resource Manager can send the MEC.RRO.Command.ResourceID to the subordinate Automation Module. ~~This input is the Module ID of the specific equipment entity that is requesting to change the value of the resource supervising the subordinate Automation Module.~~ This input is implemented using an integer value.

MEC.RRO.Status.ResourceID – (INT) – This input tells the Resource Manager what resource ~~will be supervising~~has control of the subordinate Automation Module. ~~This input is the Module ID of the specific equipment entity that will be supervising the subordinate Automation Module.~~ This is implemented using an integer value.

MEC.RRO.Status.RequestStatus – (INT) – This input tells the Resource Manager the value of the subordinate Automation Module's request status. This is implemented using an integer value and will support at least the following values.

Formateret: Hold sammen med næste, Hold linjer sammen

Formateret: Hold sammen med næste, Hold linjer sammen

Formateret: Hold sammen med næste, Hold linjer sammen

Formateret: Hold sammen med næste, Hold linjer sammen

Formateret: Hold sammen med næste, Hold linjer sammen

Formateret: Hold sammen med næste, Hold linjer sammen

Feltkode ændret

-2	Failed to release Subordinate Automation Module
-1	Failed to control Subordinate Automation Module
0	Initialized
1	Success
<u>2</u>	<u>Pending Request</u>
<u>3</u>	<u>Acquiring Subordinate Automation Modules</u>
<u>24</u>	<u>Releasing Subordinate Automation Modules</u> <u>Pending Request</u>

Resource Request Outputs (RRO) Commands

MEC.RRO.Command.RequestID – (INT) – This output is sent to the Subordinate Resource Manager to request access to change the resource that is supervising the Subordinate Automation Module. This output is the Module ID of this Automation Module. This output is implemented using an integer value and has a range from 1 to 32767.

MEC.RRO.Command.ResourceID – (DINT) – This output is sent to the Subordinate Resource Manager to change what resource will be supervising-controlling the Subordinate Automation Module. This output is only set if the MEC.RRO.Status.RequestID is the Module ID of this Automation Module. The upper INT is the Module ID of the Automation Module sending the value and would be the Module ID of this Module. The lower INT is the Module ID of the specific equipment entity that will be supervising-controlling the Automation Module. Typically this will be the Module ID of this Automation Module. This lower INT can also be a -1 which will send a reset command to the subordinate modules. See Table below. This output is implemented using a double integer value.

-1	Resource Manager Reset
0	No Module in Control
1	Module ID 1
•	
•	
•	
32767	Module ID 32767

Feltkode ændret

5.3.8 Resource Manager (RM) Administrative Interface

Resource Manager (RM) Administration

MEC.RM.Admin.ValidRequestID[] – (INT Array) – This value is usually configured at design time, but can be exposed to allow runtime changes. It lists the valid Module IDs that can supervise this Automation Module. The valid values follow the supported types of the Command Processor “WHO” array, see section XXXXX Editors Note: Enter correct Section. Using the instance format, one set of values are as follows:

Formateret: Fremhævning

Formateret: Fremhævning

→	Resource Manager Reset Request
0	No Module in Control
1	Module ID 1
•	
•	
•	
32767	Module ID 32767

MEC.RM.Admin.SubordinatePassThru – (Boolean) – This value is usually configured at design time, but can be exposed to allow runtime changes. It determines whether the Resource Manager will automatically pass thru the resource requests or whether the subordinate interface will be controlled by the Function Manager.

MEC.RM.Admin.TimeoutPreset – (INT) – This value is typically configured at design time, but can be exposed to allow runtime changes. It indicates the preset for the timeout timer. This is the time that the Resource Manager will wait for the acquisition or release of subordinate Automation Modules before returning an error to MEC.RRI.Status.RequestStatus. This value is the number of milliseconds to wait.

MEC.RM.Admin.ReleaseDelay – (INT) – This value is typically configured at design time, but can be exposed to allow runtime changes. It indicates the preset for the release timer. This is the time the Resource Manager will wait before commanding a release of the subordinate Automation Module. This will keep the subordinate Automation Modules from being released and re-acquired very quickly. This value is the number of milliseconds to wait.

MEC.RM.Admin.ModuleID – (INT) – This value is typically configured at design time, but can be exposed to allow runtime changes. This value is the unique identifier for this Automation Module. This value is implemented as an integer.

MEC.RM.Admin.AsyncPreset – (INT) – This value is typically configured at design time, but can be exposed to allow runtime changes. This value is the Asynchronous communications timer preset that is used to delay further processing of the Resource Manager. This allows for a slow communications link between the parent Automation Module and the Resource Manager. This value is the number of milliseconds to wait.

Feltkode ændret

MEC.RM.Admin.AllowChangeNoRelease – (BOOL) – This value is typically configured at design time, but can be exposed to allow runtime changes. This value is used to allow Resource Manager to bypass the requirement for ResourceID to be zero (0 = No Resource in control), when determining when to process a request to change RequestID or owner.

5.3.9 Function Manager Overview

The Function Manager is designed to accept command and control attributes from a supervising controlling Automation Module and send command and control attributes to subordinate Automation Modules. The supervising-controlling Automation Module is determined by the Resource Manager.

The Function Manager will accept command requests which cause an action to be performed in the Functional Strategy. Command Requests also have a status for the command that is sent back to the supervising-controlling Automation Module. Each Command Request is processed by Function Manager and mapped to the inputs ~~for-of~~ the Functional Strategy.

As mentioned above in section 5.3.1, there are several ways the inputs to the Function Manager can be implemented. If you follow the Class 1~~first~~ example, where there is only one input for all the supervisory Automation Modules to use, then all command variables used in the Function Manager will have two parts, the upper integer will be the ModuleID of the Automation Module sending the command. The Function Manager will use the list of permitted supervisory modules, which resides in the attribute MEC.FM.Admin.ValidModuleID[], to determine if the command is valid. The lower integer of the command variables will be for the specific command. These commands are processed by one Command Processor.

If you follow the ~~second~~-Class 2 example where there are multiple command inputs for every supervisory Automation Module, then the Function Manager will use the supervising resource value from the Resource Manager to determine which set of inputs to process and map to the Functional Strategy. It will also use the list of permitted supervisory modules, which resides in the attribute MEC.FM.Admin.ValidModuleID[], to determine if the command is valid. In this example there will be multiple Command Processors to handle the multiple inputs.

For the ~~third~~-Class 3 example which is a combination of the ~~first two~~Class 1 and Class 2 examples, then the Function Manager would have to implement both types of command inputs.

The common Command Request Input attributes for an Automation Module are listed below.

~~The basic Control Shell defined in this standard will use a single input for each command input. From this basic functionality any requirement for multiple inputs or other implementation specific requirements can be supported using additional code. The common incoming command requests for all Automation Modules are listed below.~~

Control variables are different from the Command Variables in that the Control Variables are monitored all the time. The control variables are typically used by the Functional Strategy to maintain Basic Control. These control variables are typically interlock and permissive attributes or parameters that are passed to the Functional Strategy. Since these control variables are used to maintain basic control, they do not require multiple inputs or passing the ModuleID of the sending Automation Module. These control variables are passed through the appropriate Command Processor of the Function Manager to be mapped and converted to the control inputs

Feltkode ændret

of the Functional Strategy. The control variables are custom for every type of Automation Module, so there are no common control variables defined in this document.

The Function Manager also interfaces with the Resource Manager, using the Acquire attribute, to control whether the Resource Manager performs the Acquire or Release of Subordinate Automation Modules.

The Function Manager will also take output command requests and control variables from the Functional Strategy and send these attributes out to subordinate Automation Modules. These are not sent to the subordinate Automation Modules until the Resource Manager Subordinate interface has validated that the module is in control of all the subordinate modules. The command requests are used to execute actions in the subordinate Automation Modules. The control variables are used by the subordinate Automation Modules to perform Basic Control. The common command requests and control variables for an Automation Module are listed below.

~~The Basic Control Shell defined in this standard will use a single output for each command request or control variable. If the Function Manager outputs need to be linked to more than one subordinate Automation Module, there can be implementation specific code added to multiplex the attributes to all the subordinate Automation Modules.~~The Function Manager Control Request Output algorithm will have customer code that is used to send command and control attributes to all the Subordinate Automation Modules that are owned by this Automation Module. This custom code will use the Control Request Output attributes to send commands and report status.

The common outgoing command requests for all Automation Modules are listed below.

The Function Manager is also in charge of determining the status of the Functional Strategy and determining whether the Automation Module is in a safe state to allow the supervising module to change in the Resource Manager.

~~The basic Control Shell for The Function Manager has attributes that may be exposed to configure the Function Manager algorithms. This section is called Administration. The developer of this Automation Module will decide whether to expose these attributes or not. may also expose administrative attributes from the Functional Strategy. These administrative~~Typically these attributes ~~usually~~ are setup at design time using a programming terminal for the control system. If it is desired to change these administrative attributes without the need for a programming terminal, then these attributes ~~should can~~ be exposed through the Function Manager Manager as variables (usually with values that are initialized at startup or at least preset at during system design/commissioning and maintained through power cycles).

5.3.10 Function Manager (FM) Interface

Control Request Inputs (CRI) Commands

MEC.CRI.Command – (DINT) – This input allows the resource that is in command to request that the Automation Module execute a command. The upper integer will hold the ModuleID of the sending Automation Module. This value is validated against the MEC.FM.Admin.ValidModuleID[] array. The lower integer will hold the Command value which will be one of the following:

Feltkode ændret

0	Undefined
1	Auto Mode of Action Request
2	Semi-Auto Mode of Action Request
3	Manual Mode of Action Request
4	Alarm Acknowledge
5	Reset Function Manager and Functional Strategy

The Mode of Action determines the action of the Functional Strategy. For example when a PID loop is put in to manual the input setpoint is written directly to the output of the PID function bypassing the PID algorithm altogether.

Control Request Inputs (CRI) Status

MEC.CRI.Status – (INT) – This is an output back to the commanding Automation Module to indicate the status of the ~~last~~ command. The value will be one of the following:

-2	Command Failed – Not valid command
-1	Command Failed – Not Authorized
0	Not Defined
1	Command Succeeded and Processed
2	Command Succeeded – Not in Control of Module

MEC.CRI.LastCommand – (DINT) – This value is a copy of the last command executed regardless of command status. This value is updated every time a new command is received.

MEC.CRI.LastCommandStatus – (INT) – This value is a copy of the status after the last command was executed. This value is updated every time a new command is received.

Control Variable Inputs (CVI) Inputs

Currently there are no control inputs defined as being common to all Automation Modules. This is typically where physical inputs, interlocks, permissives, parameters or feedback signals are input to the Automation Module. Since every type of Automation Module is different, this specification is not defining inputs of this type.

Feltkode ændret

If there were control inputs defined the attributes would have the following naming:

MEC.CVI.XXXXXX

where the XXXXXX would be replaced with the attribute name.

Feltkode ændret

5.3.11 Functional Manager (FM) Outgoing Command Outputs

Control Request Outputs (CRO) Commands

MEC.CRO.Command – (DINT) – This output allows the Automation Module to request a Subordinate Automation Module to change the Mode of Action. The upper integer will hold the ModuleID of this Automation Module. The lower integer value for ModeRequest will be one of the following:

0	Undefined
1	Auto Mode of Action Request
2	Semi-Auto Mode of Action Request
3	Manual Mode of Action Request
4	Alarm Acknowledge
5	Reset Request

Then Mode of Action determines the action of the Functional Strategy. For example when a PID loop is put in to manual the input setpoint is written directly to the output of the PID function bypassing the PID algorithm altogether.

Control Request Outputs (CRO) Status

MEC.CRO.Status – (INT) – This is an output back to the commanding Automation Module to indicate the status of the last command. The value will be one of the following:

-2	Command Failed – Not valid command
-1	Command Failed – Not Authorized
0	Not Defined
1	Command Succeeded and Processed
2	Command Succeeded – Not in Control of Module

Control Variable Outputs (CVO) Outputs

Currently there are no control outputs defined as being common to all Automation Modules. This is typically where physical outputs, or outputs to other control modules are specified. Since every Automation module is different, this specification is not defining inputs of this type.

Feltkode ændret

If there were control outputs defined the attributes would have the following naming:
MEC.CVO.XXXXXXX
where the XXXXXX would be replaced with the attribute name.

5.3.12 Function Manager (FM) Administrative Inputs

Function Manger (FM) Administration

MEC.FM.Admin.ValidModuleID[] – (INT Array) – This value is typically configured at design time, but can be exposed to allow for runtime changes. It lists the valid Module IDs that can send commands to this Automation Module. The valid values follow the supported types of the Command Processor “WHO” array, see section XXXXX Editors Note: Enter correct Section. Using the instance format, one set of values are as follows:

<u>0</u>	<u>No Module in Control</u>
<u>1</u>	<u>Module ID 1</u>
<u>.</u>	
<u>.</u>	
<u>.</u>	
<u>32767</u>	<u>Module ID 32767</u>

- ← Formateret: Punktopstilling
- ← Formateret: Punktopstilling
- ← Formateret: Punktopstilling

Feltkode ændret

5.4 Functional Strategy Overview

Functional Strategy (FS) implements the core Basic Control, Principle Control, or Procedural Control function of the Automation Module, resulting in command and control outputs to the Function Manager which will be sent to the subordinate Automation Modules or (if executing Basic Control) physical outputs.

It is presumed that the Functional Strategy of an Automation Module can be represented as one or more state diagrams, no matter where it fits in the ANSI/ISA 88 continuum of basic control, principle control, procedure aware principle control, phase, operation, unit procedure, and procedure.

The Functional Strategy implements the control that is necessary to make the Automation Module work. The Functional Strategy can represent physical devices, equipment modules, procedural elements, processes or complete machines. Some examples of devices that would be implemented in a Functional Strategy are Servos, Material Transfers, Bipolar Devices, Two-way Valves, Motors, Pumps, etc.

The Functional Strategy inputs and outputs are mapped and converted to inputs and outputs in the Function Manager. This mapping and converting should only expose the necessary command requests, control variables or administrative attributes.

There are many different organizations that have already defined Functional Strategies for devices, processes and machines that are in use in manufacturing. These definitions can be used to extend this standard and create new types of Automation Modules.

5.5 MEC Status Overview

Every Automation Module will have status data that will indicate the state of the Automation Module. This data is an overall status and not limited to any one internal algorithm.

MEC Status

MEC.Status.AlarmPresent – (BOOL) – This status signal indicates that the Automation Module is in an alarm condition.

MEC.Status.AlarmAcknowledged – (BOOL) – This status signal indicates that the Automation Module has acknowledged all alarm.

MEC.Status.Reset – (BOOL) – This status signal indicates that the Automation Module has been reset to a default state.

MEC.Status.ModeCurrent – (INT) – This status value indicates the current Mode of Action for the Automation Module. The value will be one of the following values:

0	Undefined
1	Auto Mode of Action
2	Semi-Auto Mode of Action
3	Manual Mode of Action

Feltkode ændret

MEC.Status.ModeRequested – (INT) – This status value indicates the Mode of Action that has been requested. The value will be one of the following values:

0	Undefined
1	Auto Mode of Action
2	Semi-Auto Mode of Action
3	Manual Mode of Action

MEC.Status.RequestStatus – (INT) – This status value indicates the status of the request to change the resource in command. This input has many different values that indicate the specific equipment entity that can be in command. At a minimum there will be three (3) values as follows:

-1	Failed
0	Initialized
1	Success

The Initialized value is the default before any request is processed. With these three values you will always know that there was a change when the request is processed.

5.5.1 MEC Reset Command

Every Automation Module will have a global command to reset the entire Automation Module.

MEC.Reset.ResetRequest – (DINT) – This input allows the resource that is in command to request that the Automation Module be reset to a default state. The upper integer will hold the ModuleID of the sending Automation Module. The lower integer holds the commanded state for the Reset, True or False. This is used throughout the Automation Module to reset all pieces, Resource Manager, Functional Manager and Functional Strategy, to a known default state.

Feltkode ændret

5.6 MEC Control Shell Variable List

The following is a list of all the common variables defined in the Control Shell.

Feltkode ændret

						Data Type
DeviceName					DeviceName	
	MEC				DeviceName.MEC	S8805
		RRI			DeviceName.MEC.RRI	Supervisory
			Command		DeviceName.MEC.RRI.Command	Command
				RequestID	DeviceName.MEC.RRI.Command.RequestID	Int (16 bit)
				ResourceID	DeviceName.MEC.RRI.Command.ResourceID	Int (32 bit)
			Status		DeviceName.MEC.RRI.Status	Status
				RequestID	DeviceName.MEC.RRI.Status.RequestID	Int(16 bit)
				ResourceID	DeviceName.MEC.RRI.Status.ResourceID	Int(16 bit)
				RequestStatus	DeviceName.MEC.RRI.Status.RequestStatus	Int(16 bit)
		RRO			DeviceName.MEC.RRO	Subordinate
			Command		DeviceName.MEC.RRO.Command	Command
				RequestID	DeviceName.MEC.RRO.Command.RequestID	Int (16 bit)
				ResourceID	DeviceName.MEC.RRO.Command.ResourceID	Int (32 bit)
			Status		DeviceName.MEC.RRO.Status	Status
				RequestID	DeviceName.MEC.RRO.Status.RequestID	Int(16 bit)
				ResourceID	DeviceName.MEC.RRO.Status.ResourceID	Int(16 bit)
				RequestStatus	DeviceName.MEC.RRO.Status.RequestStatus	Int(16 bit)
		RM			DeviceName.MEC.RM	RM
			Admin		DeviceName.MEC.RM.Admin	Admin
				ValidRequestID[#]	DeviceName.MEC.RM.Admin.ValidRequestID[#]	Array of Int (16 bit)
				SubordinatePassThru	DeviceName.MEC.RM.Admin.SubordinatePassThru	Bool
				TimeOutPreset	DeviceName.MEC.RM.Admin.TimeOutPreset	Int(16 bit)
				ReleaseDelay	DeviceName.MEC.RM.Admin.ReleaseDelay	Int(16 bit)
				ModuleID	DeviceName.MEC.RM.Admin.ModuleID	Int(16 bit)
				AsyncPreset	DeviceName.MEC.RM.Admin.AsyncPreset	Int(16 bit)
		CRI			DeviceName.MEC.CRI	CommandRequestInput
			Command		DeviceName.MEC.CRI.Command	Int (32 bit)
			Status		DeviceName.MEC.CRI.Status	Int (16 bit)
		CVI			DeviceName.MEC.CVI	ControlVariable
			XXXXXXX		DeviceName.MEC.CVI.XXXXXXX	Any
		CRO			DeviceName.MEC.CRO	CommandRequestOutput
			Command		DeviceName.MEC.CRO.Command	Int (32 bit)
			Status		DeviceName.MEC.CRO.Status	Int (16 bit)
		CVO			DeviceName.MEC.CVO	ControlVariable
			XXXXXXX		DeviceName.MEC.CVO.XXXXXXX	Any
		FM			DeviceName.MEC.FM	FM
			Admin		DeviceName.MEC.FM.Admin	Admin
				ValidModuleID[#]	DeviceName.MEC.FM.Admin.ValidModuleID[#]	Array of Int (16 bit)
		Status			DeviceName.MEC.Status	Status
			AlarmPresent		DeviceName.MEC.Status.AlarmPresent	Bool
			AlarmAcknowledged		DeviceName.MEC.Status.AlarmAcknowledged	Bool
			Reset		DeviceName.MEC.Status.Reset	Bool
			ModeCurrent		DeviceName.MEC.Status.ModeCurrent	Int(16 bit)
			ModeRequested		DeviceName.MEC.Status.ModeRequested	Int(16 bit)
			RequestStatus		DeviceName.MEC.Status.RequestStatus	Int(16 bit)
		Reset			DeviceName.MEC.Reset	Reset
			ResetRequest		DeviceName.MEC.Reset.ResetRequest	Int(32 bit)

Feltkode ændret

DeviceName					DeviceName	Data Type
	MEC				DeviceName.MEC	S8805
		RRI			DeviceName.MEC.RRI	Supervisory
			Command		DeviceName.MEC.RRI.Command	Command
				RequestID	DeviceName.MEC.RRI.Command.RequestID	Int (16 bit)
				ResourceID	DeviceName.MEC.RRI.Command.ResourceID	Int (32 bit)
			Status		DeviceName.MEC.RRI.Status	Status
				RequestID	DeviceName.MEC.RRI.Status.RequestID	Int(16 bit)
				ResourceID	DeviceName.MEC.RRI.Status.ResourceID	Int(16 bit)
				RequestStatus	DeviceName.MEC.RRI.Status.RequestStatus	Int(16 bit)
				LastResourceID	DeviceName.MEC.RRI.Status.LastResourceID	Int(32 bit)
				LastRequestStatus	DeviceName.MEC.RRI.Status.LastRequestStatus	Int(16 bit)
		RRO			DeviceName.MEC.RRO	Subordinate
			Command		DeviceName.MEC.RRO.Command	Command
				RequestID	DeviceName.MEC.RRO.Command.RequestID	Int (16 bit)
				ResourceID	DeviceName.MEC.RRO.Command.ResourceID	Int (32 bit)
			Status		DeviceName.MEC.RRO.Status	Status
				RequestID	DeviceName.MEC.RRO.Status.RequestID	Int(16 bit)
				ResourceID	DeviceName.MEC.RRO.Status.ResourceID	Int(16 bit)
				RequestStatus	DeviceName.MEC.RRO.Status.RequestStatus	Int(16 bit)
		RM			DeviceName.MEC.RM	RM
			Admin		DeviceName.MEC.RM.Admin	Admin
				ValidRequestID[#]	DeviceName.MEC.RM.Admin.ValidRequestID[#]	Array of Int (16 bit)
				SubordinatePassThru	DeviceName.MEC.RM.Admin.SubordinatePassThru	Bool
				TimeOutPreset	DeviceName.MEC.RM.Admin.TimeOutPreset	Int(16 bit)
				ReleaseDelay	DeviceName.MEC.RM.Admin.ReleaseDelay	Int(16 bit)
				ModuleID	DeviceName.MEC.RM.Admin.ModuleID	Int(16 bit)
				AsyncPreset	DeviceName.MEC.RM.Admin.AsyncPreset	Int(16 bit)
				AllowChangeNoRelease	DeviceName.MEC.RM.Admin.AllowChangeNoRelease	Bool
		CRI			DeviceName.MEC.CRI	CommandRequestInput
			Command		DeviceName.MEC.CRI.Command	Int (32 bit)
			Status		DeviceName.MEC.CRI.Status	Int (16 bit)
			LastCommand		DeviceName.MEC.CRI.LastCommand	Int (32 bit)
			LastCommandStatus		DeviceName.MEC.CRI.LastCommandStatus	Int (16 bit)
		CVI			DeviceName.MEC.CVI	ControlVariable
			XXXXXXX		DeviceName.MEC.CVI.XXXXXXX	Any
		CRO			DeviceName.MEC.CRO	CommandRequestOutput
			Command		DeviceName.MEC.CRO.Command	Int (32 bit)
			Status		DeviceName.MEC.CRO.Status	Int (16 bit)
		CVO			DeviceName.MEC.CVO	ControlVariable
			XXXXXXX		DeviceName.MEC.CVO.XXXXXXX	Any
		FM			DeviceName.MEC.FM	FM
			Admin		DeviceName.MEC.FM.Admin	Admin
				ValidModuleID[#]	DeviceName.MEC.FM.Admin.ValidModuleID[#]	Array of Int (16 bit)
			Status		DeviceName.MEC.Status	Status
			AlarmPresent		DeviceName.MEC.Status.AlarmPresent	Bool
			AlarmAcknowledged		DeviceName.MEC.Status.AlarmAcknowledged	Bool
			Reset		DeviceName.MEC.Status.Reset	Bool
			ModeCurrent		DeviceName.MEC.Status.ModeCurrent	Int(16 bit)
			ModeRequested		DeviceName.MEC.Status.ModeRequested	Int(16 bit)
			RequestStatus		DeviceName.MEC.Status.RequestStatus	Int(16 bit)
		Reset			DeviceName.MEC.Reset	Reset
			ResetRequest		DeviceName.MEC.Reset.ResetRequest	Int(32 bit)

Feltkode ændret

Completeness, compliance and conformance

6 Completeness

The number of object models, objects and attributes supported, as defined in Clause 6 shall determine the degree of completeness of a specification or application.

6.1 Compliance

Any assessment of the degree of compliance of a specification shall be qualified by the following:

1. The use of object names
2. The use of the attributes for each supported object
3. A statement of the degree to which they then conform partially or totally to definitions and attribute names;

In the event of partial compliance, areas of non-compliance shall be explicitly identified.

6.2 Conformance

Any assessment of the degree of conformance of an application shall be qualified by the following:

1. Documentation of the object models and objects, as listed in Clause 5.6 through Clause 5.19, conformed to;
2. Documentation of the attributes conformed to;
3. A statement of the mapping of the application's attributes and object names to the objects and attributes listed in part of the standard.

In the event of partial conformance, areas of non-conformance shall be explicitly identified.

Any additional batch production record objects and attributes supported by an application should be explicitly identified as extensions to the standard format.

6.3 Extending the object model

The objects in a batch production record represent a wide range of data types and formats intended to cover common requirements in the industry. In order to accommodate industry, business and application requirements in the future it may be necessary to add new objects and/or attributes to an implementation of the batch production record.

When objects or attributes are added to an implementation of the object model the following rules shall be followed to be in compliance with this part.

1. New objects and attributes may be created provided their names clearly identify them as different from objects and attributes defined in clause ~~Fejl! Henvisningskilde ikke fundet.~~Error! Reference source not found. of this part;
2. Existing objects and attributes shall not be redefined.

Feltkode ændret

Annex A Working Notes – Definitions

Shared Concepts (common terms for Making and Packing, but not explicitly defined in S88)

Term	Make2Pack Description
Command	A signal to a component to change mode or state. Commands are generally momentary in nature and latched by the receiving component and reset when the command has been executed.
Control Component	Control software part of a Unit, Equipment Module or Control Module.
Cycling	The repetitive motion of a machine for discrete manufacturing. The machine typically performs 1 cycle for each processed item.
Device	An apparatus for performing a prescribed function [Definition from ANSI/ISA-51.1 – 1979 (R1993)] Used here to describe elements of the physical environment that control systems connect to/control.
External Control	Control that acts externally to a control component.
Internal Control	Control that acts internally within a control component. This will only apply to procedural elements in automatic mode.
State	Defined in S88.01; hold pending changes there. The operating conditions of a component, whether it is a lower level control module or a higher level procedural component. States are only relevant if the component is in automatic or semiautomatic mode. A state can be stationary or transient. A transient state is a temporary state that the component is in while moving from one quiescent state to another. Examples of quiescent states are: • Stopped • Producing • Cleaning • Cycling • Halted Examples of transient states are: • Starting • Stopping S88: <i>state: The condition of an equipment entity or of a procedural element at a given time.</i> <i>NOTE — The number of possible states and their names vary for equipment and for procedural elements.</i>

S88 Physical Model (parallel terms for Making and Packing) – Non-Normative

Batch term	Packaging term	Make2Pack Description
Equipment Entity	Mechatronics	Object oriented machine design. In object oriented machine design an object is the combination of hardware and logic.
Process Cell	Production Line (per S95 maybe)	A collection of one or more machines, linked together, to perform one or multiple tasks of the process for one or more products in a defined sequence. • Continuous Process (e.g. forming line in the food industry) • Converting Line (e.g. paper, fibers) • Discrete Manufacturing (e.g. assembly) • Packaging Line (from filling to secondary and tertiary packaging) S88: A process cell contains all of the units, equipment modules, and control modules required to make one or more batches/lots.
Unit	Machine	In packaging, the unit corresponds to the logical grouping of mechanical and electrical assemblies that historically have been called machines. The term unit may apply to single function machine (filler, capper) or a multifunctional machine (monoblock filler/capper or any other configuration that combines functions within a single machine frame and control system). A multifunctional machine/unit can perform some or all functions of a packaging line, corresponding to process cell, that perform some or all of the functions of primary, secondary and tertiary packaging. A multifunctional machine may be logically broken down into several units corresponding to the individual functions. S88: A unit is made up of equipment modules and control modules. The modules that make up the unit may be configured as part of the unit or may be acquired temporarily to carry out specific tasks.
Equipment Module	Equipment Module	A group of equipment and its associated control located within the context of a machine or unit, designed and or arranged to perform a certain function. This can be equivalent to what is often called a Station in a discrete machine. The equipment module may be made up of control modules and subordinate equipment modules . E.g. the capper station in bottle filling machine. S88: Physically, the equipment module may be made up of control modules and subordinate equipment modules. An equipment module may be part of a unit or a stand-alone equipment grouping within a process cell. If engineered as a stand-alone equipment grouping, it can be an exclusive-use resource or a shared-use resource.

Feltkode ændret

Batch term	Packaging term	Make2Pack Description
Control Module	Control Module	A Control Module is the lowest module in a physical model breakdown of a unit. The term Control Module relates to the combination of (a) physical device(s) and the lowest level control component that controls this(these) device(s) to carry out a physical process action. There may be control modules without directly associated physical devices. These control modules coordinate/supervise/sequence other control modules. NOTE: The use of the term control module to describe the supervisory/sequencing/coordinating functions is proving confusing and difficult to convey the concepts. Needs further consideration. Make2Pack examples of Control Modules: Servo Conveyor Pneumatic Cylinder with feed-back <i>S88: A control module is typically a collection of sensors, actuators, other control modules, and associated processing equipment that, from the point of view of control, is operated as a single entity. A control module can also be made up of other control modules. For example, a header control module could be defined as a combination of several on/off automatic block valve control modules.</i>
Compound Control Module	n/a	Control Module that coordinates/sequences lower level Control Modules. The Compound Control Module in general does not connect directly to physical devices like lower level Control Modules.
Compound Equipment Module	n/a	Equipment Module that coordinates/sequences lower level Equipment Modules. The Compound Equipment Module, like lower level Equipment Modules, does not connect directly to physical devices.
Unit/Machine Control Element (placeholder for Coordination Control element)	n/a	Supervisory Control Element on the unit level that handles general unit functionality, (e.g. reset, mode propagation, alarm management)

S88 Procedural Model (Making terms from S88 part 1 adopted by Packing)

Procedural Control: *Control that directs equipment-oriented actions to take place in an ordered sequence in order to carry out some process-oriented task.*

Recipe: *The necessary set of information that uniquely defines the production requirements for a specific product.*

NOTE — *There are four types of recipes defined in this standard: general, site, master, and control.*

Feltkode ændret

Control Recipe: *A type of recipe which, through its execution, defines the manufacture of a single batch of a specific product.*

NOTE 1 — *The control recipe may not be omitted from the recipe types model.*

Equipment Control: *The equipment-specific functionality that provides the actual control capability for an equipment entity, including procedural, basic, and coordination control, and that is not part of the recipe. Code in equipment control that issues commands to underlying control components in order to affect the process as intended. A procedural element is only active while carrying out its task, then becomes idle.*

Procedural Control Element	Recipe Procedural Element	Equipment Procedural Element	Making Examples	Packing Examples
Procedure: <i>The sequencing strategy for carrying out a process, defined in terms of an ordered set of unit procedures.</i> NOTE — <i>In general, it refers to the strategy for making a batch within a process cell. It may also refer to a process that does not result in the production of product, such as a clean-in-place procedure.</i>	Recipe Procedure: <i>The part of a recipe that defines the [sequencing] strategy for producing a batch.</i>	Equipment Procedure: <i>A procedure that is part of equipment control.</i> NOTE: An equipment procedure has the same function as a recipe procedure.	"Make PVC"	The Procedure for a discrete Production Line relates to the higher level process of producing goods, e. g. how to get the line ready, how to get into the producing state or how to clean out the line.
Unit Procedure: <i>A unit procedure consists of an ordered set of operations that cause a contiguous production sequence to take place within a unit. Only one unit procedure may be active in a unit at any time.</i>	Recipe Unit Procedure: <i>A unit procedure that is part of a recipe procedure in a master or control recipe.</i>	Equipment Unit Procedure: <i>A unit procedure that is part of equipment control.</i>	—Polymerize VCM. —Recover residual VCM —Dry PVC	Producing Cleanout Trimset Setup Maintenance Jogging

Feltkode ændret

Procedural Control Element	Recipe Procedural Element	Equipment Procedural Element	Making Examples	Packing Examples
Operation: <i>An operation is an ordered set of phases that defines a major processing sequence that takes the material being processed from one state to another, usually involving a chemical or physical change.</i> <i>Operation boundaries should be located at points in the procedure where normal processing can safely be suspended. Only one operation should be active in a unit at any time.</i>	Recipe Operation: <i>An operation that is part of a recipe procedure in a master or control recipe.</i>	Equipment Operation: <i>An operation that is part of equipment control.</i>	• <i>Preparation : Pull a vacuum on the reactor and coat the walls with antifoulant.</i> • <i>Charge: Add demineralized water and surfactants.</i> — • <i>React: Add VCM and catalyst, heat, and wait for the reactor pressure to drop.</i>	<i>Starting</i> <i>Aborting</i> <i>Holding</i> <i>Suspending</i> <i>Held</i>
Phase: <i>The smallest element of procedural control that can accomplish a process-oriented task is a phase. Phases are implemented by principle control within equipment.</i> NOTE: A procedural element can only have one master at a time.	Recipe Phase: <i>A phase that is part of a recipe procedure in a master or control recipe.</i>	Equipment Phase: <i>A phase that is part of equipment control.</i> NOTE: Equipment phases commanded by recipe phases require procedural states similar to those of the S88 example, but it is possible to have equipment phases that are directed by other control components based on different state models.		

Kommentar [RD1]: Move to future section on Recipe to Equipment linking.

Feltkode ændret

Procedural Control Element	Recipe Procedural Element	Equipment Procedural Element	Making Examples	Packing Examples
Principle Control: <i>Principle control is responsible for achieving the control actions of equipment procedural elements (e.g. the phase). Principle control is sequential in nature and has quiescent state(s) which typically require an external command to become active or to move from state to state.</i>	n/a	Principle Control: Code in equipment control that issues commands to other control components in order to affect the process as intended. An equipment phase will only be active while it is carrying out its task. When complete the equipment phase becomes idle.		Equipment Phase Logic

Modes of Procedural Control (terminology differences for Making and Packing)

Making term	Packing term	Make2Pack Description
Mode (Procedural Mode)	Mode (Unit Control Mode)	Modes apply to 3 different types of entities Unit Modes: define the operation of a machine in terms of what it does. Examples might be: - Producing product 1 - Producing product 2 - Clean-out - Jogging - Single Step Unit modes are machine dependent and defined by the machine designers. - Procedure Modes: define how procedural elements respond to command inputs. Only 3 different procedural modes should be used: - Automatic - Semiautomatic - Manual Mode for Equipment Entities: Equipment Entities are the lowest level control component and the

Feltkode ændret

Making term	Packing term	Make2Pack Description
		only type of component that connect to the physical devices. Equipment entities do not have procedural control. There are only 2 modes for equipment entities: Manual: The logic within the control component is not active. The physical device is controlled directly by the operator. Automatic: The logic within the control component is active. The logic controls the physical device based on inputs from other control components. The mode definitions for Procedure Modes and Equipment Entity Modes correspond to the definitions in S88 Part 1. S88: mode: <i>The manner in which the transition of sequential functions are carried out within a procedural element or the accessibility for manipulating the states of equipment entities manually or by other types of control.</i>
Unit Procedure	Mode	Procedural element for the highest level procedural control within a Unit. S88: unit procedure: <i>A strategy for carrying out a contiguous process within a unit. It consists of contiguous operations and the algorithm necessary for the initiation, organization, and control of those operations.</i> <i>A unit procedure consists of an ordered set of operations that cause a contiguous production sequence to take place within a unit. Only one operation is presumed to be active in a unit at any time. An operation is carried to completion in a single unit. However, multiple unit procedures of one procedure may run concurrently, each in different units. Examples of unit procedures include the following:</i> — Polymerize VCM. — Recover residual VCM. — Dry PVC.
Automatic Mode	everything	The component does not respond to human commands, except as explicitly required by the automation sequence. Responds only to commands from other logic while executing its own internal logic.
Semiautomatic Mode	n/a	The component responds to human commands in lieu of commands from other logic while still executing its own internal logic. Procedural components will typically advance to the next step in its sequence at human command. In some applications this mode may be known as Maintenance.
Manual Mode	n/a	The component responds to human commands to activate its outputs and does not execute its own internal logic. For procedural components manual mode overrides the procedure and human commands affect the process directly. Manual mode is optional.

Kommentar [RD2]: Duplicate entry to flag correlation to Packing "Mode"

Kommentar [RD3]: Update per S88.01

Kommentar [RD4]: Update per S88.01

Kommentar [RD5]: Update per S88.01

Feltkode ændret

Annex B Questions and Answers

Feltkode ændret

Annex C PID Example

C.1 PID Implementation

The purpose of this example is to show how the Control Shell can be used for a PID controller, implemented as a compound control module.

C.2 MEC PID Control Module

Figure 11 illustrates the MEC elements that make up a PID Control Module. It contains a Resource Manager (RM), Functional Manager (FM), and a Functional Strategy (FS). The FS illustrates that the FS is made up of contained or referenced Control Modules.

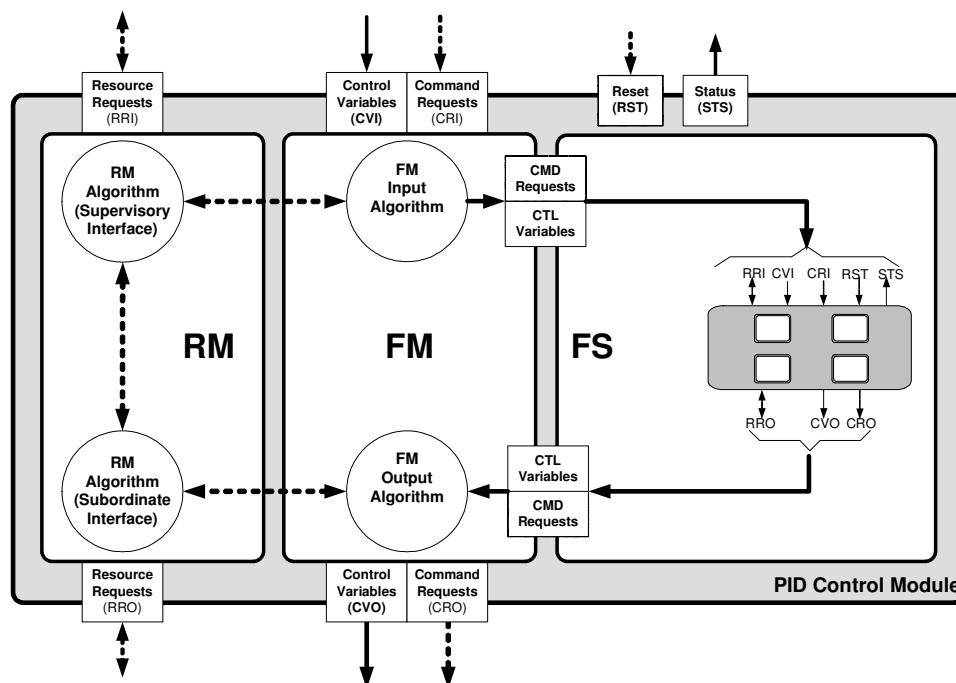


Figure 11 - MEC PID Control Module

Figure 12 illustrates the Functional Strategy of the PID Control Module. It is made up of Control Modules for Analog Inputs, Virtual Inputs, Analog Outputs, and Control Shell around a PID Function Block.

Virtual Inputs are inputs from HMI systems or other equipment entities.

All of the inputs to the basic function block have a Control Shell wrapper. The output from the basic function block has a Control Shell wrapper, and the basic PID function block also has a Control Shell wrapper.

Feltkode ændret

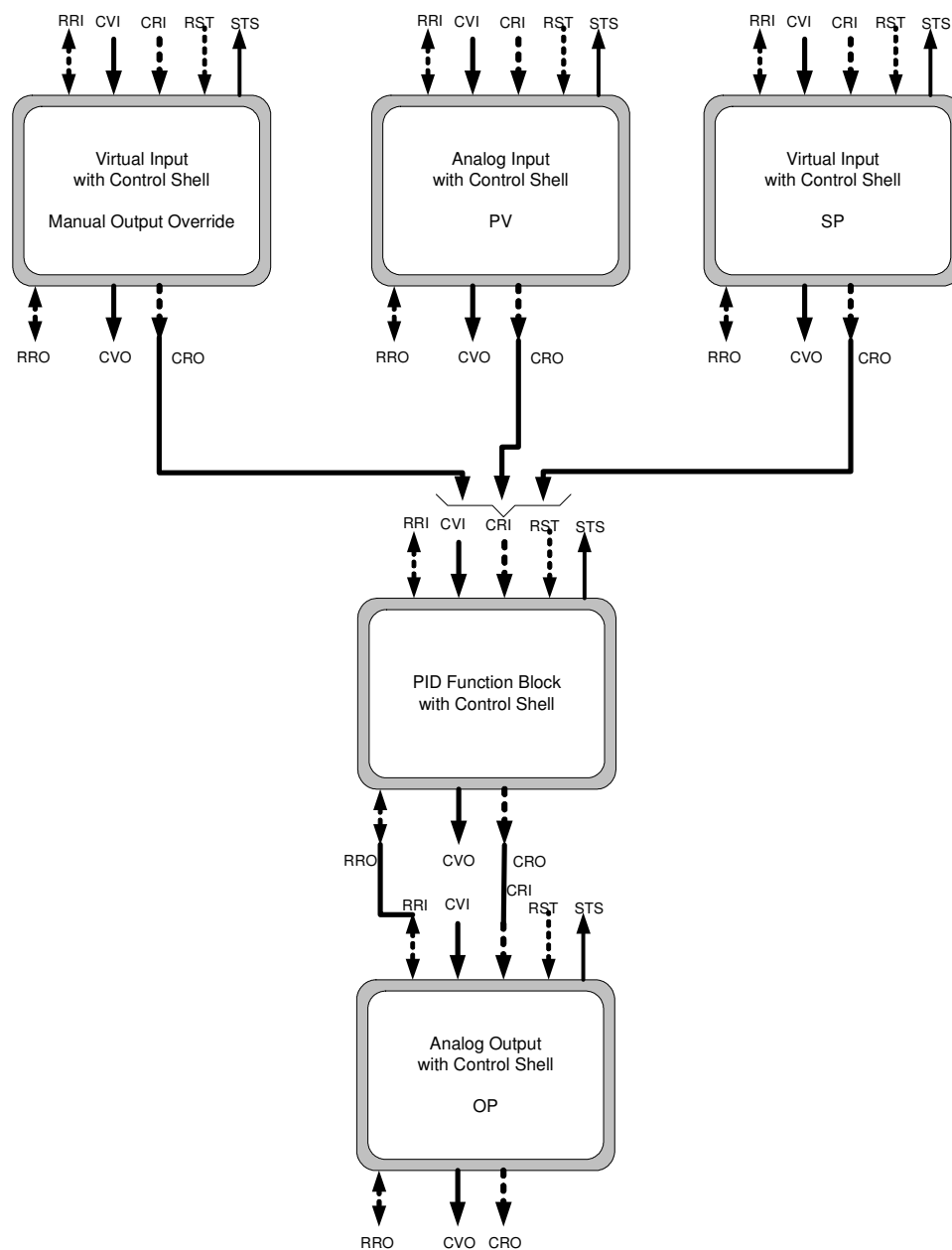


Figure 12 – PID basic controller in MEC format

C.3 PID Control Element with Control Shell

Feltkode ændret

[Figure 13](#)[Figure 12](#) illustrates the encapsulation of the PID function block with a control shell. This provides the resource management and switching capability contained in the shell with the functional strategy defined in the IEC 61499 PID Function Block.

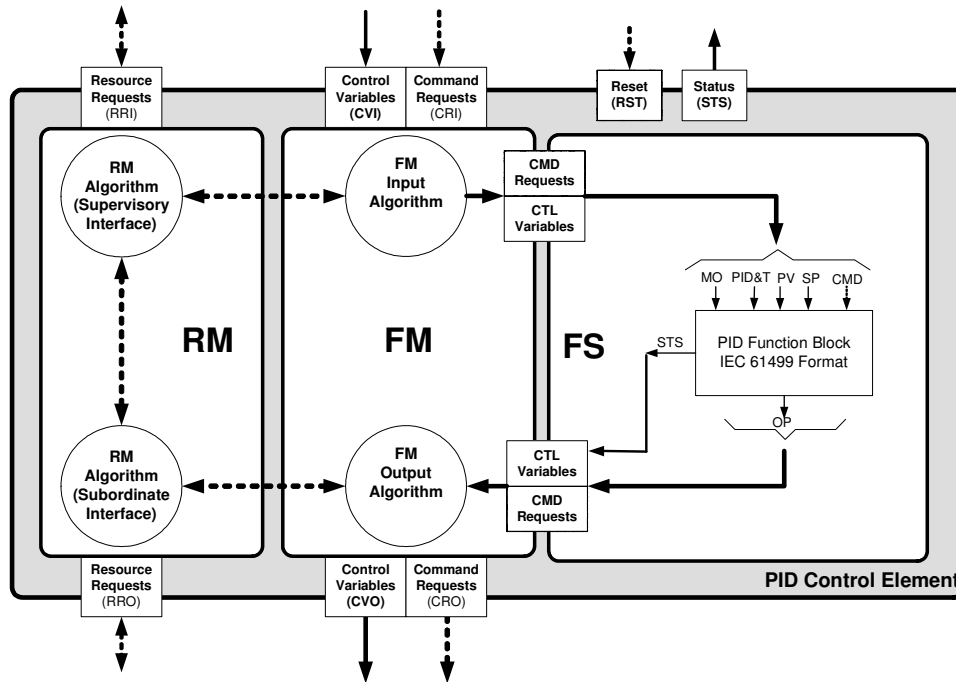


Figure 13 - PID Control Element

[Figure 14](#)[Figure 13](#) below shows Functional Strategy represented using an IEC 61499-1 Function Block for a simple PID control with typical PID variables exposed. [Figure 14](#)[Figure 13](#) also shows a typical implementation for Auto/ Manual functionality that may be included with the function block.

Feltkode ændret

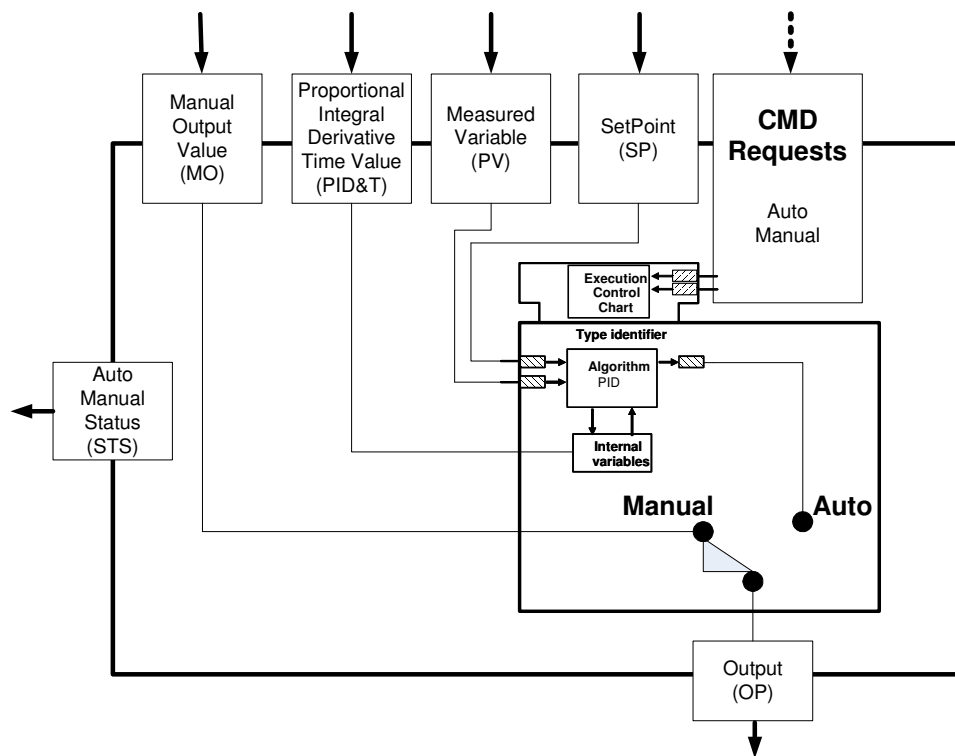


Figure 14 – PID Functional Strategy

C.4 MEC Analog Input Functional Strategy for PID Measured Variable Input

Feltkode ændret

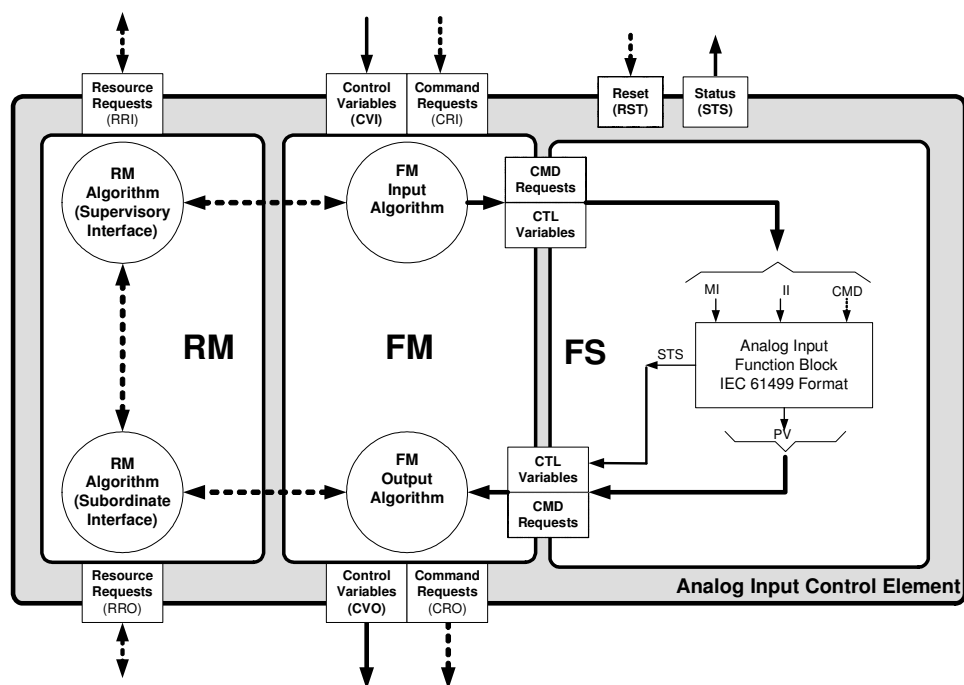


Figure 15 - MEC Analog Input

Figure 15 below shows the IEC 61499-1 Function Block for a simple Analog Input control with typical Analog Input variables exposed. This Analog Input is designed to connect to real inputs in the IO system. Figure 15 also shows the classical implementation for Auto / Manual functionality for an Analog Input.

Feltkode ændret

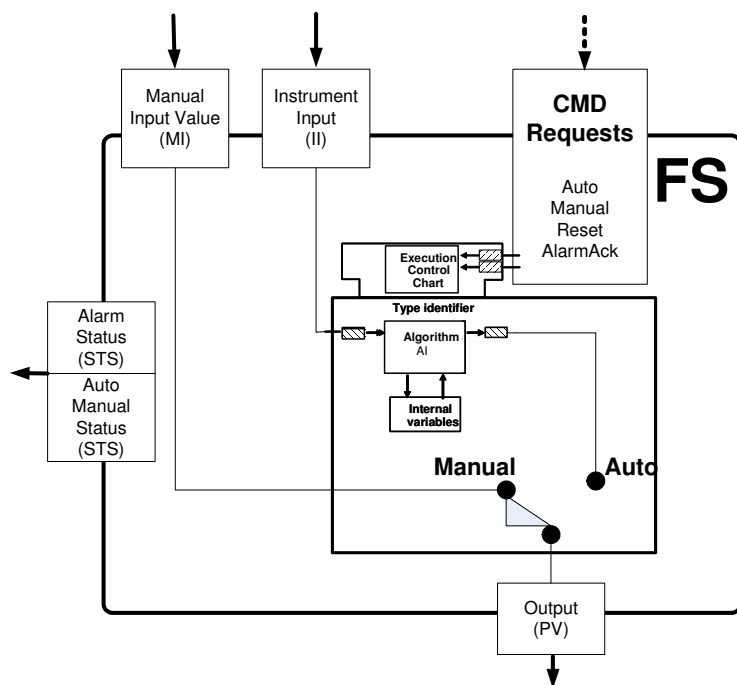


Figure 16 - Analog Input

C.5 MEC Virtual Input Functional Strategy for PID Manual Output and PID Setpoint Inputs

Feltkode ændret

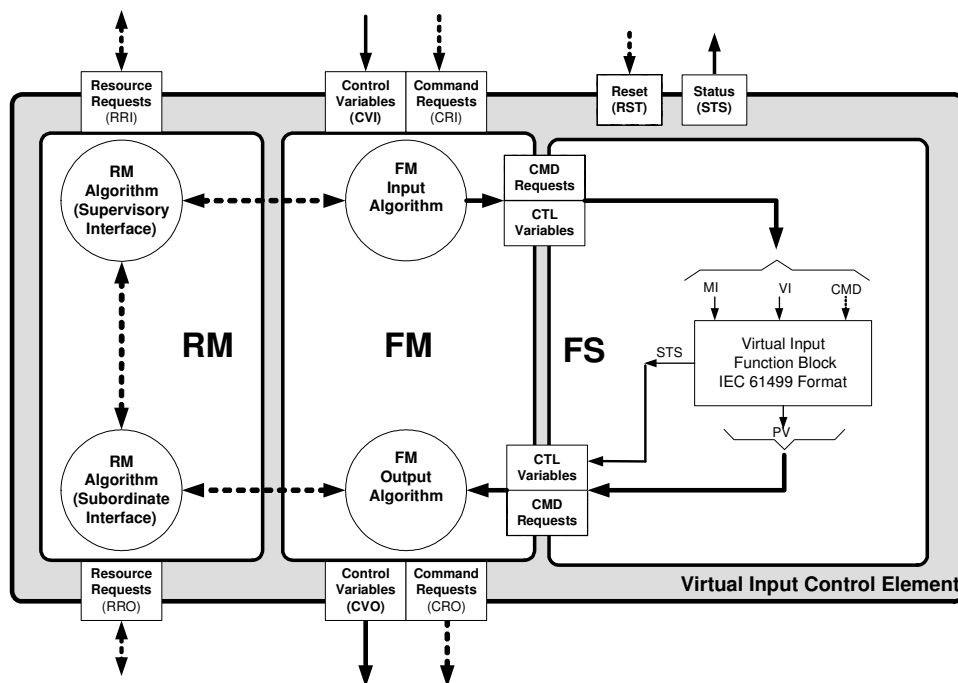


Figure 17 – MEC Virtual Input

Figure 18 below shows the IEC 61499-1 Function Block for a simple Virtual Input control with typical Virtual Input variables exposed. This Virtual Input is designed to connect to outputs from other control components. Figure 18 also shows the classical implementation for Auto / Manual functionality for an Virtual Input.

Feltkode ændret

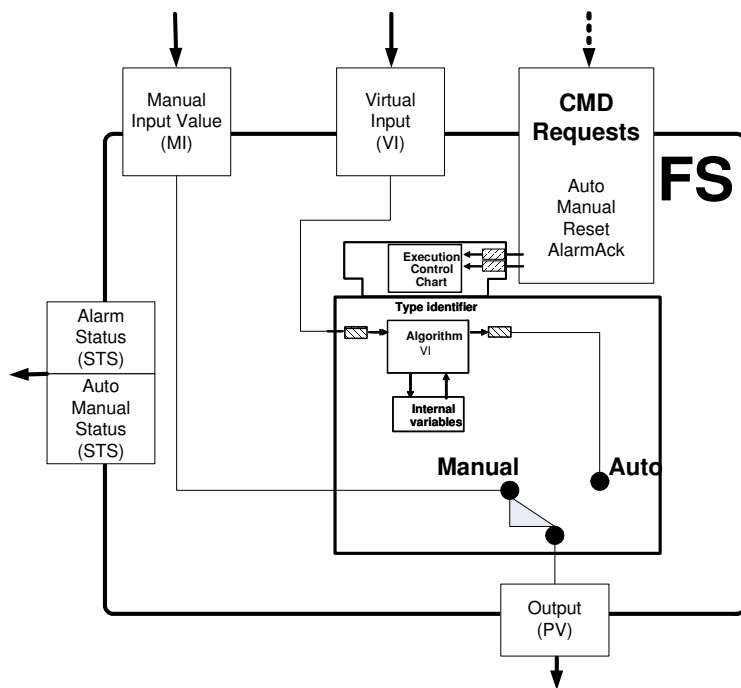


Figure 18 - Virtual Input

C.6 MEC Analog Output Functional Strategy for PID Output

Feltkode ændret

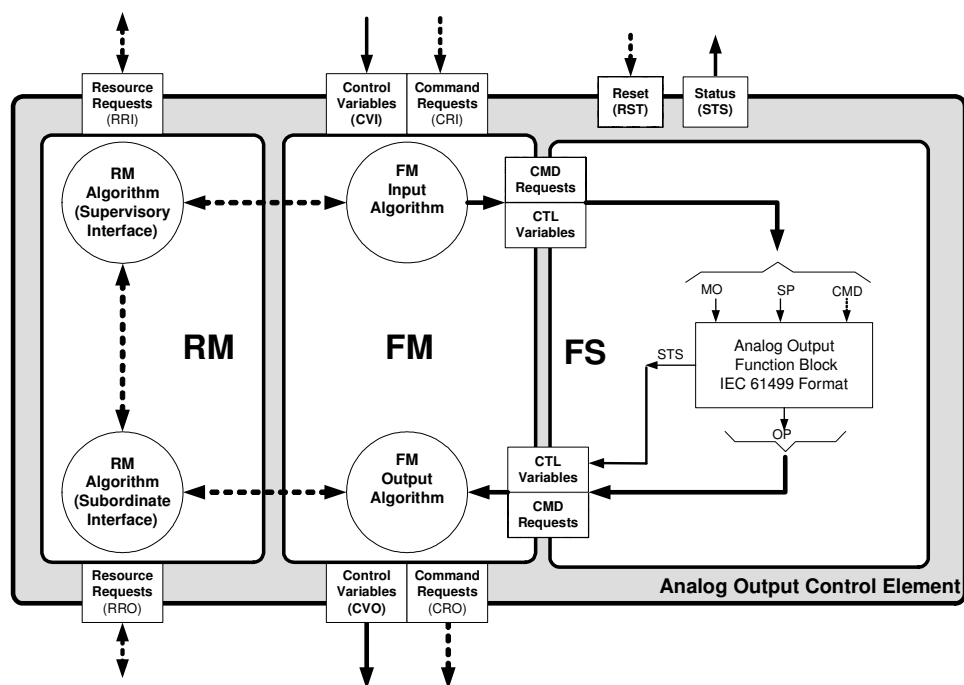


Figure 19 – MEC Analog Output

Figure 20 below shows the IEC 61499-1 Function Block for a simple Analog Output control with typical Analog Output variables exposed. This Analog Output is designed to connect to outputs in the IO system. Figure 20 also shows the classical implementation for Auto / Manual functionality for an Analog Output.

Feltkode ændret

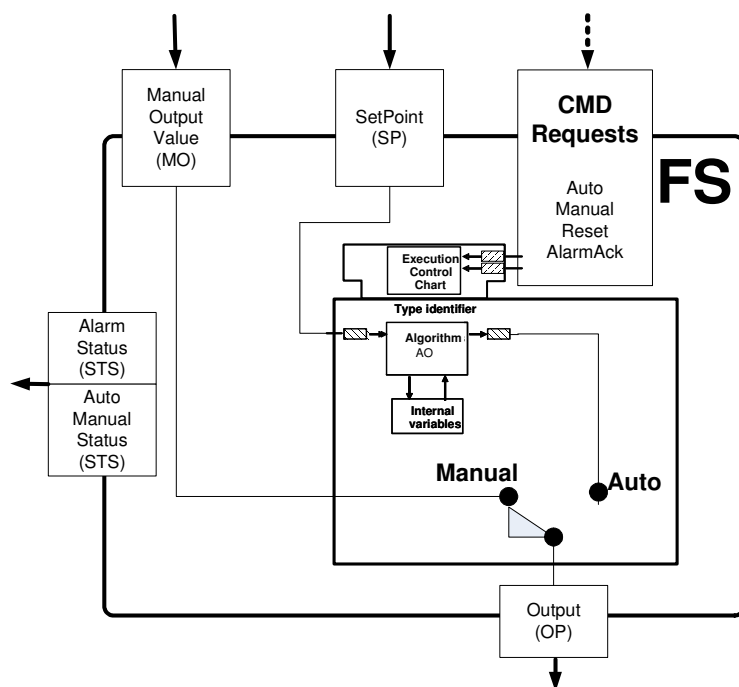


Figure 20 - Analog Output

Feltkode ændret

Annex D - Cascaded PID Implementation

6.4 MEC Cascaded PID Implementation

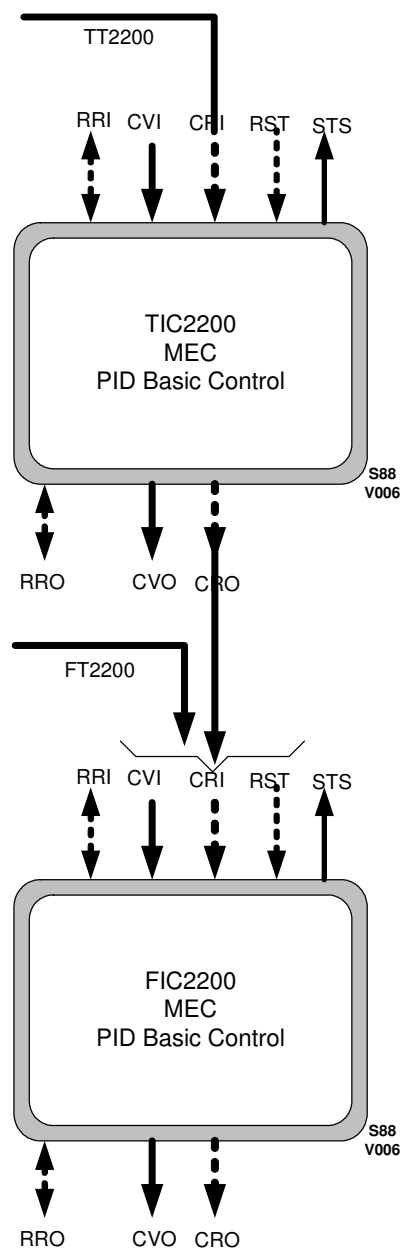


Figure 13

Feltkode ændret

Annex E– Material Transfer Model

Editor's Note: This section to be updated by Dave Chappell.

1. PURPOSE

The purpose of this document is to describe Material Transfer (Material Feed) Equipment Modules, relative to their deployment in the manufacturing areas of MAKING and PACKING.

2. SCOPE

The scope of this document is limited to the discussion of relevant DEFINITIONS, PROCESS DISCRPTIONS and TERMINOLOGY necessary for the deployment of BEST PRACTICE, BASIC PROCESS MEASUREMENT & CONTROL techniques. The discussion of the deployment of ADVANCED PROCESS CONTROL techniques is only briefly introduced.

3. THE DEFINITION & GOALS OF A MATERIAL TRANSFER

The definition of a material transfer is the movement of a specified amount of material (solid, powder, liquid or slurry) from one location to another.

Different regions of the world have different names for this process phase. It is also know as a **material feed, a dose, a charge, a discharge, an ingredient addition, a fill, a feed or a dump**. It is likely there are other terms in use not mentioned in this list. In this document we will use the terms Material Transfers or Material Feeds.

The goal of a Material Feed is to safely feed the exact amount of material required, in the minimum amount of time, every time a material feed is required.

4. USE OF MATERIAL TRANSFERS IN MAKING & PACKING

Material Feed Equipment Modules are widely deployed in process automation, being extensively used in each of the following 5 applications. For the sake of clarity, these 5 applications are described below as their application names are sometimes mistakenly used to describe a Material Feed.

- **FILLING**

The SINGLE transfer (movement) of a specified amount of product from one single location to another location

- **DOSING**

The SINGLE transfer (movement) of a specified amount of product from one location into a continuous process

- **FORMULATION**

Feltkode ændret

MULTIPLE transfers (movements) of specified amounts of products from various locations into a single location

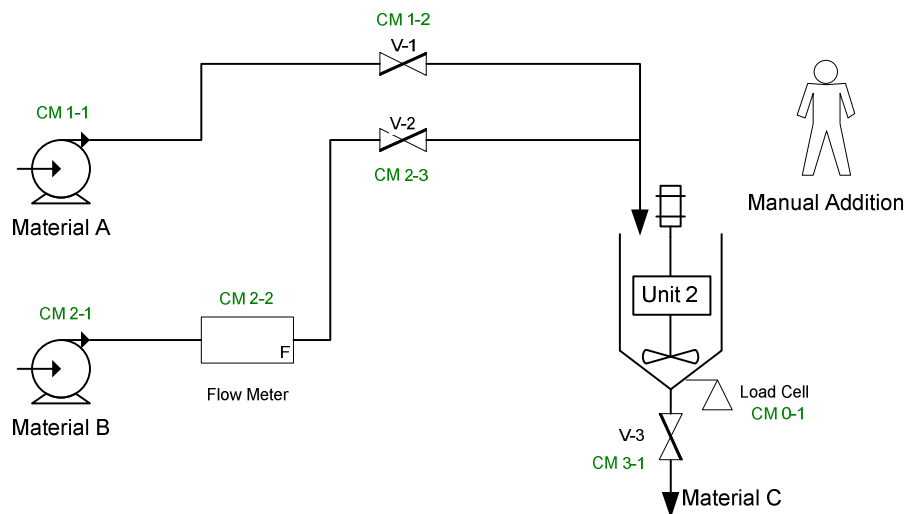
- **BLENDING**

MULTIPLE transfers (movements) of specified amounts of products from various locations into a single location plus a single ADDITIONAL process phase - mixing

- **BATCHING**

MULTIPLE transfers (movements) of specified amounts of products from various locations into a single location plus multiple ADDITIONAL process phases – heating, cooling, wait, mix, agitate, dump etc.

5. MATERIAL TRANSFERS - 5 TYPES



Material Feed Types

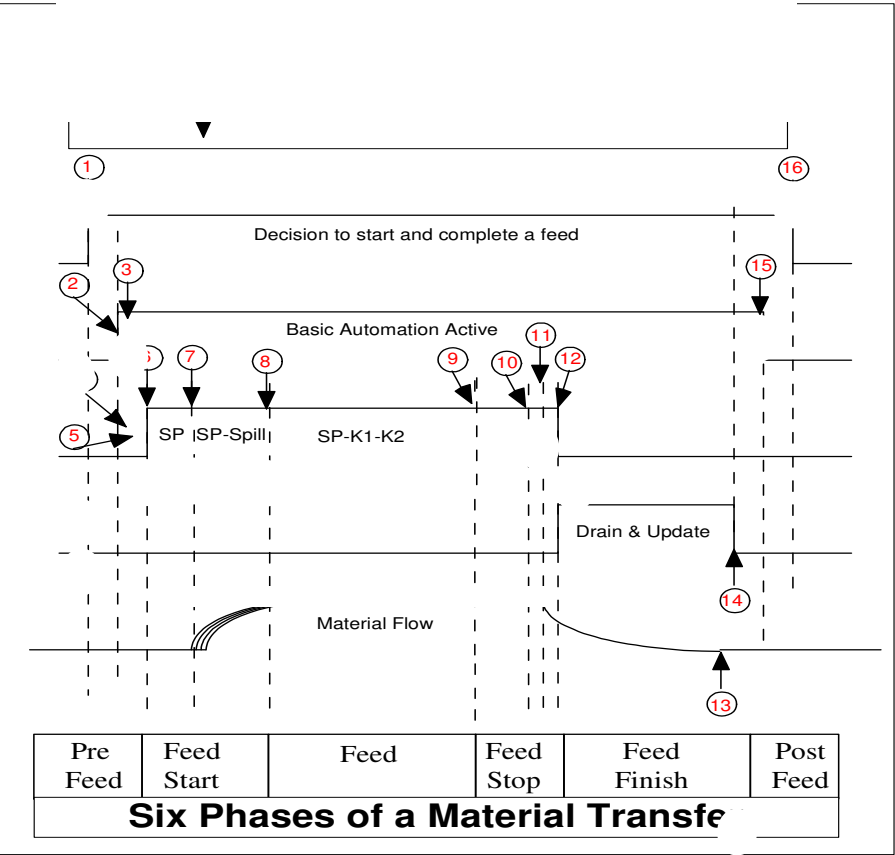
The diagram above illustrates a simple material transfer delivery system. Material feeds may be done with using a number of measurement devices (the above diagram shows a load cell and a flow meter) and many different FCE (Final Control Elements – in the case above the diag shows valves). There are 5 types of Material Feeds.

Feltkode ændret

- Gain In Weight
 - Loss In Weight
 - Flow Meter Feed
 - Dump To Empty
 - Hand Add
- Material A fed through V-1 into Unit 2 with defined target weight
 - Material C fed through V-3 with target weight
 - Material B fed through V-2 with defined target weight
 - Material C through V-3 no specified target weight
 - Material D by manual (human) addition

Feltkode ændret

6. MATERIAL TRANSFERS - THE 6 OPERATIONAL STAGES



THE 6 STAGES OF A MATERIAL TRANSFER

Overview

There are 6 stages in a material feed.

- Pre Feed
- Feed Start
- Feed
- Feed Stop
- Feed Finish

Each of the six stages of a BASIC material feed are explained below.

Stage 1 - Pre Feed

A decision to start a material transfer (feed) occurs. The Material Transfer Equipment Module receives a request. Everything required to perform a successful material transfer start is verified. If any situation is detected which would prevent the material feed from successfully starting, the operation is aborted. The operator is informed of the failure and the reason/s.

During this stage:

1. Material Transfer data/values (e.g. feed target, feed set-point, tolerances etc.) are presented to Control.
2. Control (the Material Feed Equipment Module - consisting of supervisory material transfer control and set-point control) activates upon receipt of the request.
3. Control (supervisory) evaluates the request, and completes any system checks. If all is okay, the request continues to be processed.
4. Control (supervisory) acquires a stable instrument reading.
5. Control (supervisory) activates the transfer.
6. Control verifies that the material transfer activated the FCE (Final Control Element).

Feltkode ændret

...move into Stage 2

Stage 2 - Feed Start

Automation is waiting to verify that the material transfer has started by monitoring flow rate. If this fails to happen after a reasonable time, the automation aborts the feed operation and informs the operator of the failure.

During this stage:

7. Control (supervisory) verifies material is transferring

...move into Stage 3

Stage 3 – Feed

The material has established a flow rate. This stage stays active until very close to feed completion.

During this stage:

8. Control (set point) repeatedly compares the measured weight to the set point weight.

...move into Stage 4

Stage 4 – Feed Stop

The material transfer approaches completion and is close to the point where the set point will interrupt the transfer when it detects the set point to be equal to or less than the measured instrument reading.

During this stage:

Feltkode ændret

9. Control (set point) evaluates the when the set point is reached based on the instrument reading, the final control element is de-energized. The material stops transferring.

...move into Stage 5

Stage 5 - Feed Finish

The controller identifies that the final control element has closed, acquires accurate information about the transfer, determines if completion of the operation was within tolerances, sets the success/fail status.

During this stage:

10. Control (supervisory) identifies the closure of the FCE (final control element).

11. Control (supervisory) waits for a stable measurement reading and records the reading.

12. Control (supervisory) compares the data with the tolerance parameters.

13. Control (supervisory) sends a material transfer report

...move into Stage 6

Stage 6 - Post Feed

The operator and/or HOST controller are informed of the transfer completion, and take actions indicated by the resulting information such as tolerance errors. At the end of this stage, the material transfer is complete and the control system can continue.

14. An evaluation is made of the results of the transfer in relation to the overall batch. If actions are required, Operations is informed.

15. The material transfer is determined to be satisfactory or Operations determines it is necessary to continue, accepting any errors.

...end

Feltkode ændret

Feltkode ændret

7. Material Transfer EM – State Diagram

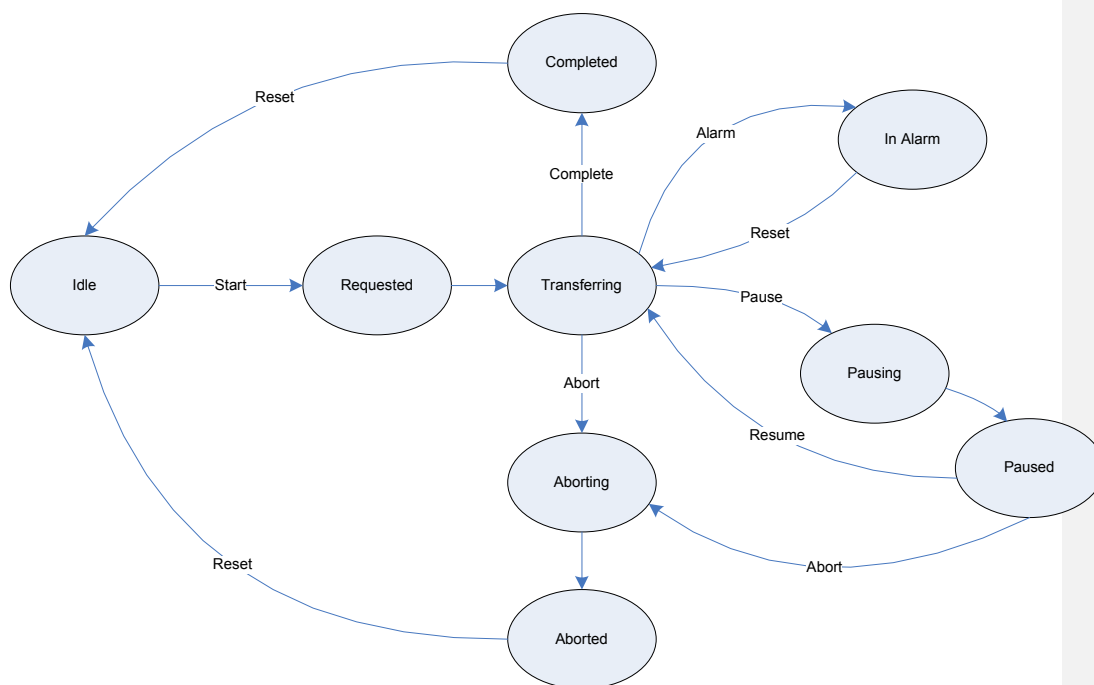


Figure xx.x Material Transfer EM – State Diagram

Kommentar [mkc6]: Rev 1.0 of State Diagram, needs review.
Requested ? Permissives OK?

Feltkode ændret

8. Commands & Status

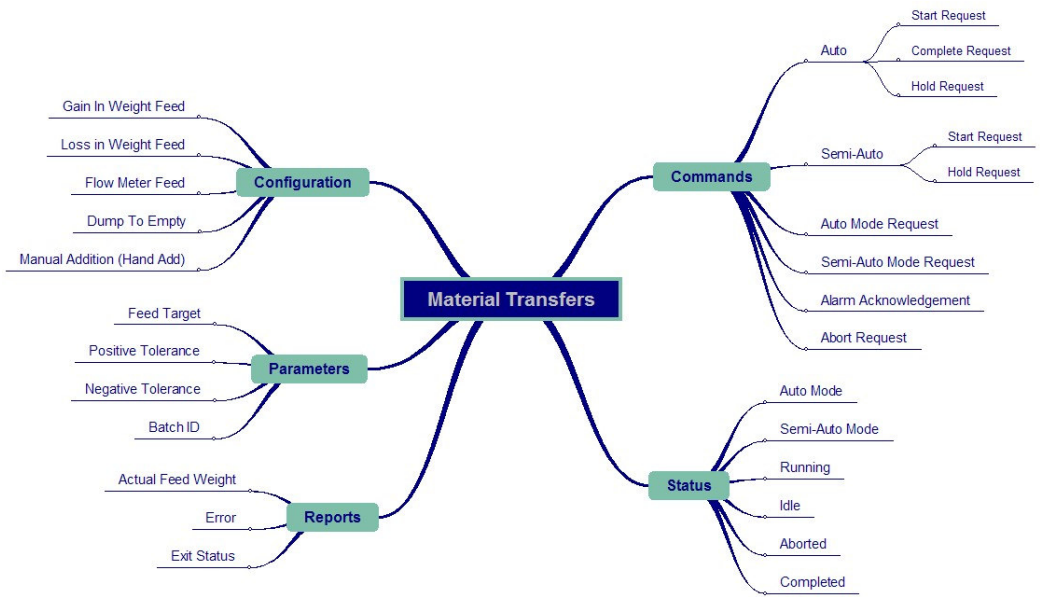


Figure xx.x – Material Transfer EM – Data Structure

Kommentar [mkc7]: Needs to be updated to include Rev 1.2 commands and status

Feltkode ændret

a. COMMANDS

Auto

- Command.Auto.StartRequest
- Command.Auto.CompleteRequest
- Command.Auto.PauseRequest
- Command.Auto.BypassRequest
- Command.Auto.ControlledRequest

Auto Start Request	The request to start the material transfer from the control system.
Complete Request	The request from the control system to complete the material transfer normally.
Auto Pause Request	The Request to Pause the material transfer from the control system.
Auto Bypass Request	
Auto Controlled Request	

Semi Auto

- Command.SemiAuto.StartRequest
- Command.SemiAuto.PauseRequest
- Command.SemiAuto.BypassRequest
- Command.SemiAuto.ControlledRequest

Semi-Auto Start Request	The request to start the material transfer from the operator.
Semi-Auto Pause Request	The Request to Pause the material transfer from the operator.
Semi-Auto Bypass Request	

Feltkode ændret

Semi-Auto Controlled Request

Feltkode ændret

Common

- Command.AutoModeRequest
- Command.SemiAutoModeRequest
- Command.AlarmAcknowledgement
- Command.AbortRequest
- Command.ResetRequest

Auto Mode Request	Request to change mode from Semi-Auto to Auto Mode.
Semi-Auto Request	Request to change mode from Auto to Semi-Auto Mode.
Alarm Acknowledge	The request to Acknowledge Alarms
Abort Request	The request to Abort the material transfer.
Reset Request	The request to disable the Material Transfer and return to the Idle state.

Kommentar [mkc8]: Review the word "disable"

Feltkode ændret

b. STATUS

- Status.Mode.AutoMode
- Status.Mode.SemiAutoMode
- Status.Mode.Bypassed
- Status.Mode.Controlled

- Status.State.Transferring
- Status.State.Idle
- Status.State.Aborted
- Status.State.Aborting
- Status.State.Completed
- Status.State.Paused
- Status.State.Pausing
- Status.State.Requested

- Status..Alarm.AlarmPresent

Status

State

Transferring	The material transfer is Active and Running
Idle	The material transfer is Idle
Aborted	The material transfer has been Aborted
Aborting	
Completed	The material transfer is Complete
Paused	The material transfer is Paused
Pausing	

Feltkode ændret

The material transfer has been Requested

Auto Mode

The material transfer is in Auto Mode

The material transfer is in Semi Auto Mode

Controlled

Alarm Present

The material transfer has an Alarm Present

6/16/20082/25/20082/20/2008ISA Draft 88.00.05 (Draft WD04)

c. PARAMETERS

- Parameters.FeedWeightTarget
- Parameters.PositiveTolerance
- Parameters.NegativeTolerance
- Parameters.BatchId ??

Feed Weight Target	The Feed Weight setpoint for the material transfer. Maybe absolute or incremental.
Positive Tolerance	The upper limit for the actual feed weight to consider this a "Good Feed"
Negative Tolerance	The lower limit for the actual feed weight to consider this a "Good Feed"

d. REPORTS

- Reports.ActualFeedWeight
- Report.Error
- Report.ExistStatus

Actual Feed Weight	The actual amount of material transferred.
Error	The difference between the Actual Feed Weight and the Feed Weight Setpoint. The state of the material transfer when it completed?
Exit Status?	- Completed Normally - Aborted - In Alarm - Paused - Auto/Semi Auto - Bypassed/Controlled

Feltkode ændret

Feltkode ændret

e. CONFIGURATION

- Config.GainInWeight
- Config.LossInWeight
- Config.FlowMeter
- Config.DumpToEmpty
- Config.ManualAddition
- Config.Other

Gain In Weight	The addition of material to a unit. If the target is an incremental setpoint then the target weight would be ADDED to the current weight in the vessel.
Loss In Weight	The removal of material from a unit If the target is an incremental setpoint then the target weight would be SUBTRACTED from the current weight in the vessel.
Flow Meter	The addition of material to a unit through a flow meter Typically this method of material transfer requires interfacing with a device that generates a pulsed output. Additional logic is usually required to reset or capture the current values and then accumulate the pulses during the feed.
Dump To Empty	The removal of all material from a unit Dump to Empty material transfers do not have a target weight associated to them. A dump to empty will continue transferring material until the vessel is empty.
Manual Addition	A non-control system (external) material addition to a unit. A manual addition or external addition is any Gain In Weight (GIW) or Loss In Weight (LIW) that occurs that is outside the boundaries of the control system. The control system maybe aware that the material is being added or removed from the vessel but it is unable to initiate or terminate the feed.

Feltkode ændret

Feltkode ændret

9. Material Transfer Terminology

The following is an example of terminology used in regards to the measurement functions of a Material Feed.

Weighing Terminology

- Gross - explanation to be inserted
- Nett - explanation to be inserted
- Units - explanation to be inserted
- Zero - explanation to be inserted
- Center of Zero - explanation to be inserted
- Under Zero - explanation to be inserted
- Tare - explanation to be inserted
- Clear (Tare) - explanation to be inserted
- Over Capacity - explanation to be inserted
- Motion - explanation to be inserted
- Print - explanation to be inserted

Material Transfer Terminology

- Target (= Setpoint + Spill)
- Set Point (= Target – Spill)
- Spill (=Target –Set Point) (aka - preact, in-flight, offset, bias)
 - Fixed Spill
 - Adaptive Spill
 - Predictive Adaptive Spill
- Fast Feed (aka – coarse feed)
- Feed (aka – fine feed, slow feed, dribble feed)
- Control Methods
 - BASIC Control – single speed transfer control
 -
-
- Dribble Setpoint or Dribble Target or Slow Feed Target(aka slow-feed setpoint, dribble setpoint,
- Flow Rate (aka – feed rate)
- Tolerance (aka –
- Jog (aka - redispensing)
 - Manual Jog – operator starts jog feed and operator ends jog feed
 - Semi Automatic Jog – operator acks out of tol (-ve), starts jog feed and controller ends jog feed
 - Automatic Jog – operator acks selects auto accept out tol (-ve), controller starts jog feed and ends jog feed

Setpoint Error, Target Error, Feed Error ?(aka

10.Spare

Feltkode ændret

Annex ZZ - Bipolar Device Control Shell Wrapper

1. Resource Manager (RM) Supervisory Interface

MEC.RRI.Command.RequestID – (INT) – This input tells the Resource Manager who is requesting a change in the resource that will be controlling the Automation Module. This value is verified against the MEC.RM.Admin.ValidRequestID[] to validate the sending Module ID is allowed to send a value. This input is the Module ID of the specific equipment entity that is requesting to change the value of the resource that is controlling the Automation Module. This input is implemented using an integer value and has a range from 1 to 32767.

MEC.RRI.Command.ResourceID – (DINT) – This input tells the Resource Manager what resource will be controlling the Automation Module. This input is only set by the equipment entity specified by the Module ID in MEC.RRI.Command.RequestID, when the supervisor verifies the Automation Module is listening to it. The upper INT is the Module ID of the Automation Module sending the value and must be the same as the value of the MEC.RRI.Command.RequestID. The lower INT is the Module ID of the specific equipment entity that will be controlling the Automation Module. The value for this Resource Module ID is 1 to 32767. However there is a special value, -1, that will reset the Resource Manager back to a default state. See the **Fejl! Henvisningskilde ikke fundet.**Table 1 below. This input is implemented using a double integer value.

Table 34- MEC.RRI.Command.ResourceID (Lower INT)

-1	Resource Manager Reset Request
0	No Module in Control
1	Module ID 1
•	
•	
•	
32767	Module ID 32767

- Formateret: Venstre
- Formateret: Punktopstilling
- Formateret: Punktopstilling
- Formateret: Punktopstilling

Resource Request Inputs (RRI) Status

MEC.RRI.Status.RequestID – (INT) – This status value indicates the Module ID of the supervising equipment entity that has been acknowledged by the Resource Manager to change the controlling resource of the Automation Module. Once this value is set and read by the supervisor, the supervisor can then send the value for the resource it wants to control this Automation Module. This input is implemented using an integer value.

MEC.RRI.Status.ResourceID – (INT) – This status value indicates the Module ID of the equipment entity that has been allowed to control the Automation Module. This input is implemented using an integer value.

MEC.RRI.Status.RequestStatus – (INT) – This status value indicates the status of the request to change the resource controlling the Automation Module. This input has many different values that indicate the specific states of the request. At a minimum there will be five (5) values as shown in Table 2:

Feltkode ændret

Table 42 - MEC.RRI.Status.RequestStatus

<u>-2</u>	<u>Failed to Release</u>
<u>-1</u>	<u>Failed to Control</u>
<u>0</u>	<u>Initialized</u>
<u>1</u>	<u>Success</u>
<u>2</u>	<u>Pending Request</u>

MEC.RRI.Status.LastResourceID – (DINT) – This status value is the last ResourceID command that was sent. This input is a copy of the incoming command and is implemented using a double integer value. This value is updated everytime a new ResourceID is received.

MEC.RRI.Status.LastRequestStatus – (INT) – This status value is the returned status for the last ResourceID command that was sent. This input is a copy of the status, MEC.RRI.Status.RequestStatus, when the last command was executed. This value is updated everytime a new ResourceID is received.

2 Resource Manager (RM) Subordinate Interface

Resource Request Outputs (RRO) Status

MEC.RRO.Status.RequestID – (INT) – This input tells the Resource Manager who is allowed to request a change in the resource controlling the subordinate Automation Module. If this input matches the Automation Module's ID, then the Resource Manager can send the MEC.RRO.Command.ResourceID to the subordinate Automation Module. This input is implemented using an integer value.

MEC.RRO.Status.ResourceID – (INT) – This input tells the Resource Manager what resource has control of the subordinate Automation Module. This is implemented using an integer value.

MEC.RRO.Status.RequestStatus – (INT) – This input tells the Resource Manager the value of the subordinate Automation Module's request status. This is implemented using an integer value and will support at least the following values.

<u>-2</u>	<u>Failed to release Subordinate Automation Module</u>
<u>-1</u>	<u>Failed to control Subordinate Automation Module</u>
<u>0</u>	<u>Initialized</u>
<u>1</u>	<u>Success</u>
<u>2</u>	<u>Pending Request</u>
<u>3</u>	<u>Acquiring Subordinate Automation Modules</u>
<u>4</u>	<u>Releasing Subordinate Automation Modules</u>

Resource Request Outputs (RRO) Commands

MEC.RRO.Command.RequestID – (INT) – This output is sent to the Subordinate Resource Manager to request access to change the resource that is supervising the Subordinate Automation Module. This output is the Module ID of this Automation Module. This output is implemented using an integer value and has a range from 1 to 32767.

MEC.RRO.Command.ResourceID – (DINT) – This output is sent to the Subordinate Resource Manager to change what resource will be controlling the Subordinate Automation Module. This output is only set if the MEC.RRO.Status.RequestID is the Module ID of this Automation Module. The upper INT is the Module ID of the Automation Module sending the value and would be the Module ID of this Module. The lower INT is the Module ID of the specific equipment entity that will be controlling the Automation Module. Typically this will be the Module ID of this Automation Module. This lower INT can also be a -1 which will send a reset command to the subordinate modules. See Table below. This output is implemented using a double integer value.

<u>-1</u>	<u>Resource Manager Reset</u>
<u>0</u>	<u>No Module in Control</u>
<u>1</u>	<u>Module ID 1</u>
<u>•</u>	
<u>•</u>	
<u>•</u>	
<u>32767</u>	<u>Module ID 32767</u>

← **Formateret:** Punktopstilling

← **Formateret:** Punktopstilling

← **Formateret:** Punktopstilling

3 Resource Manager (RM) Administrative Interface

Resource Manager (RM) Administration

MEC.RM.Admin.ValidRequestID[] – (INT Array) – This value is usually configured at design time, but can be exposed to allow runtime changes. It lists the valid Module IDs that can supervise this Automation Module. The valid values follow the supported types of the Command Processor “WHO” array, see section XXXXX Editors Note: Enter correct Section. Using the instance format, one set of values are as follows:

<u>0</u>	<u>No Module in Control</u>
<u>1</u>	<u>Module ID 1</u>
<u>•</u>	
<u>•</u>	
<u>•</u>	
<u>32767</u>	<u>Module ID 32767</u>

← **Formateret:** Automatisk nummerering
+ Niveau: 1 + Nummereringstypografi:
1, 2, 3, ... + Begynd med: 2 +
Justeret: Venstre + Justeret: 0,63 cm
+ Tabulator efter: 1,27 cm +
Indrykning: 1,27 cm

← **Formateret:** Punktopstilling

← **Formateret:** Punktopstilling

← **Formateret:** Punktopstilling

Feltkode ændret

MEC.RM.Admin.SubordinatePassThru – (Boolean) – This value is usually configured at design time, but can be exposed to allow runtime changes. It determines whether the Resource Manager will automatically pass thru the resource requests or whether the subordinate interface will be controlled by the Function Manager.

MEC.RM.Admin.TimeoutPreset – (INT) – This value is typically configured at design time, but can be exposed to allow runtime changes. It indicates the preset for the timeout timer. This is the time that the Resource Manager will wait for the acquisition or release of subordinate Automation Modules before returning an error to MEC.RRI.Status.RequestStatus. This value is the number of milliseconds to wait.

MEC.RM.Admin.ReleaseDelay – (INT) – This value is typically configured at design time, but can be exposed to allow runtime changes. It indicates the preset for the release timer. This is the time the Resource Manager will wait before commanding a release of the subordinate Automation Module. This will keep the subordinate Automation Modules from being released and re-acquired very quickly. This value is the number of milliseconds to wait.

MEC.RM.Admin.ModuleID – (INT) – This value is typically configured at design time, but can be exposed to allow runtime changes. This value is the unique identifier for this Automation Module. This value is implemented as an integer.

MEC.RM.Admin.AsyncPreset – (INT) – This value is typically configured at design time, but can be exposed to allow runtime changes. This value is the Asynchronous communications timer preset that is used to delay further processing of the Resource Manager. This allows for a slow communications link between the parent Automation Module and the Resource Manager. This value is the number of milliseconds to wait.

MEC.RM.Admin.AllowChangeNoRelease – (BOOL) – This value is typically configured at design time, but can be exposed to allow runtime changes. This value is used to allow Resource Manager to bypass the requirement for ResourceID to be zero (0 = No Resource in control), when determining when to process a request to change RequestID or owner.

34 Function Manager Incoming Commands

Control Request Inputs (CRI) Commands

Control Request Inputs (CRI) Commands

MEC.CRI.Command – (DINT) – This input allows the resource that is in command to request that the Automation Module execute a command. The upper integer will hold the ModuleID of the sending Automation Module. This value is validated against the MEC.FM.Admin.ValidModuleID[] array. The lower integer will hold the Command value which will be one of the following:

0	Undefined
1	Auto Mode of Action Request
2	Semi-Auto Mode of Action Request
3	Manual Mode of Action Request
4	Alarm Acknowledge
5	Reset Function Manager and Functional

	Strategy
6	Activate Request
7	Deactivate Request

The Mode of Action determines the action of the Functional Strategy. For example when a PID loop is put in to manual the input setpoint is written directly to the output of the PID function bypassing the PID algorithm altogether.

Control Request Inputs (CRI) Status

MEC.CRI.Status – (INT) – This is an output back to the commanding Automation Module to indicate the status of the command. The value will be one of the following:

-2	Command Failed – Not valid command
-1	Command Failed – Not Authorized
0	Not Defined
1	Command Succeeded and Processed
2	Command Succeeded – Not in Control of Module

MEC.CRI.LastCommand – (DINT) – This value is a copy of the last command executed regardless of command status. This value is updated every time a new command is received.

MEC.CRI.LastCommandStatus – (INT) – This value is a copy of the status after the last command was executed. This value is updated every time a new command is received.

Control Variable Inputs (CVI) Inputs

This is typically where physical inputs, interlocks, permissives, parameters or feedback signals are input to the Automation Module.

MEC.CVI.Permissive – (BOOL) – This input determines if the Automation Module has the appropriate permissions to activate or deactivate. This input does not generate a fault when not enabled. This input controls the operation of the device. If there is a command and the permissive is off, the Functional Strategy will be ready to execute the command as soon as the permissive becomes true. If device is running and the permissive goes False, the device will stop running until the permissive becomes true again, at which time the device will start running. The Permissive logic for multiple permissive signals is done outside this Automation Module because this logic is custom for every implementation.

MEC.CVI.Interlock – (BOOL) – This input determines if the Automation Module is blocked from activating or deactivating by any other signals. The interlock will generate a fault if the interlock is false. To clear the fault and restart the device will require some external action. The Interlock logic for multiple interlock signals is done outside this Automation Module because this logic is different for every implementation.

Feltkode ændret

MEC.CVI.Feedback.Activated – (BOOL) – This input is the status of the module output command from the Functional Strategy. This input lets the module know if the output actually came on.

MEC.CVI.Feedback.Deactivated – (BOOL) – This input is the status of the module output command from the Functional Strategy. This input lets the module know if the output actually went off.

45 Functional Manager (FM) Outgoing Command Outputs

Control Request Outputs (CRO) Commands

MEC.CRO.Command – (DINT) – This output allows the Automation Module to request a Subordinate Automation Module to change the Mode of Action. The upper integer will hold the ModuleID of this Automation Module. The lower integer value for ModeRequest will be one of the following:

0	Undefined
1	Auto Mode of Action Request
2	Semi-Auto Mode of Action Request
3	Manual Mode of Action Request
4	Alarm Acknowledge
5	Reset Request

Then Mode of Action determines the action of the Functional Strategy. For example when a PID loop is put in to manual the input setpoint is written directly to the output of the PID function bypassing the PID algorithm altogether.

Control Request Outputs (CRO) Status

MEC.CRO.Status – (INT) – This is an output back to the commanding Automation Module to indicate the status of the last command. The value will be one of the following:

-2	Command Failed – Not valid command
-1	Command Failed – Not Authorized
0	Not Defined
1	Command Succeeded and Processed
2	Command Succeeded – Not in Control of Module

Control Variable Outputs (CVO) Outputs

Currently there are no control outputs defined as being common to all Automation Modules. This is typically where physical outputs, or outputs to other control modules are specified. Since every Automation module is different, this specification is not defining inputs of this type.

If there were control outputs defined the attributes would have the following naming:

MEC.CVO.XXXXXXX

where the XXXXXX would be replaced with the attribute name.

Functional Manager (FM) Administrative Inputs

This is typically configuration type data. The developer of the Automation Module may not even expose any attributes of this type.

Function Manger (FM) Administration

MEC.FM.Admin.ValidModuleID[] – (INT Array) – This value is typically configured at design time, but can be exposed to allow for runtime changes. It lists the valid Module IDs that can send commands to this Automation Module. The valid values follow the supported types of the Command Processor “WHO” array, see section XXXXX Editors Note: Enter correct Section. Using the instance format, one set of values are as follows:

<u>0</u>	<u>No Module in Control</u>
<u>1</u>	<u>Module ID 1</u>
<u>.</u>	
<u>.</u>	
<u>.</u>	
<u>32767</u>	<u>Module ID 32767</u>

Formateret: Punktopstilling

Formateret: Punktopstilling

Formateret: Punktopstilling

MEC.FM.Admin.Alarm.ActionOnAcknowledge – (BOOL) – This input allows the resource that is in command to request that the Automation Module activate or deactivate after an alarm acknowledgement. If the value is 1 then the Automation Module will activate after an alarm acknowledgement and if the value is a 0 it will deactivate after an alarm acknowledgement.

MEC.FM.Admin.Alarm.ActionOnAlarm – (BOOL) – This input allows the resource that is in command to request that the Automation Module activate or deactivate after an alarm is present. If the value is 1 then the Automation Module will activate after an alarm is present and if the value is a 0 it will deactivate after an alarm is present.

MEC.FM.Admin.Alarm.ActionOnClear – (BOOL) – This input allows the resource that is in command to request that the Automation Module activate or deactivate after an alarm is cleared. If the value is 1 then the Automation Module will activate after an alarm is cleared and if the value is a 0 it will deactivate after an alarm is cleared.

MEC.FM.Admin.ModeTransition.ActivateRequest – (BOOL) – This input allows the resource that is in command to request that the Automation Module activate after the procedure mode has changed.

MEC.FM.Admin.ModeTransition.DeactivateRequest – (BOOL) – This input allows the resource that is in command to request that the Automation Module deactivate after the procedure mode has changed.

MEC.FM.Admin.ModeTransition.MaintainLastRequest – (BOOL) – This input allows the resource that is in command to request that the Automation Module to maintain the last command, activate or deactivate, after the procedure mode has changed.

Feltkode ændret

MEC.FM.Admin.StateChangeAlarmTime – (INT) – This input allows the resource that is in command to configure the Automation Module’s alarm timer. This value indicates the maximum time that it should take to change from activate to deactivate. If this time is exceeded an alarm will be generated. This time is specified in milliseconds to wait.

56 MEC Status Overview

Every Automation Module will have status data that will indicate the state of the Automation Module. This data is an overall status and not limited to any one internal algorithm.

MEC Status

MEC.Status.AlarmPresent – (BOOL) – This status signal indicates that the Automation Module is in an alarm condition.

MEC.Status.AlarmAcknowledged – (BOOL) – This status signal indicates that the Automation Module has acknowledged all alarm.

MEC.Status.Reset – (BOOL) – This status signal indicates that the Automation Module has been reset to a default state.

MEC.Status.ModeCurrent – (INT) – This status value indicates the current Mode of Action for the Automation Module. The value will be one of the following values:

0	Undefined
1	Auto Mode of Action
2	Semi-Auto Mode of Action
3	Manual Mode of Action

MEC.Status.ModeRequested – (INT) – This status value indicates the Mode of Action that has been requested. The value will be one of the following values:

0	Undefined
1	Auto Mode of Action
2	Semi-Auto Mode of Action
3	Manual Mode of Action

MEC.Status.RequestStatus – (INT) – This status value indicates the status of the request to change the resource in command. This input has many different values that indicate the specific equipment entity that can be in command. At a minimum there will be three (3) values as follows:

-1	Failed
0	Initialized

1	Success
---	---------

The Initialized value is the default before any request is processed. With these three values you will always know that there was a change when the request is processed.

67 MEC Reset Command

Every Automation Module will have a global command to reset the entire Automation Module.

MEC.Reset.ResetRequest – (DINT) – This input allows the resource that is in command to request that the Automation Module be reset to a default state. The upper integer will hold the ModuleID of the sending Automation Module. The lower integer holds the commanded state for the Reset, True or False. This is used throughout the Automation Module to reset all pieces, Resource Manager, Functional Manager and Functional Strategy, to a known default state.

78 MEC Control Shell Variable List

DeviceName						DeviceName	Data Type
	MEC					DeviceName.MEC	S8805
		RRI				DeviceName.MEC.RRI	Supervisory
			Command			DeviceName.MEC.RRI.Command	Command
				RequestID		DeviceName.MEC.RRI.Command.RequestID	Int (16 bit)
				ResourceID		DeviceName.MEC.RRI.Command.ResourceID	Int (32 bit)
				Status		DeviceName.MEC.RRI.Status	Status
				RequestID		DeviceName.MEC.RRI.Status.RequestID	Int(16 bit)
				ResourceID		DeviceName.MEC.RRI.Status.ResourceID	Int(16 bit)
				RequestStatus		DeviceName.MEC.RRI.Status.RequestStatus	Int(16 bit)
				LastResourceID		DeviceName.MEC.RRI.Status.LastResourceID	Int(32 bit)
				LastRequestStatus		DeviceName.MEC.RRI.Status.LastRequestStatus	Int(16 bit)
		RRO				DeviceName.MEC.RRO	Subordinate
			Command			DeviceName.MEC.RRO.Command	Command
				RequestID		DeviceName.MEC.RRO.Command.RequestID	Int (16 bit)
				ResourceID		DeviceName.MEC.RRO.Command.ResourceID	Int (32 bit)
				Status		DeviceName.MEC.RRO.Status	Status
				RequestID		DeviceName.MEC.RRO.Status.RequestID	Int(16 bit)
				ResourceID		DeviceName.MEC.RRO.Status.ResourceID	Int(16 bit)
				RequestStatus		DeviceName.MEC.RRO.Status.RequestStatus	Int(16 bit)
		RM				DeviceName.MEC.RM	RM
			Admin			DeviceName.MEC.RM.Admin	Admin
				ValidRequestID[#]		DeviceName.MEC.RM.Admin.ValidRequestID[#]	Array of Int (16 bit)
				SubordinatePass Thru		DeviceName.MEC.RM.Admin.SubordinatePassThru	Bool
				TimeOutPreset		DeviceName.MEC.RM.Admin.TimeOutPreset	Int(16 bit)
				ReleaseDelay		DeviceName.MEC.RM.Admin.ReleaseDelay	Int(16 bit)
				ModuleID		DeviceName.MEC.RM.Admin.ModuleID	Int(16 bit)
				AsyncPreset		DeviceName.MEC.RM.Admin.AsyncPreset	Int(16 bit)
				AllowChangeNoRelease		DeviceName.MEC.RM.Admin.AllowChangeNoRelease	Bool
		CRI				DeviceName.MEC.CRI	CommandRequestInput
			Command			DeviceName.MEC.CRI.Command	Int (32 bit)
			Status			DeviceName.MEC.CRI.Status	Int (16 bit)
			LastCommand			DeviceName.MEC.CRI.LastCommand	Int (32 bit)
			LastCommandStatus			DeviceName.MEC.CRI.LastCommandStatus	Int (16 bit)
		CVI				DeviceName.MEC.CVI	ControlVariable
			Permissive			DeviceName.MEC.CVI.Permissive	Bool
			Interlock			DeviceName.MEC.CVI.Interlock	Bool
			Feedback			DeviceName.MEC.CVI.Feedback	Bool
				Activated		DeviceName.MEC.CVI.Feedback.Activated	Bool
				Deactivated		DeviceName.MEC.CVI.Feedback.Deactivated	Bool
		CRO				DeviceName.MEC.CRO	CommandRequestOutput
			Command			DeviceName.MEC.CRO.Command	Int (32 bit)
			Status			DeviceName.MEC.CRO.Status	Int (16 bit)
		CVO				DeviceName.MEC.CVO	ControlVariable
			XXXXXXX			DeviceName.MEC.CVO.XXXXXXX	Any
		FM				DeviceName.MEC.FM	FM
			Admin			DeviceName.MEC.FM.Admin	Admin
				ValidModuleID[#]		DeviceName.MEC.FM.ValidModuleID[#]	Array of Int (16 bit)
				Alarm		DeviceName.MEC.FM.Admin.Alarm	Alarm
					ActionOnAcknowledge	DeviceName.MEC.FM.Admin.Alarm.ActionOnAcknowledge	Bool
					ActionOnAlarm	DeviceName.MEC.FM.Admin.Alarm.ActionOnAlarm	Bool
					ActionOnClear	DeviceName.MEC.FM.Admin.Alarm.ActionOnClear	Bool
				ModeTransition		DeviceName.MEC.FM.Admin.ModeTransition	ModeTransition
					ActivateRequest	DeviceName.MEC.FM.Admin.ModeTransition.ActivateRequest	Bool
					DeactivateRequest	DeviceName.MEC.FM.Admin.ModeTransition.DeactivateRequest	Bool
					MaintainLastRequest	DeviceName.MEC.FM.Admin.ModeTransition.MaintainLastRequest	Bool
				StateChangeAlarmTime		DeviceName.MEC.FM.Admin.StateChangeAlarmTime	Int (16 bit)
		Status				DeviceName.MEC.Status	Status
			AlarmPresent			DeviceName.MEC.Status.AlarmPresent	Bool
			AlarmAcknowledged			DeviceName.MEC.Status.AlarmAcknowledged	Bool
			Reset			DeviceName.MEC.Status.Reset	Bool
			ModeCurrent			DeviceName.MEC.Status.ModeCurrent	Int(16 bit)
			ModeRequested			DeviceName.MEC.Status.ModeRequested	Int(16 bit)
			RequestStatus			DeviceName.MEC.Status.RequestStatus	Int(16 bit)
			Activated			DeviceName.MEC.Status.Activated	Bool
			Deactivated			DeviceName.MEC.Status.Deactivated	Bool
			Indeterminant			DeviceName.MEC.Status.Indeterminant	Bool
		Reset				DeviceName.MEC.Reset	Reset
			ResetRequest			DeviceName.MEC.Reset.ResetRequest	Int(32 bit)

Feltkode ændret

Material Deliver Process Example for Discussion and Concept Development

To better enable the Make2Pack Part 5 group share ideas and concepts a detailed process example based upon the Best Darn Glue Company using a single batch material delivery system that has the ability to vary the temperature of the material has been created. The diagrams and operational descriptions used here for discussion will change as concepts and the Make2Pack ISA S88 Part 5 effort evolve so always look to the Make2Pack FTP site for the current material. This description refers to many manufacturing organizations that are required to support it and details of those organizations are not found in this description.

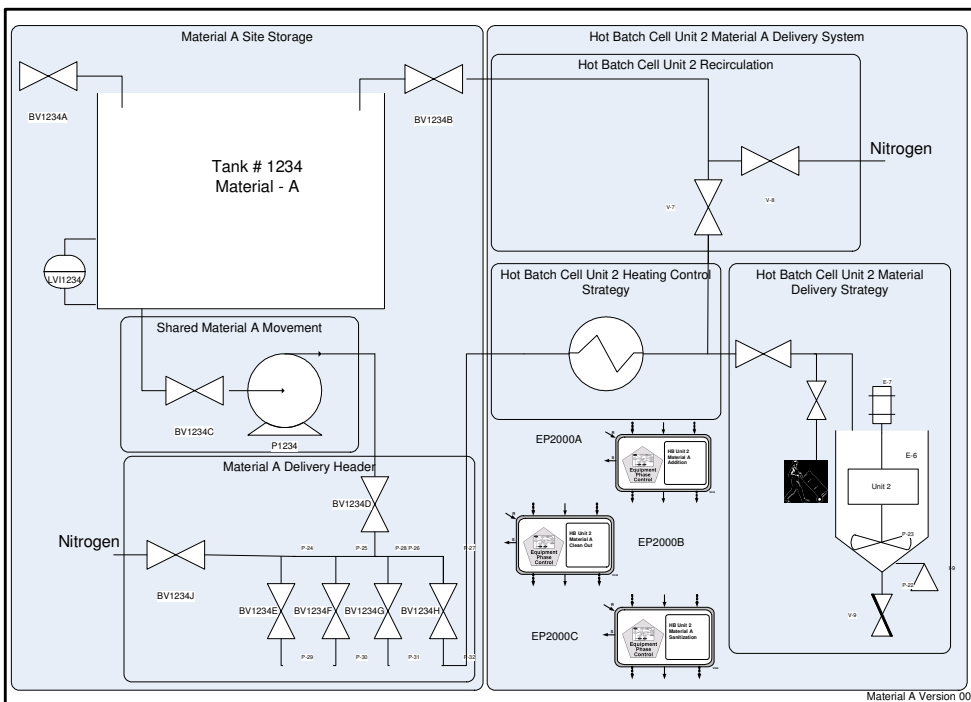


Figure 21 Material A Storage & HBC U2

Material A Overview for Best Glue Company's North Site:

The diagram in Figure 1 represents the Best Glue Company's North Site Material A Storage and Transfer system and the Hot Batch Cell part.

Feltkode ændret

Material A Processing overview:

The Site Storage Area (Equipment Tags Range 1000-1999) is responsible for managing the amount of Material A on hand and the deliver of Material A to the four processing cells found in the Production Area. These Cells are:

- Hot Batch (Equipment Tags Range 2000-2999)
- Cold Batch (Equipment Tags Range 3000-3999)
- Continuous (Equipment Tags Range 4000-4999)
- Test Market (Equipment Tags Range 5000-5999)

Material A is stored in Tank # 1234, material quantity is measured and managed using Tank # 1234's level sensor LV11234. There are two piping connections into Tank #1234. The primary valve BV1234A is used by the material transport unloading system to supply Material A to Tank # 1234 per the process engineer's directions. Material can be delivered to Tank # 1234 at any time as long as the Tank # 1234 is in service.

Material A can be supplied to no more than two processing cells simultaneously and will be managed by the Material Storage area using a first come first serve method of arbitration.

When any Material A delivery subsystem will not be used within an 8 hour time period it is necessary to clean out that piping structure using a nitrogen blow down operation as specified by the process engineer. Transport of reclaimed Material A will be a manual operation with Reclaimed Material A being returned to the transport unloading area for disposition per the guidance of the analysis lab. Rejected Reclaimed Material A will be neutralized by the Site Material Storage per the lab guidelines and disposed of per the local requirements. Every Material A Equipment Subsystem will have procedures for clean out and sanitization as specified by the process engineer. These will be executed by the operations management of each production cell at least once a week and if indicated by the process engineer more often.

The recirculation Valve BV1234B is used by the Hot Batch Cell Unit 2 recirculation system and is only used during production of Good and Super Glue in the Hot Batch Cell.

The Shared Material A Movement subsystem consists valve BV1234C and pump P1234, if no production cell has requested use of Material A for 15 minutes the Shared Material A Movement subsystem will be shutdown until another request occurs.

Feltkode ændret

Material A Hot Batch Cell (HBC) Unit 2 (U2) Process Strategy

For both Good & Super Glue Material A must be combined with other material in the Hot Batch (HB) Cell through the HB Unit 2. The temperature range for deliver to the Hot Batch Cell Unit 2 is ambient to 90 degrees C +/- 2.5 degree C and the desired temperature is specified by the recipe during manufacturing. When OK glue is made on this system the temperature of Material A is ambient and the heating system is not used.

The HBC-U2-Material A Equipment Module has three different control strategies that are supported by this EM. Each strategy is contained in an S88 Equipment Phase Control structure and can be managed by the Recipe or Equipment Procedural control as dedicated by the Process Engineer or the Product Specialist. Only one strategy can be active at a time and each strategy will direct the subordinate EMs that make up the HBC-U2-Material A Equipment Module as appropriate. The three control strategies are:

Addition: Description of EP2000A-----

Clean-Out: Description of EP2000B -----

Sanitization: Description of EP2000C -----

Interlocks and Permissives used by all the Control Components of this EM are evaluated and managed by IP2200A

The HBC-U2-Material A Equipment Module consists of three sub-Equipment Modules that manage the equipment as follows:

Hot Batch Cell Unit 2 Recirculation: Description -----

Hot Batch Cell Unit 2 Heating: The heating equipment for HBC U2 consists of a plate and frame heat exchanger, HE2200, with a steam control valve FV2200 used to control the output temperature using Cascaded PID control. When the heating system is not active the steam block valve, BV2200, will be closed. When heating is required Material A will be requested for delivery to HBC U2 with the recirculation system activated when

Feltkode ændret

Material A is available.

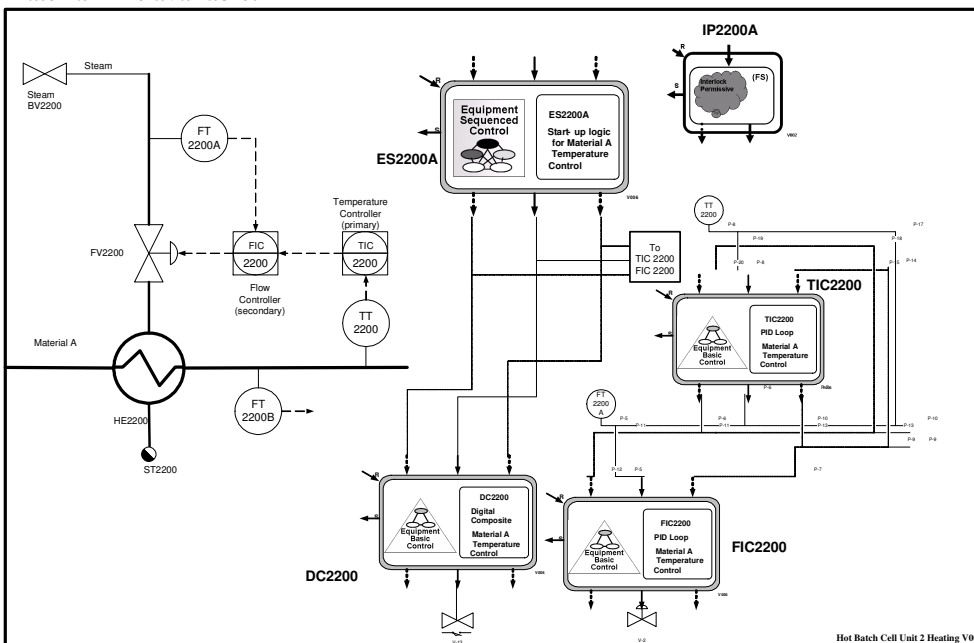


Figure 22 HBC U2 Heating Control Strategy

When the Material A is in recirculation as indicated by the flow indicator FT2200B Steam Valve BV2200 will be opened and FV2200 will be fully opened. When Material A has reached the minimum required temperature as specified by the procedure the Cascade Control system will be activated using the procedure target temperature as the control target. At this time the HBC U2 Material A delivery system will be activated and the Recirculation system deactivated. When the current requested activity is complete the Heating system will be deactivated. The activity of this EM is managed by the S88 Equipment Sequence Control ES2200A and directs the S88 Basic Control Components DC2200, FIC2200 and TIC2200 as specified by the Process Engineer.

Hot Batch Cell Unit 2 Material Addition: Description -----

Feltkode ændret