Machine Template for Production Machines

V1.1.1 - Using OMAC and ISA-88 compliant

https://support.industry.siemens.com/cs/ww/en/view/109804620

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.
Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.
By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.
The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with Industrial Security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.
Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.
For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.
Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: https://www.siemens.com/industrialsecurity.

Table of contents

Legal information **2**

1 Introduction **5**

1.1 Overview 5

1.2 Modular project architecture compliant to ISA-88 6

1.3 OMAC Mode and State Manager 7

1.4 Components used 9

2 Engineering **10**

2.1 Organization Blocks 10

2.2 Machine Template 11

2.3 Unit 11

2.4 Equipment Modules 12
2.4.1 Control Modules 13

2.5 Alarm handling 13
2.5.1 Error detection 13
2.5.2 Unit Status 15

3 Working with the Machine Template **16**

3.1 Subdivision of the machine 16

3.2 Project configuration 16
3.2.1 Adaption of User Defined Types 16
3.2.2 PLC tags of MTplPlc 20
3.2.3 Unit DBs 21
3.2.4 Initialization of machine using StartupUnit FB 24

3.3 Module adaption 26
3.3.1 Adaption of the Technical Module 26
3.3.2 Adaption of the Equipment Module 29
3.3.3 Adaption of the Unit 32

3.4 Module extension 32
3.4.1 Implementation of a Technical Module 32
3.4.2 Implementation of a Equipment Module 33

3.5 Project testing 34

4 Program blocks **36**

4.1.1 MTplPlc_SequenceTemplate (FB) 36
4.1.2 MTplPlc_TechModuleTemplate (FB) 40
4.1.3 MTplPlc_UnitCommandCollector (FB) 43
4.1.4 MTplPlc_UnitConditionCollectorTemplate (FB) 47

5 PLC data types **49**

5.1 Commands 49
5.1.1 MTplPlc_typeManagerBooleanInterface (UDT) 49
5.1.2 MTplPlc_typeManagerCommands (UDT) 49
5.1.3 MTplPlc_typeManagerIntegerInterface (UDT) 49
5.1.4 MTplPlc_typeManagerModeBoolean (UDT) 50
5.1.5 MTplPlc_typeManagerRemoteInterface (UDT) 50
5.1.6 MTplPlc_typeManagerStateBoolean (UDT) 51

5.2 Configuration 52
5.2.1 MTplPlc_typeManagerConfiguration (UDT) 52

5.2.2 MTplPlc_typeManagerDisableModeChange (UDT) ... 52
5.3 Status ... 53
5.3.1 MTplPlc_typeCmDiagnostics (UDT) ... 53
5.3.2 MTplPlc_typeEmDiagnostics (UDT) ... 53
5.3.3 MTplPlc_typeEmStateMonitoring (UDT) ... 53
5.3.4 MTplPlc_typeManagerMonitoring (UDT) ... 54
5.3.5 MTplPlc_typeManagerStatus (UDT) ... 55
5.3.6 MTplPlc_typeTmDiagnostics (UDT) ... 55
6 PLC tags & constants ... 56
6.1.1 MTplPlc_Constants ... 56
7 Appendix ... 57
7.1 Service and support ... 57
7.2 Links and Literature ... 58
7.3 FAQ / Frequently Asked Questions ... 58
7.4 Change log ... 59

1 Introduction

1.1 Overview

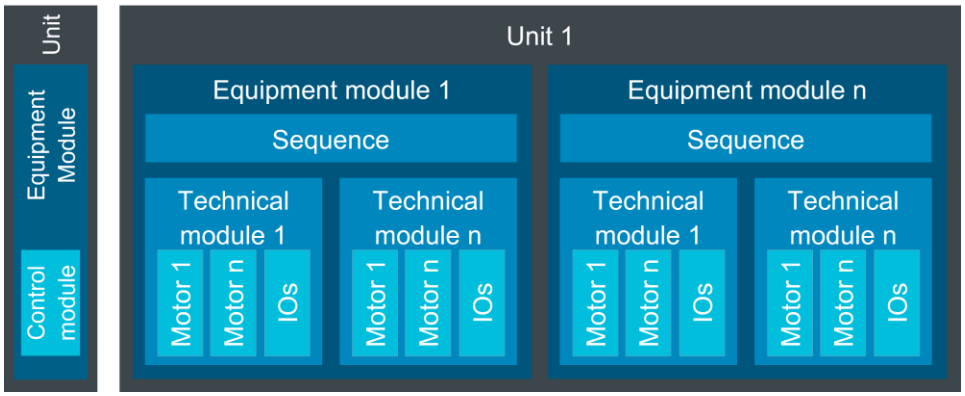
The Machine Template is a modular TIA Portal project for S7-1500 T controllers. It is aimed for machine builders for implementing the automation program for different machines. It provides the basic functionalities for controlling and diagnostics of machine modules, using a standardized operating mode manager and a standardized interface.

Therefore, the Machine Template covers the three relevant levels of the physical model according to ISA-88. The machine is represented by the Unit layer which collects all commands from remote interfaces and the subordinate Equipment Modules and forwards them to the OMAC manager.

The Equipment Module layer is representing one processing step of the machines production chain. It contains a summary of several Control Modules, which can be bundled in so called Technical Modules. This level is strongly dependent on the respective technology of the machine. In order to build up the Equipment Module as modular as possible, it is divided into a so-called Sequence and several Technical Modules, which are both contained in the project by a template.

The sequence provides a state machine for all operating modes that correspond to the user program. The Technical Module contains one or more Control Modules which can be technically combined. The Technical Module receives the commands from the sequence and passes them on to the subordinate Control Modules. Technology-specific functions such as cam calculation are also carried out here.

Figure 1-1: Architecture of the Machine Template



The advantage of this Machine Template is that the user program is separated from the technology. This means that the user program is programmed in the sequence of the Equipment Module and the technology is programmed in the related Technical Modules. It is easy to integrate technology specific libraries. Figure 1-1 shows the architecture of the described Machine Template and its physical model.

Advantages

- Uses OMAC and is ISA-88 compliant
- Provides framework for all PLC projects regardless of branch
- Already existing PLC program can be integrated
- Can be modularly adapted and extended
- Standardized interface (internal and to external interfaces)

1.2 Modular project architecture compliant to ISA-88

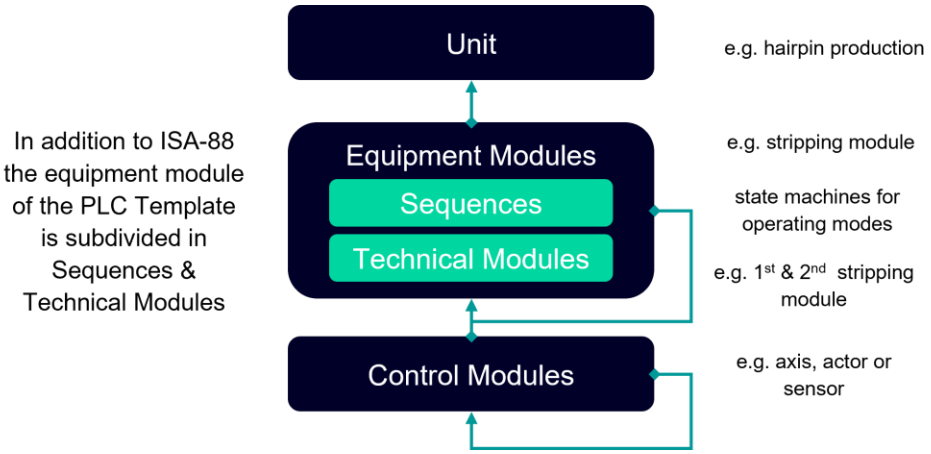
The ISA-88 is a standard which describes a method to structure batch processes. It is a design philosophy for software, Equipment and the process flow. The first part was already released by ISA in 1995. ISA-88 has been adopted by the IEC as IEC 61512.

The ISA-88 defines three main models:

1. The process model, which handles the aspects of the chemical and physical changes during production.
2. The physical model, which handles the equipment being used during production.
3. The procedural control model, which handles the recipe structure for the product.

For the modular automation of production machines, the physical model of ISA-88 is an essential element for the programming structure. It is hierarchically composed of the layers that can be seen in figure 1-2. The Machine Template comprises the physical model from Unit level to the Control Module level because they are the integral parts of a production machine controlled by a PLC project.

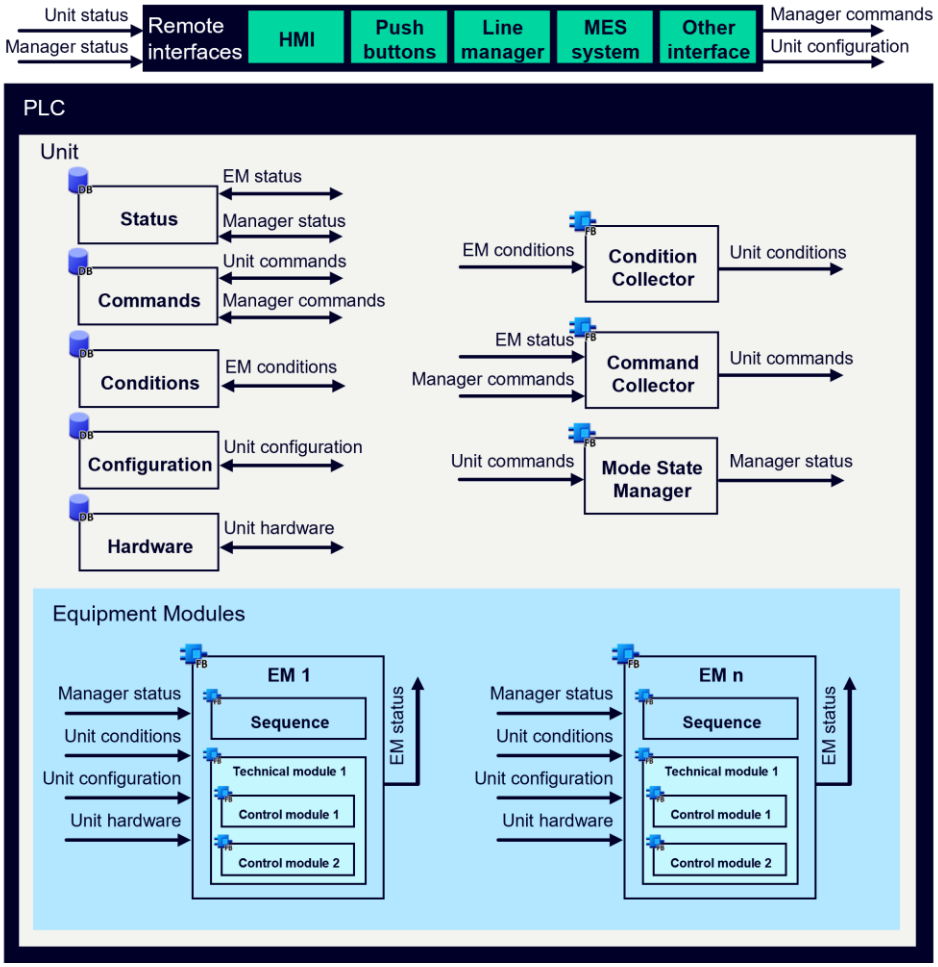
Figure 1-2: Physical model of Machine Template on the basis of ISA-88



A Unit typically is a group of Equipment Modules, which in turn generally is implemented as an aggregation of Control Modules to fulfill certain tasks or process actions. An example for a Unit would be a hairpin production line which is part of a stator production. An Equipment Module is typically centered around a piece of process such as separating of pins or bending hairpins. A Control Modules represent sensors, actuators or the actual processing axis that perform specific tasks.

Figure 1-3 shows the implementation of the Machine Template based on the physical model according to ISA-88. The figure shows the three levels Unit, Equipment Module and Control Module and a sample configuration of a Unit, consisting of two Equipment Modules.

Figure 1-3: Machine Template scheme



The first Equipment Module represent the standard implementation of an Equipment Module which is provided by this template. More information about Equipment Modules can be found in chapter 2.4. In addition, the data exchange between the layers and external interfaces can also be seen in the figure 1-3.

Note For more information about the state machine model which is used in this template, please refer to:
https://support.industry.siemens.com/cs/de/de/view/109784331

1.3 OMAC Mode and State Manager

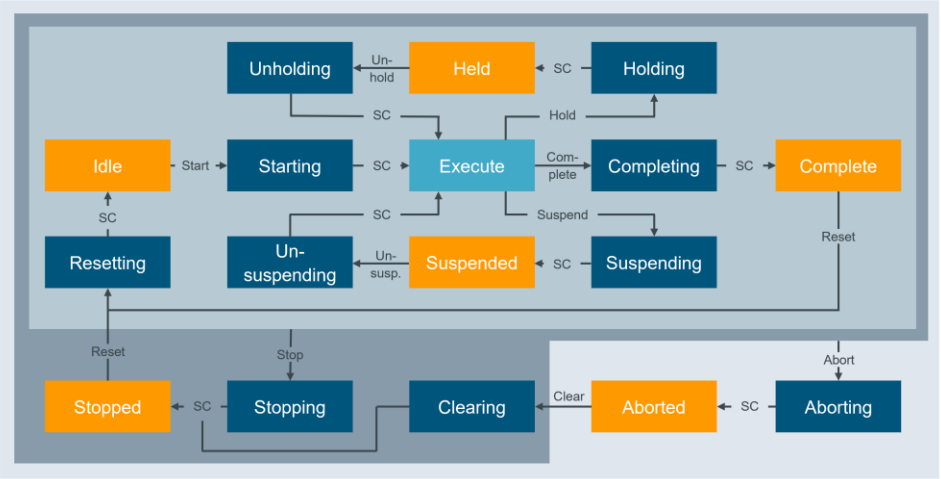
OMAC (The Organization for Machine Automation and Control) is the organization for automation and manufacturing professionals that is dedicated to support the machine automation and operation needs of manufacturers. The “ISA Technical Report TR88.00.02 Machine and Unit States” generally describes the OMAC mode management (Unit Mode State Manager and state machine).

Note For more information about the state machine model which is used in this template, please refer to:
https://support.industry.siemens.com/cs/de/de/view/49970441

In the Machine Template the OMAC mode management is provided by the library OMAC PackML V3.0 (LPMLV30). It provides the modes *Production*, *Maintenance*, *Manual* and up to 8 *User-defined modes*.

Each Unit mode has its own state machine which represents a control model that is defined by fixed operating states. These states have defined requirements in order to move from one state to another. The machines operation can be summarized by the provided states. Which states are used in the individual mode is not standardized and users can define them as required. The state model for the *Production* mode should be considered as the maximum quantity structure.

Figure 1-4: State machine model



This structure may be reduced for the other modes, but should not be increased. This means that the state machine of the *Production* mode is always used and for smaller quantity structures, individual states are directly run-through or skipped.

The state machine model from OMAC contains a set of 17 predefined states, devided into three categories, wait states in orange, acting states in dark blue and the dual state *Execute* in light blue, see figure 1-4. The state machine remains in a wait state, until it receives a command to enter the next state. Wait states are Idle, Stopped, Aborted, Suspended, Held, Completed, and Execute. During an acting state the machine must perform an action. Afterwards when it is finished it automatically changes to the next waiting state. Acting states have the suffix “ing”: Resetting, Starting, Clearing, Stopping, Aborting, Suspending, Unsuspending, Holding, Unholding and Completing. Not all states have to be used, but none can be added.

Note

For more information about the state machine model which is used in this template, please refer to: https://www.omac.org/packml

1.4 Components used

This application example has been created with the following hard- and software components which are shown in the following table 1-1.

Table 1-1 Used components

Component	Number	Article Number	Note
CPU 1517T-2 PN	1	6ES7513-1AL01-0AB0	Or any other S7-1500 T-CPU which is suitable
Step7 Prof V17 Up1	2	6ES7822-1AA07-0YA5	
WinCC Unified V17 Up1	3	6AV2151-0XB01-7AA5	

2 Engineering

The Machine Template provides the necessary structure to implement an ISA-88 compliant modular machine project. In the following sections the hiercharchical structure and all project blocks, types and constants are described in detail.

The Unit of the Machine Template consists of one Equipment Module, which has one Technical Module. This Technical Module consists of one Control Module that is called cyclically and one Control Module that needs a synchronous cycle call. A second folder *Equipment Module Xy* is created to demonstrate how to implement further Equipment Modules.

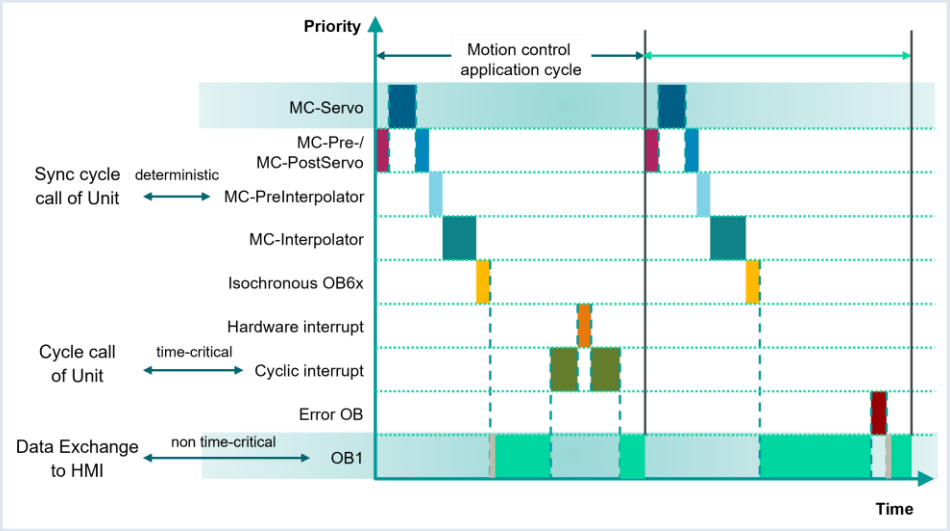
Another part of the Machine Template is the WinCC Unified HMI.

Note For more information about the Machine Template HMI please refer to the according SIOS entry.
https://support.industry.siemens.com/cs/document/109804620

2.1 Organization Blocks

In the Machine Template project four organization blocks (OBs) are used. In figure 2-1 an overview for the execution system is given.

Figure 2-1: Execution system



Main

The *Main* OB is reserved for communication and data exchange with remote interfaces like HMI or MES. Here, the PLC data from the *Unit* DBs is written to the *HMIdata* DB and visa versa.

Cyclic Interrupt

In the *Cyclic interrupt* OB, the actual call of the Unit from the Machine Template takes place. The call is made in this OB in order to achieve a cyclic call and to avoid issues due to communication load. To achieve a higher priority for the PLC project, the OB priority for *cyclic interrupt* is set to 16 due to the communication priority of 15.

All FBs of the Unit that are called in *cyclic interrupt* have the prefix *Cycle*. Within the *Equipment Module* folder the FBs for the Equipment Module, the Equipment Module sequence and the Technical Modules are found in the folder *Cycle*.

PreInterpolator

In the *PreInterpolator* OB the so called *synchronous cycle* modules are called. This means that all Equipment Modules of the Unit that consist of at least one Technical Module that requires a call in a deterministic cycle, are called in a *synchronous cycle* like *PreInterpolator*. In the Machine Template this is illustrated by the *PreInterpolator*, but depending on the use case, the OB could be also the *PostInterpolator*.

The synchronous cycle is, for example, needed when a Technical Module needs to evaluate an actual value in a fast cycle or needs to write output signals like an outputcam in a deterministic cycle. These synchronous cycle have the prefix *SyncCycle*. Technical Modules that need this deterministic call can be found in the folders *SyncCycle* underneath every Equipment Module.

Startup

The *Startup* OB contains the *StartupUnit* FB of the Unit. In this FB the manager configuration is initialized and further *Startup* FBs for the Equipment Modules are called. The OMAC manager configuration comprises the definition of the used operating modes and their states. There is a Startup FB for each Equipment Module to apply the concept of modularization to all organizational blocks. This makes it possible for a programmer to work encapsulated on one Equipment Module. More information about the initialization of the project is given in chapter 3.

2.2 Machine Template

The folder *Machine Template* contains four function blocks. All blocks are structured in the same way. They include the region *BLOCKHEADER*, which contains the essential information of the block, the region *DESCRIPTION*, which describes the content of the block in detail, a region *FIRST CALL*, which initializes the function block. In the region *CYCLIC CALL* variables are evaluated and written cyclically. The specific content of the FBs is processed in one or more regions like *COMMAND* or *CONFIGURATION HANDLING*. In the last region *OUTPUTS* the static variables are copied to the output, the error evaluation takes place and, if necessary, is copied to the program_alarm.

2.3 Unit

In the Machine Template project, the machine is represented by one Unit, according to ISA-88. All unit specific FBs can be found in the folder *Unit*.

There are several DBs that contain all relevant information of one Unit that can be found in the folder *Unit DBs*. The *Status* DB provides all relevant information about the *Unit*. It includes the monitoring and diagnostics for the manager as well as for all subordinate Equipment Modules, Technical Modules and Control Modules of the Unit. The *Commands* DB represents a direct connection from PLC to external interfaces such as HMI, line manager or MES. In *Configuration* DB the configuration of the Unit Mode State Manager as well as the configuration of the Unit modules are defined. This refers to all data that is relevant for the machine and the recipe and must be defined before production. Data which are changed during production on the fly, e.g. the override, are defined in DB *Commands*. The *Hardware* DB contains the complete hardware which is used in the project, e.g. sensors and actors. All DBs are described in detail in chapter 3.

2.4 Equipment Modules

The folder *Equipment Modules* of the Machine Template contains all used Equipment Modules of one Unit. One Equipment Module includes one processing step of the machines production chain. It is divided into two blocks, the sequence and the Technical Module. This splitting is done to separate the sequential user program from the actual technology. There could be more than one Equipment Module in a Unit. Each of them has exactly one sequence and one or more Technical Modules.

The sequence coordinates the different Technical Modules. The user program for the complete Equipment Module is realized as a state machine in the *Sequence* FB which applies equally to all Technical Modules underneath. In the sequence, commands are written to the Technical Modules and the feedback from the Technical Modules are read in and evaluated.

Figure 2-2: Detail view of Equipment Module

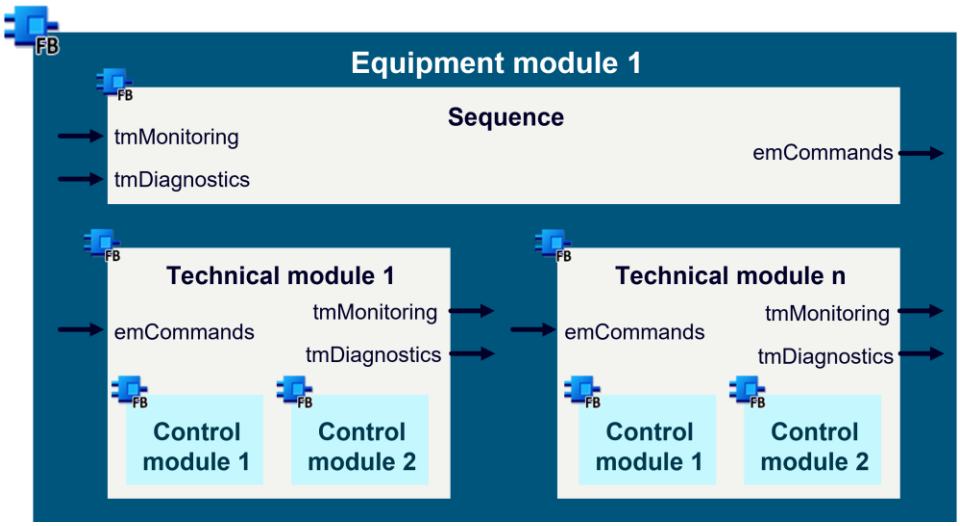
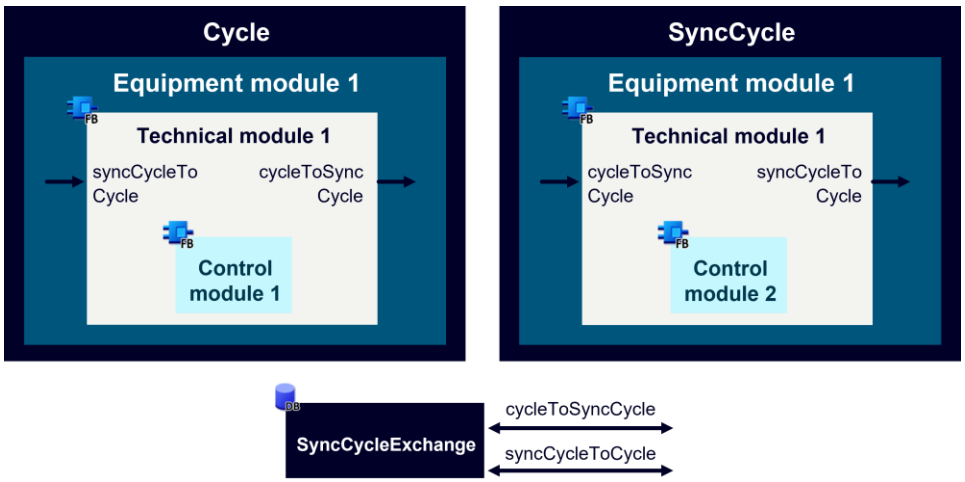


Figure 2-2 shows this handshake between the sequence and the Technical Modules. It is illustrated in the following example. First a positioning command is send from the sequence to one Technical Module via *emCommands*. When the Technical Module has reached the position, the feedback is send to the sequence via *tmMonitoring*. The sequence reads in *tmMonitoring* and evaluates the signals for its state machine. When the state machine of the sequence move to the next state, a following command is send via *emCommands*.

Both blocks, the sequence and the Technical Module, are contained as template FBs in MTplPlc as *MTplPlc_SequenceTemplate* and *MTplPlc_TechModuleTemplate*. An example is implemented in the folder *Equipment Module 1* as *CycleEm01Sequence* and *CycleEm01TechModule01*, which are described in the following sub-chapters in detail.

All Technical Modules of an Equipment Module are always called cyclically. They have the prefix *Cycle* and are found in the folder *Cycle* of an Equipment Module. If a Technical Module contains a Control Module, which must be called in the synchronous cycle, then this Technical Module is called the *sync cycle* additionally. For the communication between cycle and sync cycle, the DB *syncCycleExchangeEmXy* is used within the Equipment Module, see figure 2-3. It has to be parameterized by the user for the respective Technical Module.

Figure 2-3: Sync cycle exchange



2.4.1 Control Modules

Control Modules represent the lowest level in the ISA-88 physical model. They contain no logic. In the Machine Template in Equipment Module 1 there is one Control Module in the cyclic call and one Control Module in the synchronous cyclic call included as an example. It is only intended to show that in this lowest level only signals to the axis, to the actuator or sensor are mapped. If the call of the Control Module is time critical, e.g. actual values or setpoint values have to be connected in a fast cycle, then this has to be done in the sync cycle call environment. There is no template FB provided by this Machine Template, because there is already a separate library, the Library Basic Control (LBC). In this library digital or analog signals are evaluated, actuators can be controlled and other useful functionalities are provided.

Note For more information about the Library Basic Control (LBC), please refer to the according SIOS entry.
https://support.industry.siemens.com/cs/document/109792175

2.5 Alarm handling

To show all errors on HMI devices the system instruction Program_alarm is used to store the alarms in the diagnostic view of the HMI and this is described in the following chapter.

2.5.1 Error detection

The template uses Program_alarm to show all errors. The messages are generated in the language english inside every Sequence and every Technical Module.

Therefore, the Program_alarm is already preconfigured in the FB templates *MTplPlc_SequenceTemplate* and *MTplPlc_TechModuleTemplate*. In case an error occurs in one of these FBs, the Program_alarm is triggered, which outputs the diagnostic information of the respective module. After the message is reported via a Program alarm, the messages can be seen via system mechanism “alarm view” in the HMI panel, the SIMATIC webserver and the TIA Portal Engineering.

First of all, the correct project languages have to be chosen. The messages can only be displayed correctly from the text lists when the languages are configured

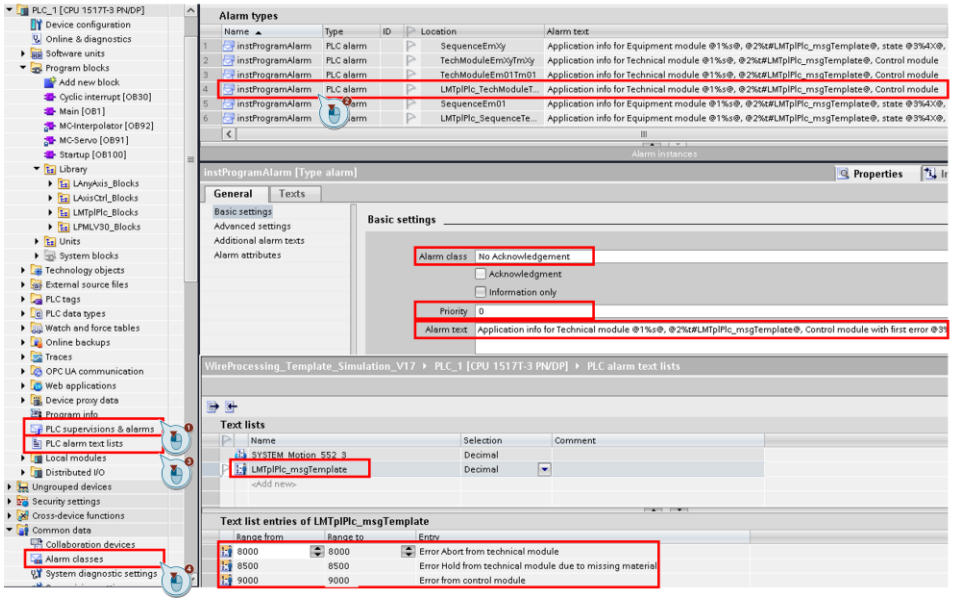
correctly in the device configuration and the according texts are available in this language.

If one or more Control Modules has an error, this is recorded in FB *TechModuleEmXyTmXy* and outputted via the Program_alarm. The FB also detects which Control Module first encountered the error. Via the interface *tmDiagnostics* the error is sent to the FB *SequenceEmXy* of the Equipment Module. All information concerning the error, such as state, substate, IDs of the relevant modules, etc., are stored in DB *StatusUnit01*.

As in FB *TechModuleEmXyTmXy*, the Technical Module where the error occurred first, is recorded at Equipment Module level in FB *SequenceEmXy*. This information is made available in both FBs based on the ID of the respective module via the Program_alarm. A detailed description of the parameters used in the Program_Alarms is given in Chapter 3.3.1 for the Technical Module and in 3.3.2 for the Equipment Module.

An overview of all Program_alarms created in the project is listed in the navigation via *PLC supervisisons & alarms*. It can be seen in figure 2-4, step 1. The Program_alarm can be edited by clicking on the respective alarm, see figure 2-4, step 2.

Figure 2-4: Alarm handling



In the properties of the Program_alarm the user has to configure the alarm class and the priority of the error. The alarm text is also written here. In the Machine Template the alarm texts include all properties that are collected by the Program_alarm. Furthermore, the alarm text is linked to a PLC alarm list, see figure 2-4, step 3. The messages from the Machine Template can be found in the list *MTplPlc_msgTemplate* under the *PLC alarm text lists*. It is already created and filled with three example error messages. If the user needs more alarm classes than defined by default, the alarm classes can be extended, see figure 2-4, step 4.

Note For more information about the system function Program_alarm, please refer to the TIA Portal Online Help.

If the user is using drives in the machine project, it is recommended to additionally use the LAlarmHdl library. This library could be used to collect all messages from SINAMICS drive and to display these messages also on Siemens HMIs via the

standard control “Alarm View”. The integration of this library is not a part of the Machine Template.

Note For more information about the LAlarmHdl library, please refer to the according SIOS entry. https://support.industry.siemens.com/cs/ww/en/view/109761931

2.5.2 Unit Status

For the evaluation of the error status of the machine, the unit-specific DB *StatusUnit01* is available. In the *Status* DB all relevant information of the Unit are saved.

This includes information from the manager, at highest level, to the error status of a Control Module, at lowest level. With these information the user can identify, which Equipment Module of the Unit, which Technical Module and which Control Module causes the error. The diagnostics is shown for every Equipment Module and every Technical Module. The diagnostics of the Control Module with the first error is included in the Technical Module by its ID and its status.

3 Working with the Machine Template

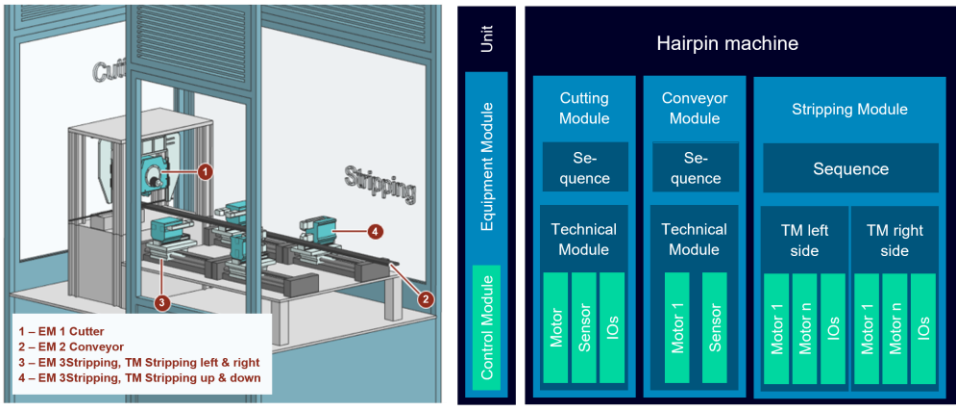
3.1 Subdivision of the machine

The Machine Template serves as a framework for a real machine project. This means that in order to ensure a modular project structure, the technology and functionality of the machine have to be integrated into the Machine Template project. It is therefore not possible to simply integrate the Machine Template functions into an existing machine project.

The first step during concept phase is to subdivide the described machine into the presented modules of the physical model according to ISA-88. This is exemplarily shown for a part of a Hairpin machine. In this example the Hairpin production machine corresponds to a Unit with its own Mode State Manager. The Equipment Modules can be found underneath. These would be, for example, the cutter, the conveyor and the stripping module, illustrated in Figure 3-1.

Each Equipment Module has its own sequence in which the operating modes are programmed by the user. The technology-specific commands are sent from the sequence to the Technical Modules. In the Equipment Module stripping, for example, the processing does not take place in a single step but sequentially one after the other. It makes sense to divide this into two Technical Modules. Both receive the commands from the sequence and report their status back to the sequence, according to Figure 2-2. In the Technical Modules, the axes involved in the machining are implemented as Control Modules. In the Machine Template project, these axes are controlled via the axis control FB from the library *LAxisCtrl*.

Figure 3-1: Dividing machine into modules illustrated by a Hairpin production



3.2 Project configuration

3.2.1 Adaption of User Defined Types

UDTs of folder *MTplPlc*

After the machine has been divided into modules, the hardware configuration and module adaption of the project must be done. The first step is to configure the PLC data types.

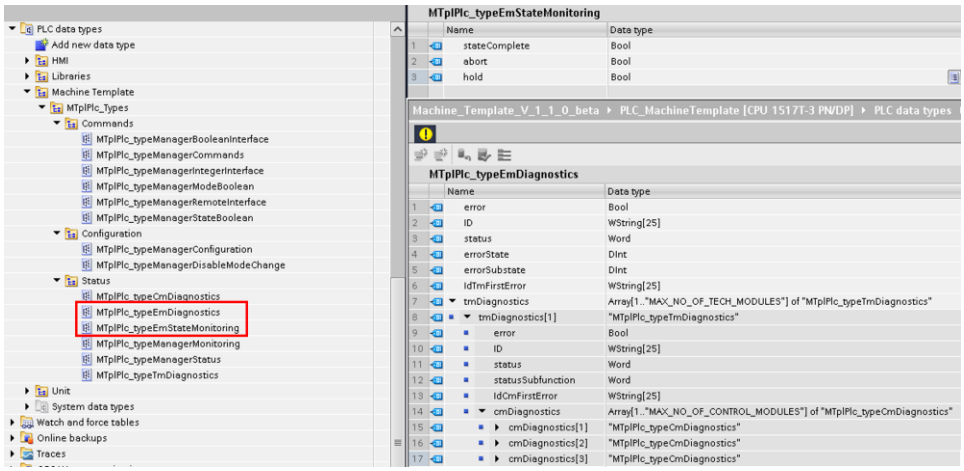
The template MTplPlc offers user defined types (UDTs) that are always valid and do not have to be adapted for a machine project. They include UDTs, which on the one hand concern the commands and the configuration of the Unit Mode State Manager. The Unit Mode State Manager is always unified and must not be adapted by the user in the machine project.

On the other hand, UDTs are provided for the Equipment Module state monitoring and diagnostics for Equipment, Technical and Control Modules, see figure 3-2. The diagnostics are pre-defined by the Machine Template and do not require any adjustment, unless the user wants to change. For such cases the UDTs of the diagnostics could be used as a copy template and should be adjusted accordingly in the *Unit* folder and replaced as data type in the project.

The UDTs regarding the Equipment Module state monitoring are always standardized in the Machine Template and should not be adapted by the user. The signals *stateComplete*, *abort* and *hold* must be written by the sequences of the Equipment Modules in order to ensure a consistent evaluation at the Unit level. This makes it possible that the Unit level of this Machine Template does not have to be adapted by the user.

The UDTs of the monitoring of the Technical Modules are specific for every machine and could be found in the folder *Unit*.

Figure 3-2: MTplPlc UDTs



UDTs of folder *Unit*

In the folder *Unit* all machine-specific UDTs can be found. They all have to be adapted by the user according to the real machine. There are UDTs for the used hardware, the configuration, the status as well as the commands included for controlling the Equipment Modules, Technical Modules and Control Modules.

Since a wide variety of Control Modules such as axes, cams, sensors or actuators can be used, it is not possible to use the UDTs of the Control Modules as arrays. For this reason, separate UDTs are defined for each Control Module, for each Technical Module and also for each Equipment Module, which are then defined in the respective superimposed UDTs.

In the Machine Template project these UDTs are pre-defined for one Equipment Module with one Technical Module and one Control Module. There are also additional UDTs for Equipment Module xy, Technical Module xy and Control Module xy to illustrate how to extend these UDTs.

In the following figures 3-3 to 3-7 the UDTs for the commands, the configuration, the hardware and the status are shown exemplarily for the configuration of the Hairpin example machine of figure 3-1. They are shown to illustrate how the Machine Template project can be adapted to a real machine project. The shown UDTs are not part of the Machine Template PLC project.

Figure 3-3: Unit specific UDTs for commands

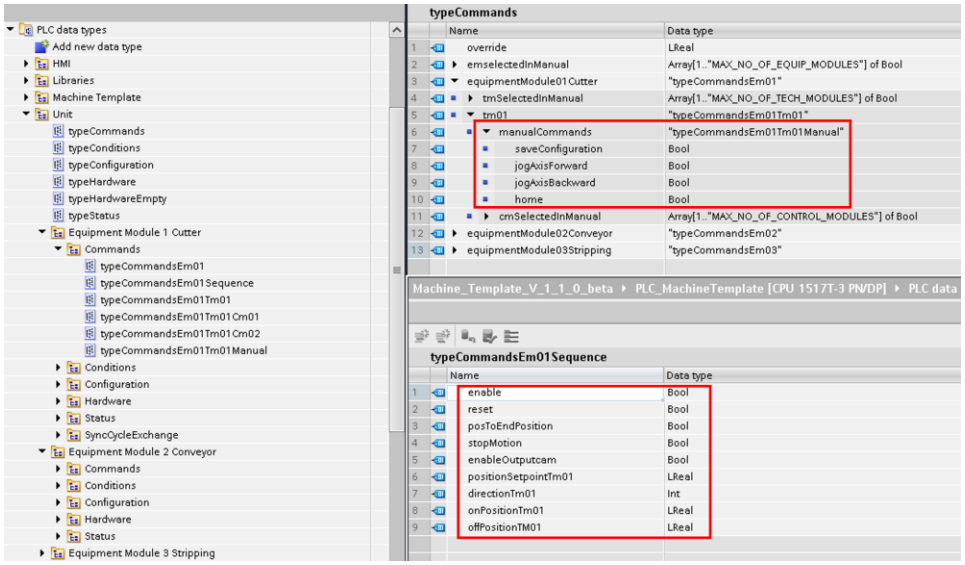
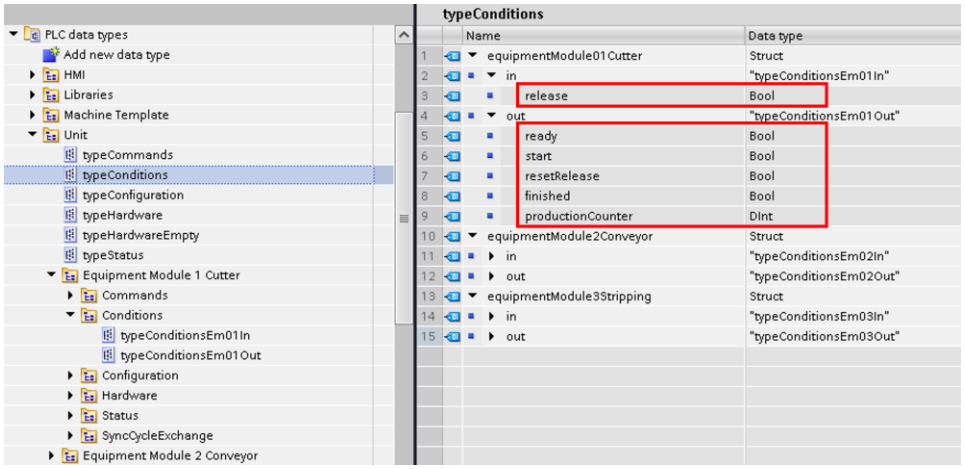


Figure 3-3 shows all relevant commands of the *Unit*. In the navigation you can see that a specific UDT has been created for each module. In the detailed view it becomes clear that the commands of the sequence can be different for each Equipment Module and must be adapted accordingly. Also the *manualCommands*, which are used for the control of a Technical Module in operating mode *Manual*, must be individually adapted.

Figure 3-4 shows the configured conditions of the Equipment Modules. These conditions are used for the communication of the Equipment Modules within the Unit. Therefore, the conditions of the Equipment Modules, e.g. ready or finished, are sent to the Unit. In FB *MTplPlc_UnitConditionCollector* these conditions are evaluated in Unit level. After this evaluation the unit conditions are sent back to the Equipment Modules and read in in the respective *Sequence* FB.

In the Machine Template, example signals are defined that describe the status of an Equipment Module during Production. This allows the coordination of the individual motion sequences of the Equipment Modules, so that the Equipment Modules get released by the unit condition. The UDT *typeConditions* can be specified the same for all Equipment Modules, or if required, individually for each Equipment Module.

Figure 3-4: Unit specific UDTs for conditions

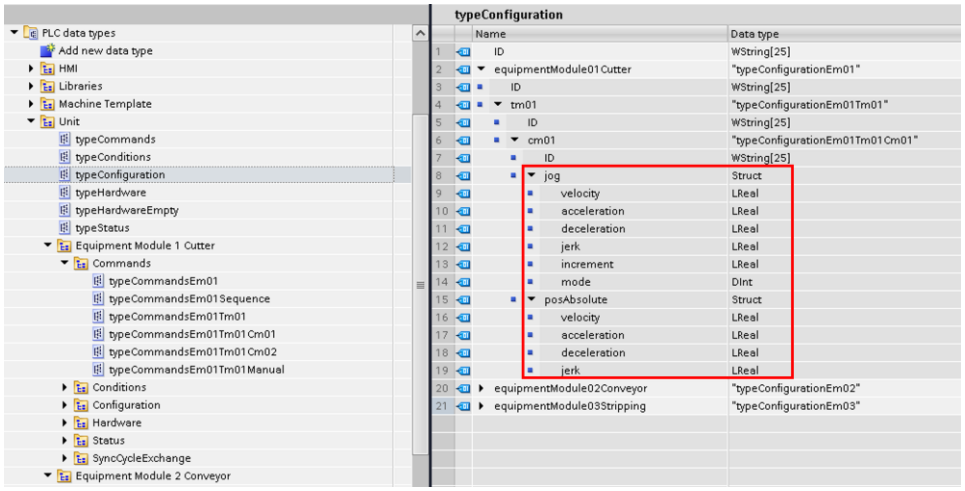


In the detail view of Figure 3-5 it is shown that the configuration parameters of a Control Module consist of the unique ID and the configuration of the relevant parameters. In this example the dynamic parameters of a positioning axis are part

of the configuration, so only the parameters that are necessary for the specific Control module are configured here.

This is the difference compared to other standard FBs like from *LAxisCtrl* library, where all parameters can be specified, regardless of whether they are relevant for the technology or not.

Figure 3-5: Unit specific UDTs for configuration



The detail view of Figure 3-6 shows the structure of the UDT *typeHardware*. It is divided into axisData, inputs and outputs. The axisData consists of the TO axis, master axis, cams and EPos. Typical inputs are sensors, valve or cylinder feedback signals and typical outputs are outputcams, valve and cylinder outputs.

Figure 3-6: Unit specific UDTs for hardware

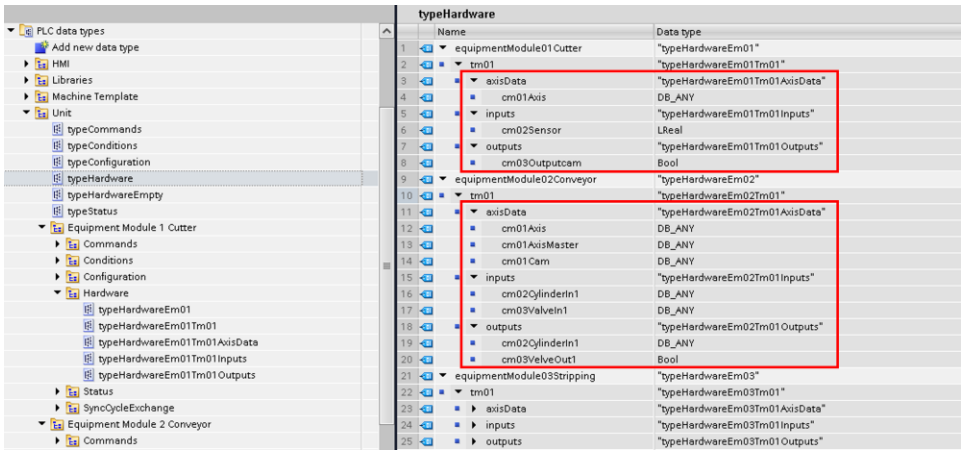
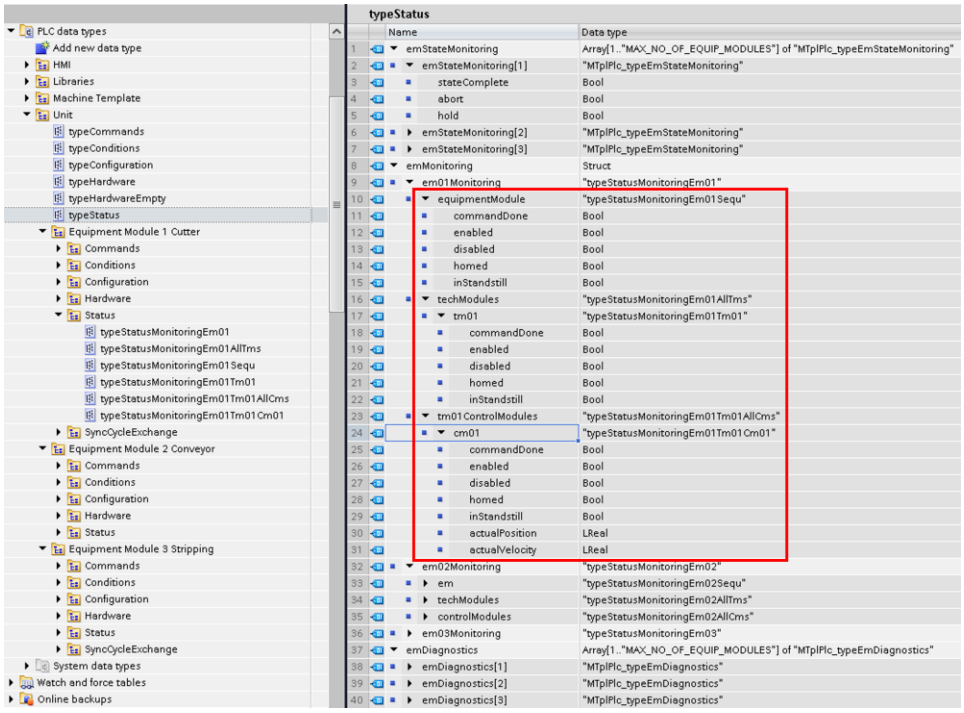


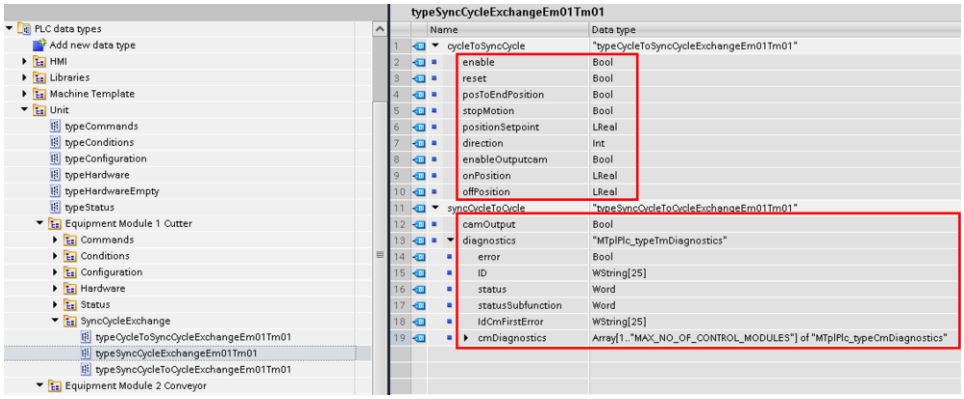
Figure 3-7 shows the feedback signals for all Equipment Modules, Technical Modules and Control Modules. The status signals of the Equipment Modules and the Technical Modules are always the sum of all subordinate modules. For example, the bit *enabled* of the Equipment Module is only TRUE, if all subordinate Technical Modules and their Control Modules are enabled. It becomes clear that all status UDTs have to be defined individually for each Equipment Module dependent on the technology and process.

Figure 3-7: Unit specific UDTs for status



If Technical Modules contain Control Modules that are called in the synchronous cycle, then the generally cycle called Technical Module is additionally called in the sync cycle, see figure 3-8. For this reason, a data exchange between sync cycle and cycle is required, which is configured in the UDT *typeSyncCycleExchangeEmXyTmXy*.

Figure 3-8: Unit specific UDTs for synchronous cycle Exchange



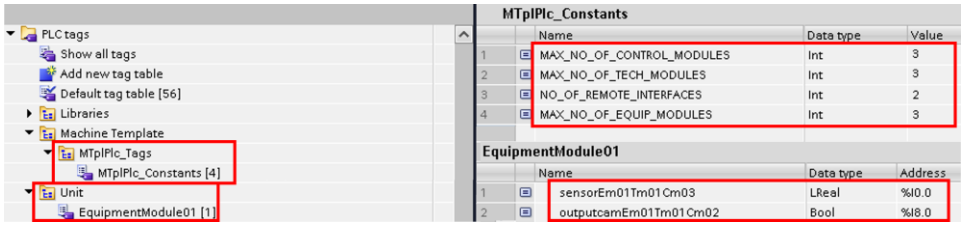
3.2.2 PLC tags of MTplPlc

The tags of *MTplPlc* define, on the one hand, the number of connected remote interfaces. This constant is used in the block *MTplPlc_CommandCollector* to collect the commands of all remote interfaces and to interconnect them with each other. On the other hand, the maximum number of Equipment Modules in a Unit, Technical Modules within one Equipment Module and Control Modules within one Technical Module is configured.

The maximum number of Equipment Modules is used in the *MTplPlc_CommandCollector* FB to collect the feedback of all Equipment Modules via *emMonitoring*. The maximum number of Technical Modules is used for the bundled forwarding of the feedback *tmMonitoring* and *tmDiagnostic* to the respective sequence. In addition, the maximum module number is used for the

selection of the respective modules in operating mode *Manual*. The tags of *MTplPlc* are shown in figure 3-9.

Figure 3-9: MTplPlc tags



The tags that are specific to the Equipment Module 1 of the Unit are also listed as examples in figure 3-9. This is used to connect the inputs, such as sensors or outputs, such as output cams.

3.2.3 Unit DBs

By configuring the UDTs in the *Unit* folder, all DBs in the program blocks in the *Unit DBs* folder are automatically configured correctly. For example the *Commands* DB uses the already preconfigured UDT *MTplPlc_typeManagerCommands* from the folder *MTplPlc* and the user configured UDT *typeCommands* from the folder *Unit*. In the following subchapters the DBs for commands, configuration, hardware and status are described in detail.

Commands DB

The *Commands* DB represents a direct connection from the PLC to all defined remote interfaces such as HMI, line manager, push buttons or OPC UA communication. It consists of two major UDTs, one for controlling the Unit Mode State Manager and one for unit-specific commands for the modules of *Unit*.

The manager UDT is the PLC's interface for controlling the operating modes from a remote interface. The number of external interfaces can be determined via the constant *NO_OF_INTERFACES* which can be defined in the *MTplPlc_Tags*. If the user wants to add another remote interface, then only this constant has to be adjusted.

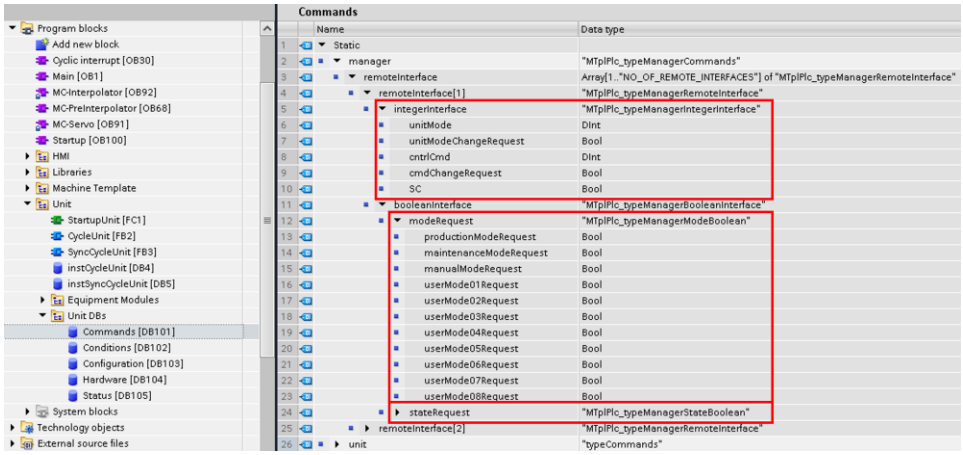
Underneath the UDT *MTplPlc_typeManagerCommands* there is one UDT for controlling the Unit Mode State Manager via Integer and one UDT for controlling the Manager via Bool. The configuration which control variant is used in the project is made in *StartupUnit* FB and must be configured by the user. Only one of the two UDTs is active in the project according to the user's selection. The Unit Mode State Manager is controlled via the commands from a remote interface via DB *Commands.manager*. The structure of the DB can be seen in figure 3-10.

The commands for *Unit* are depending on the module level of the Unit. For example, the override is specified for the Unit level, effecting all modules underneath. The *emCommands* are specified for each Equipment Module to control all of its Technical Modules. In the operating mode *Manual* the Technical Modules are directly controlled from the HMI via *manualCommands*.

In *Manual* mode it might be needed to select only one specific module. Therefore, the boolean variables *SelectedinManual* with the prefix *em* (Equipment Module), *tm* (Technical Module) and *cm* (Contro Module) are used. They are necessary for controlling the modules as well as for the evaluation of the feedback from the relevant modules to the Unit Mode State Manager during *Manual* mode.

The described variables of the *Commands* DB can be seen in Figure 3-3 showing the respective UDT. In contrast to the parameters of the DB *Configuration*, all commands of this DB are directly active in the project. If, for example, a control command is specified via the HMI, it is directly processed in the modules.

Figure 3-10: *Commands* DB



Conditions DB

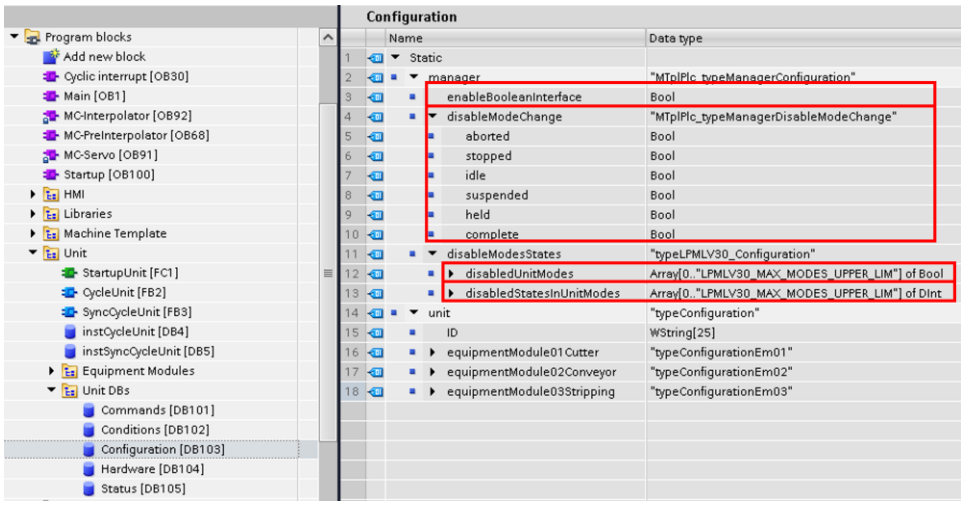
The DB *Conditions* contains the conditions of all Equipment Modules of the Unit. It uses the UDT *typeConditions* in figure 3-4. These are required for the communication of the Equipment Modules within the unit. In real machine projects, Equipment Modules often require the feedback from other Equipment Modules. In the Machine Template, this is illustrated by a series connection of Equipment Modules.

In the example case, all Equipment Modules send feedback signals, such as *finished* to the unit. In FB *MTplPlc_UnitConditionCollector* these conditions are then processed. Unit conditions, such as *release*, are then sent back to the Equipment Modules. In this way, in the example, the step chain is restarted during production.

Configuration DB

In the *Configuration* DB, all configuration parameters are specified for both, the Unit Mode State Manager and all modules of the Unit, see figure 3-11. The configuration of the manager is defined during the project development in the *StartupUnit* FB. In the manager structure the used modes and their used states are set. In addition, it can be configured wether the commands are processed in bool or integer and from which state the user can change to other modes.

Figure 3-11: *Configuration* DB



The configuration of the modules from the Unit is also defined in this DB. They can be adapted via the HMI during runtime. In contrast to the *Commands* DB these

configuration pamareters are not directly active and must be enabled via the *saveConfiguration* command.

The configuration includes an ID that is used to assign a unique name for the Unit and each subordinate module. The configuration of the Control Modules is also included in this DB. It covers the dynamic specifications and other specifications that are needed for controlling the respective Control Modules. The configuration is defined in the UDT *typeConfiguration* and can be seen in figure 3-4.

Hardware DB

The *Hardware* DB is the interface to any hardware used in the PLC project. When signals from hardware components are used in the project or if signals from the project are required for hardware components, these signals are linked via this DB.

In the pure Machine Template project, one Unit with one Equipment Module, one Technical Module is created as an example. To demonstrate how to implement a new module, *Equipment Module xy* is also implemented. The used UDT *typeHardware* of the block must first be adapted by the user in the PLC data types in the folder *Unit*. Figure 3-5 shows the configured UDTs for the Hairpin example machine. It shows hardware axis data for an axis, an input for a sensor as well as a hardware outputs example for an outputcam.

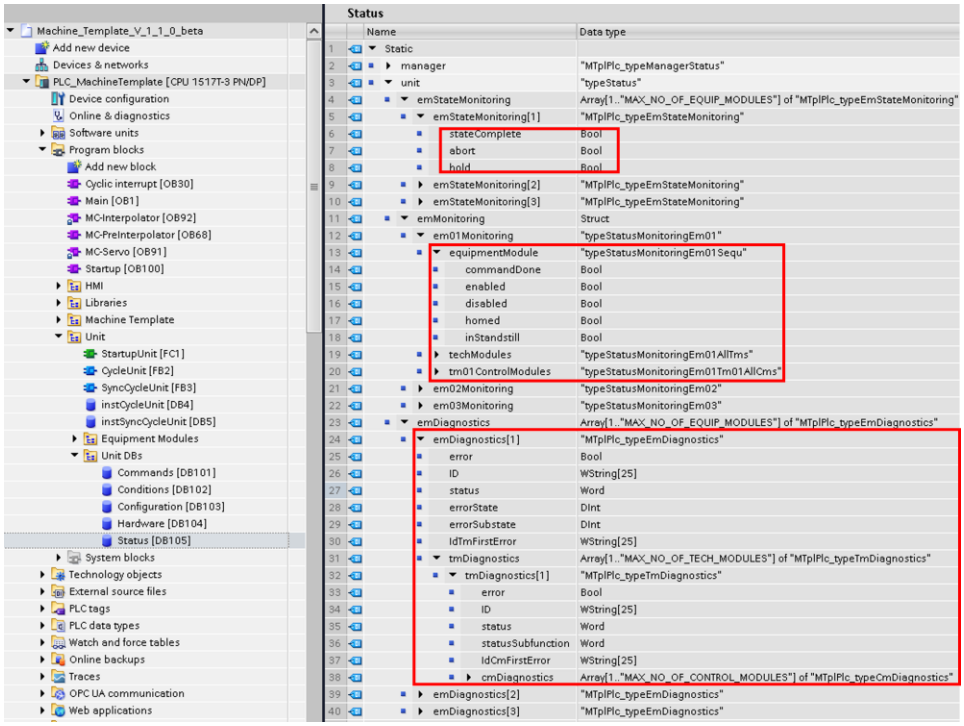
Status DB

The *Status* DB contains all relevant information about the Unit Mode State Manager and the modules of the *Unit*. It includes the monitoring and diagnostics for the manager as well as for the Unit and all its subordinate modules. The monitoring signals represent the state enabling condition for upper level modules whereas the diagnostic signals represent the cancellation condition. With these information the upper module can either move in the next state or has to change to an error case.

In the manager monitoring, the active Unit mode and the active state of this mode is shown as data type *DInt* and *Bool*. The type of the manager diagnostics is the same as in the library LPMLV30. The monitoring of the Equipment Modules provides the feedback to the Unit from the lower modules. This is either a confirmation by *stateComplete* that the active state is processed successfully, or an *abort* or *hold* signal, which correspond to a cancellation condition of the respective state. These signals are processed at the Unit level by the *MTplPlc_UnitCommandCollector* FB.

If an error has occurred within the Unit, it can be monitored in the DB under diagnostics. At Equipment Module level, this is described by a unique ID, as well as by the status, the respective state and a possibly used substate. At the Technical Module level, this information is extended by the unique ID of the Technical Module, a status and a substatus. The described *Status* DB can be seen in the figure 3-12.

Figure 3-12: *Status* DB

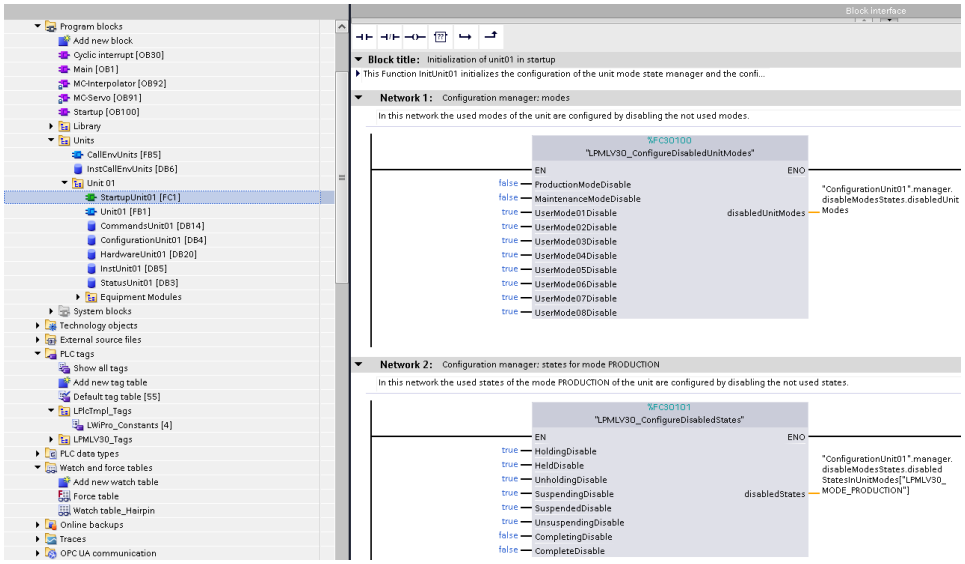


3.2.4 Initialization of machine using StartupUnit FB

After the machine specific UDTs have been adapted and the *Unit DBs* were explained, the initialization of the machine has to be done via FB *StartupUnit*.

Here, first the Unit Mode State Manager is configured. This means that the user selects the operating modes and their associated states. By default, the operating modes *Production* and *Manual* are set. If another operating mode should be added, the disable of the operation mode has to be set to *FALSE*, see Figure 3-13. In the following networks, the used states for the used operating modes are set. The configuration is also done by disabling not used states by setting it to *TRUE*.

Figure 3-13: Configuration of Unit Mode State Manager in *StartupUnit* FB



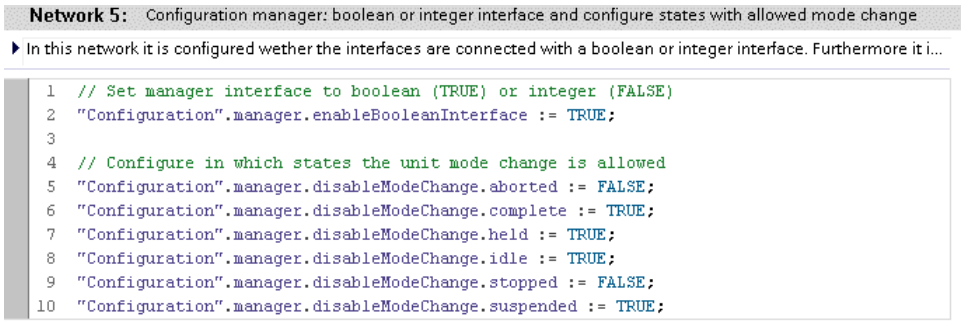
Note

For more information about how to change the configuration of the Mode State Manager, please refer to:
https://support.industry.siemens.com/cs/de/de/view/49970441

In the next network the user configures whether the commands from the remote interfaces are processed via a boolean or an integer interface. All remote interfaces have to use either the boolean or the integer interface and it is not possible to mix between them. Additionally in this network it is specified from which states an operating mode change is allowed or not.

By default, the states *STOPPED* and *ABORTED* are activated for a change, see figure 3-14. The user can allow a mode change from less or more states. However, only changes from the so-called wait states are possible, since in these states no commands are executed.

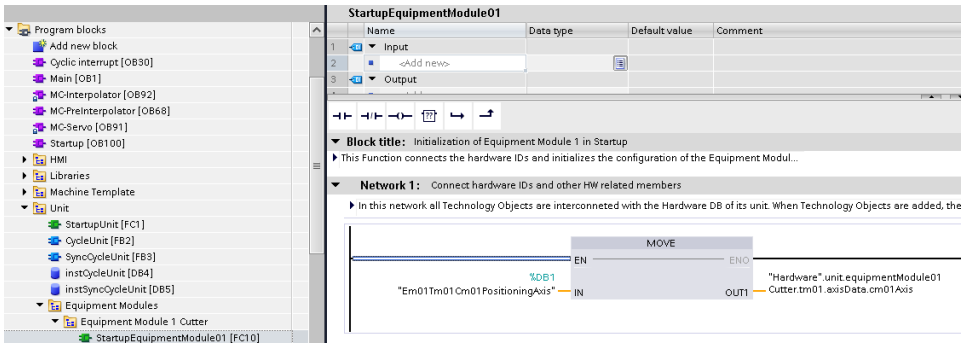
Figure 3-14: Configuration of Manager interface in *StartupUnit* FB



This is followed by the initiation of the Unit itself. This means that the ID for the Unit is assigned and the override is predefined. After that, an initialization of the module selection is done. By default, all modules are selected first. A special selection of individual modules can be done after the startup in the operating mode *Manual* via the HMI. In the following networks, the Startup FBs are called for the respective Equipment Modules.

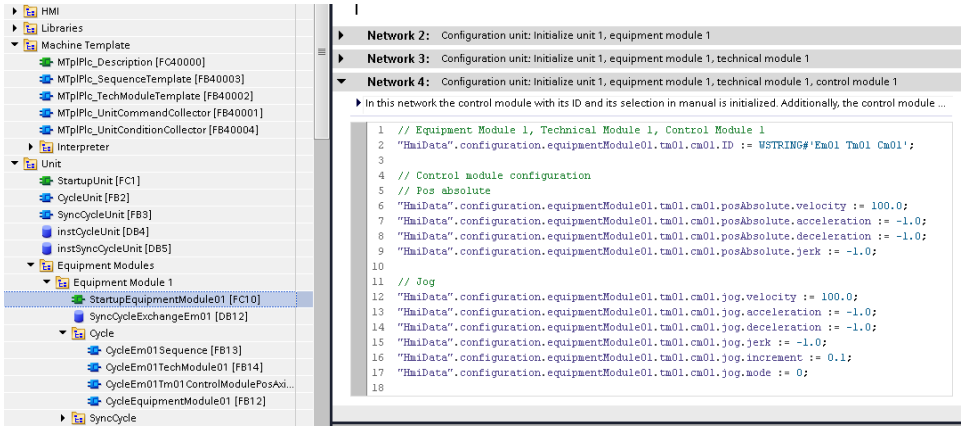
In the first network of the *StartupEquipmentModuleXy* FB the used Technology Objects have to be connected to the DB *Hardware*. Figure 3-15 shows the connection of the TOs for the example of Equipment Module Cutter in the Hairpin machine.

Figure 3-15: Initialization of the Technology Objects in the *StartupEquipmentModule01* FB



The initialization of the configuration of the respective Equipment Module is also done in this FB. A unique ID is assigned for each Equipment Module, Technical Module and Control Module. The selection for the *Manual* mode is pre-configured and the relevant configurations like motion control settings, eg. dynamics limits, are specified for each Control Module, see figure 3-16. At runtime, this configuration can be adapted via the HMI. It is important that every module that is used in the project is initialized here. So in case a user adds a module, it has to be predefined in this FB.

Figure 3-16: Configuration of Unit in *StartupEquipmentModule01* FB



3.3 Module adaption

This chapter describes how to adapt the existing modules to the project and how to add new modules.

3.3.1 Adaption of the Technical Module

Control Module

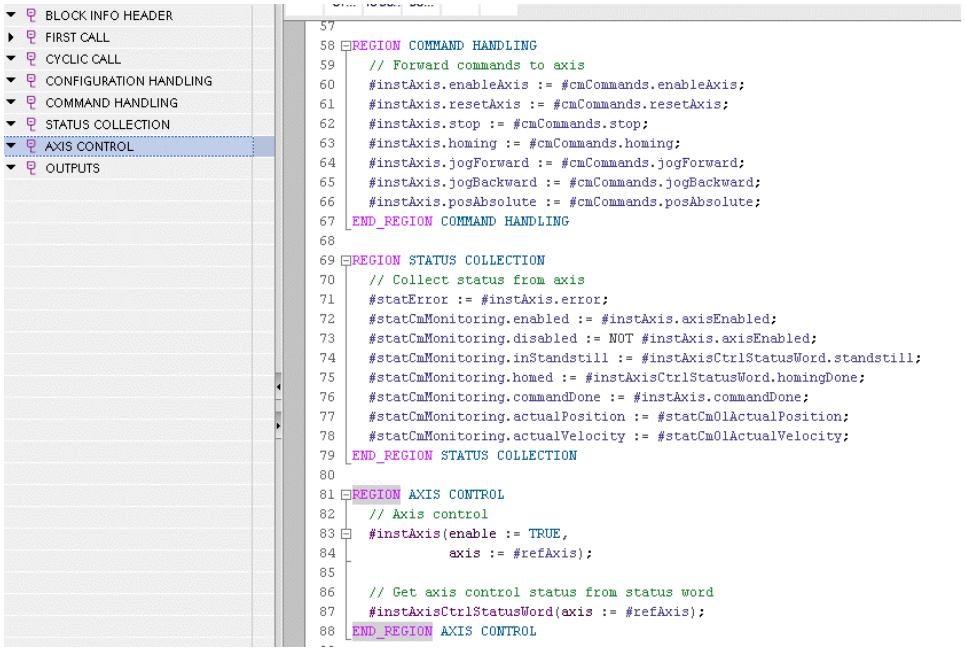
The Control Module should be a module that can no longer be divided representing the lowest level. It controls, for example, an axis or an output cam for an actuator or it provides the signal of a sensor. A Control Module does not contain any technology functions.

In the MTplPlc library, no template is provided for the Control Modules, as they are too different from each other to offer a universally valid template. However, an example Control Module has been created for a cyclic call of a positioning axis with FB *CycleControlModulePosAxis* and an example Control Module has been created for a synchronous call of a outputcam with *SyncCycleControlModuleOutputcam*.

In the *CycleControlModulePosAxis* FB, the Control Module is the interface between the Technical Module and the axis which should be controlled, see figure 3-17. The commands from the Technical Module are read in and passed on to the axis via the AxisCtrl FB. The feedback of the Axis Ctrl FB and additionally the status word of the axis is read out in the Control Module so that all information about the axis can be forwarded to the Technical Module.

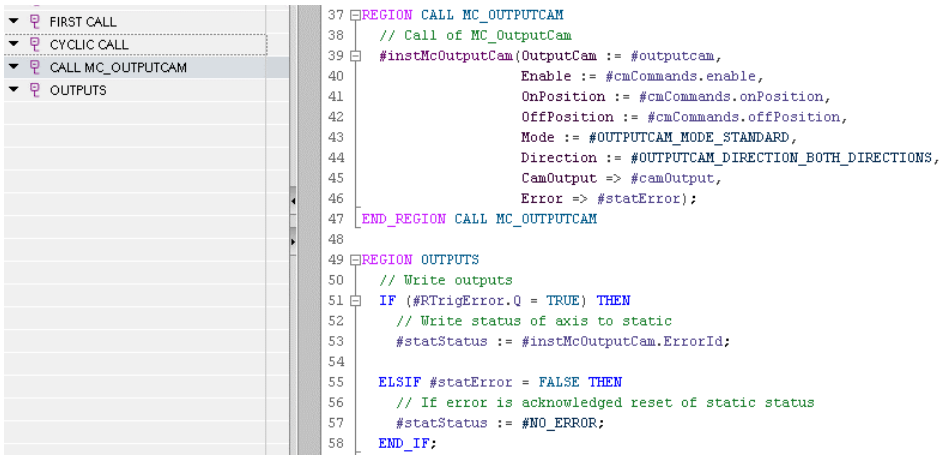
This interface via the Control Module is used so that the user only has to specify the commands and the configuration, e.g. dynamic parameters, that are actually needed for the technology. Thus, it is not necessary to parameterize all parameters of the AxisCtrl FB that are not needed. The interface is previously defined by the UDTs *typeCommandsEm01Tm01Cm01* and *typeConfigurationEm01Tm01Cm01*. This keeps the FBs and DBs clear and the interface lean.

Figure 3-17: *CycleControlModulePosAxis* FB



In the FB *SyncCycleControlModuleOutputcam*, the Control Module is not called in cycle call like the conventional modules, but in synchronous cycle. Some functionalities require that actual values are read or setpoints are written in a faster cycle. This is illustrated by the Control Module outputcam. Here the actual value of the axis is read in the PreInterpolator cycle and the output cam is written again in this clock, see figure 3-18.

Figure 3-18: *SyncCycleControlModuleOutputcam* FB



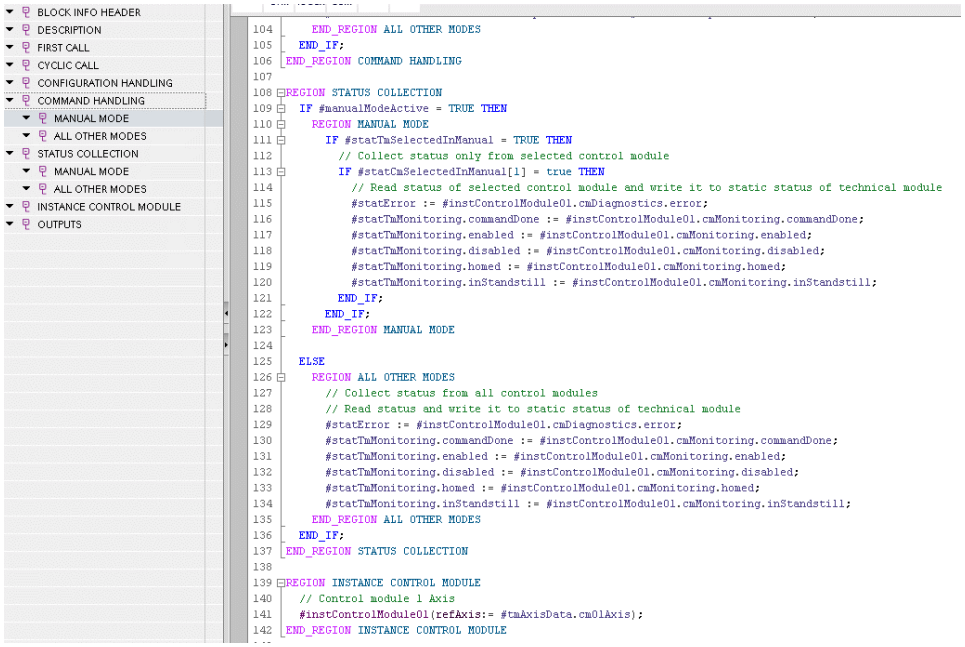
Technical Module

In the Machine Template the FB *CycleEm01TechModule01* is already prepared. It is a copy of the function block template *MTplPlc_TechModuleTemplate*. The FB of the Technical Module has two main tasks. On the one hand, in the region *COMMAND HANDLING* the commands that come from the sequence of the Equipment Module and the commands that come directly from the HMI are linked and forwarded to the Control Modules. On the other hand, in the region *STATUS COLLECTION*, the feedbacks from the control modules are collected and written to static variables.

The user must check the Technical Module whether all previously defined commands of the two interfaces *emCommands* and *manualCommands* are correctly passed on to the Control Module. It must be also checked that all required

information of the Control Modules are evaluated in order to be able to forward them to the sequence as a state enable or cancellation condition. Figure 3-19 shows the commands handling and status collection for the Technical Module 1 of Equipment Module 1.

Figure 3-19: *CycleEm01TechModule01* - Command handling and status collection

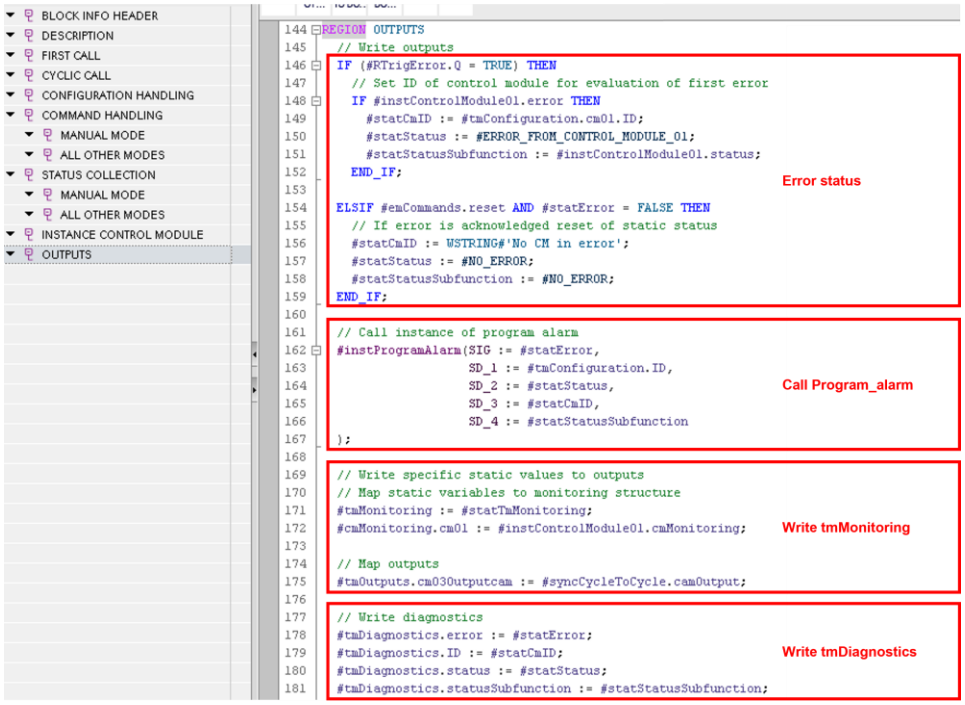


In addition, all functions concerning the actual technology are executed in the Technical Module. These functions could be technology-specific calculations, cam calculations or a state machine for processing a certain task which have to be implemented by the user. For example there are situations where not all Control Modules of one Technical Module could be moved at the same time. Therefore, a logic can be implemented in the Technical Module to move certain Controle Modules one after the other.

Figure 3-20 shows the region *Outputs*. Here the error status is set and the outputs *tmMonitoring* and *tmDiagnostics* are written. To visualize the errors of the FB and the subordinate Control Modules also on the operator panel, the Program_alarm is called and triggered by *statError*. The parameters of the Program_alarm are filled with the ID of the Technical Module, the status of the FB, the ID of the Control Module, which first reports an error and its substatus. These parameters are then outputted as alarm text on the HMI, see chapter 2.4.

If a synchronously called Control Module is required in a Technical Module, there is also a FB in the sync cycle call at the level of the Technical Module. This FB ensures the data exchange of cycle to sync cycle and visa versa. This is illustrated in figure 2-3.

Figure 3-20: *CycleEm01TechModule01* - Outputs

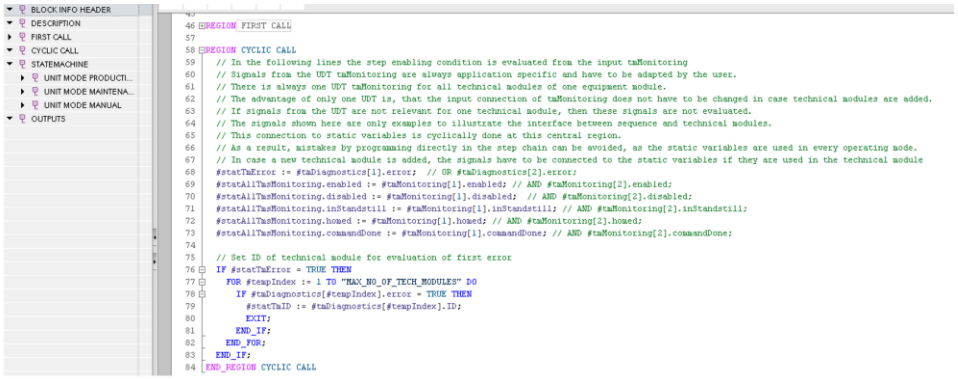


3.3.2 Adaption of the Equipment Module

Sequence

The FB *SequenceEm01* is also already created in the folder *Equipment Module 1*. It is a copy of the template *MTplPlc_SequenceTemplate*. First of all, the user has to check whether all feedback signals of the Technical Modules are collected and processed via *tmMonitoring* and *tmDiagnostics* in the region *CYCLIC CALL*. Figure 3-21 shows that all signals from the relevant Technical Modules in *tmMonitoring* are connected by *AND*. The error from *tmDiagnostics* is collected by *OR* to through an error also when only one Technical Module has an issue.

Figure 3-21: *SequenceEm01* - collection of feedback signals



In addition, the initial error evaluation can be seen in figure 3-21. If an error from one of the Technical Modules is pending, it is evaluated at this point, in which Technical Module the error occurred first. The ID of this technology module is then output via the Program_alarm of the *SequenceEmXy* FB.

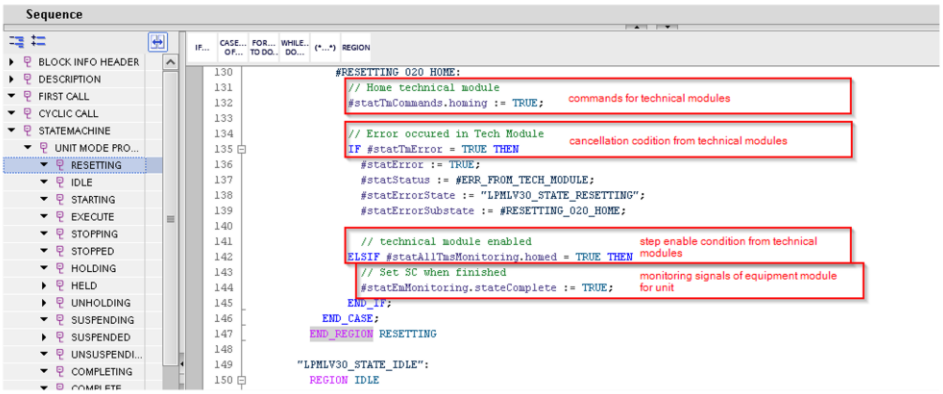
Afterwards the state machine has to be adapted to the real machine. The comments of the individual states give hints which command is sent in the respective state. On the one hand, the commands for the Technical Modules are written in the states and forwarded to the Technical Modules via the output

emCommands. On the other hand the variables *stateComplete*, *hold* and *abort* from *emStateMonitoring* must be written in the state machine in order to give feedback to the superimposed Unit.

An exemplary implementation of this is illustrated in Figure 3-22. In the region *RESETTING* the homing command is sent to the Technical Modules. When all Technical modules are homed, then this is evaluated via *tmMonitoring*. The state could thus be successfully completed and sets a *stateComplete* signal for the superimposed Unit. The Unit changes the state when the *stateComplete* signal has been received from all Equipment Modules.

If the Technical Modules report an error, this is analyzed via *tmDiagnostics* in the sequence of the Equipment Module. The sequence evaluates whether the error forces an *abort* or *hold* feedback by checking the error code. Errors from 8000 to 8999 are related to *abort* and errors from 9000 to 9999 are related to *hold*. As a result, the complete Equipment Module also goes to error and reports the corresponding *abort* or *hold* via *emStateMonitoring* to the Unit. The Unit will then initiate a state change to *ABORTING* or *HOLDING*.

Figure 3-22: SequenceEm01 - state machine



If an additional operating mode is required for the Equipment Module, it must first be activated in the *StartupUnit* FB, see Chapter 3.2.4. After that, the state machine in the *SequenceEm01* FB is extended. Figure 3-23 shows how to copy and paste an existing operating mode in the state machine. Subsequently the user has to adapt the individual states to the requirements of the new defined operating mode.

Figure 3-23: SequenceEm01 - add operating mode

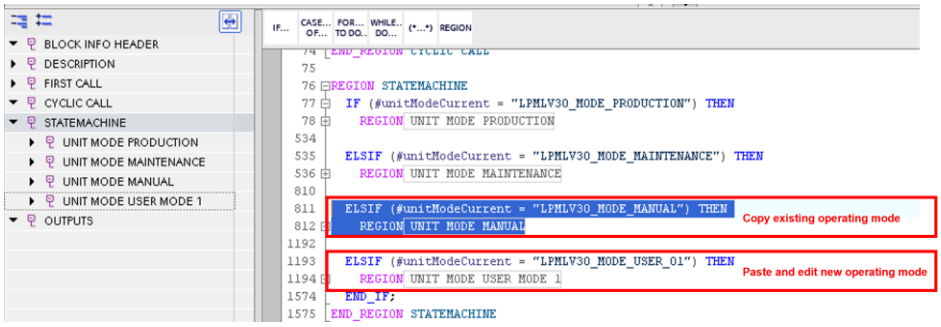


Figure 3-24 shows the region *Outputs*. Depending on the error status, an *abort* or *hold* is first written in *emStateMonitoring*. This is followed by calling the Program_Alarm. The parameters of the Program_Alarm are filled with the ID of the Equipment Module, the status of the FB, the state and the substate when the error occured. These parameters are then output as alarm text at the HMI, see chapter 2.4.

Figure 3-24: SequenceEm01 - Outputs

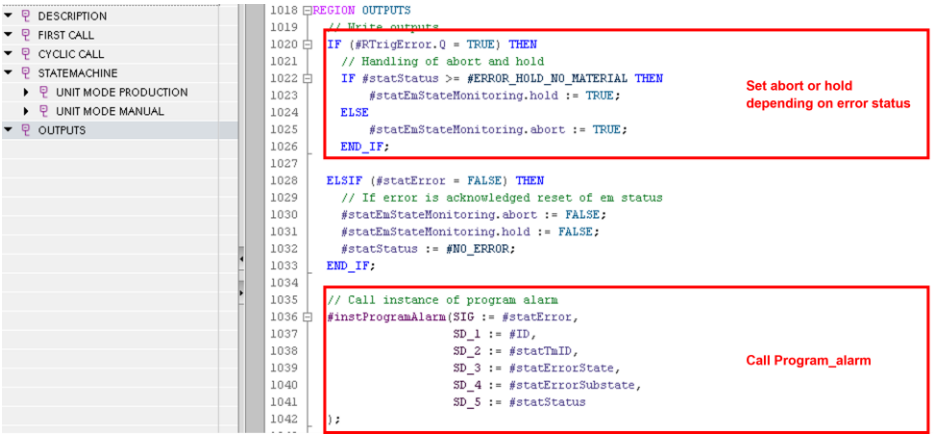


Figure 3-25 illustrates the case when the user needs to add an error. To do this, first a constant must be created in the FB and an appropriate error number has to be assigned. The error must then be triggered in the program and the status with the just defined constant must be written to the status of the Program_Alarm. After that, the specified error number must be added to the alarm list, see Figure 3-25.

Figure 3-25: SequenceEm01 - add new alarm

SequenceEm01

	Name	Data type	Default value
	Constant		
	NO_ERROR	Word	16#7000
	ERROR_ABORT_FROM_TECH_MODULE	Word	16#8000
	ERROR_ABORT_EXAMPLE	Word	16#8001
	ERROR_HOLD_NO_MATERIAL	Word	16#8500

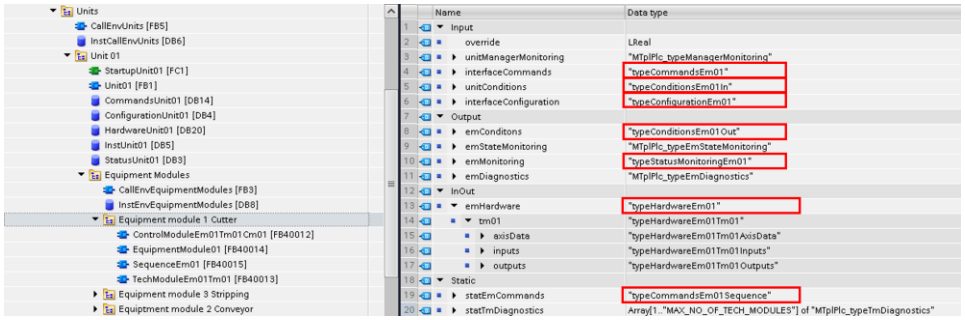
Text list entries of LMTplPlc_msgTemplate

Range from	Range to	Entry
8000	8000	Error Abort from technical module
8001	8001	Error Abort example
8500	8500	Error Hold from technical module due to missing material
9000	9000	Error from control module

Equipment Module 1

In the last step of the adaptation of an Equipment Module, *EquipmentModule01* is checked. For this purpose, the block interface is varified whether all data tyes are assigned to the correct modules of *Equipment Module 1*, see figure 3-26.

Figure 3-26: Equipment Module 1 - block interface



The first and the last network is reserved for the exchange of hardware data. These remain empty in the machine template and must be connected by the user accordingly to the real hardware. In the first network, the hardware data is read in and in the last network it is written out.

In the *CycleEquipmentModule01* FB, the *CycleEm01Sequence* of the Equipment Module and all Technical Modules, in this example only *CycleEm01TechModule01*, are instantiated. The user must check the correct connetion of the input and output interfaces. Figure 3-27 illustrates the linking of the DBs of *Unit* and the linking of Sequence and Technical Modules. In the Machine Template the Technical Modules are always linked to the Sequence. A link between each other is not intended. Only in case a Technical Module also consists of a synchronously called Control Module, it is connected to its synchrous cycle part via the DB **SyncCycleExchangeXy*.

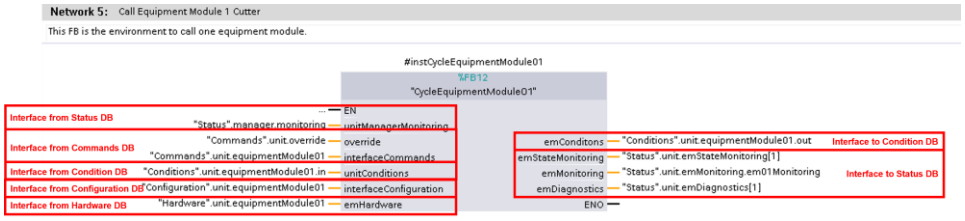
Figure 3-27: Equipment Module 1 - FB interfaces

3.3.3 Adaption of the Unit

For the adaption of the Machine Template to the real machine in the Unit level no adjustments are required. This is ensured by the standardized interface of the Equipment Module *emStateMonitoring* and the standardized interface to the remote interfaces *remoteInterface*. All adjustments are made in the previous described subordinate levels.

As all Equipment Modules are instantiated in the cyclic call of the Unit, there interface to the Unit DBs has to be checked here, see figure 3-28.

Figure 3-28: Cycle Unit - Call of Equipment Modules



3.4 Module extension

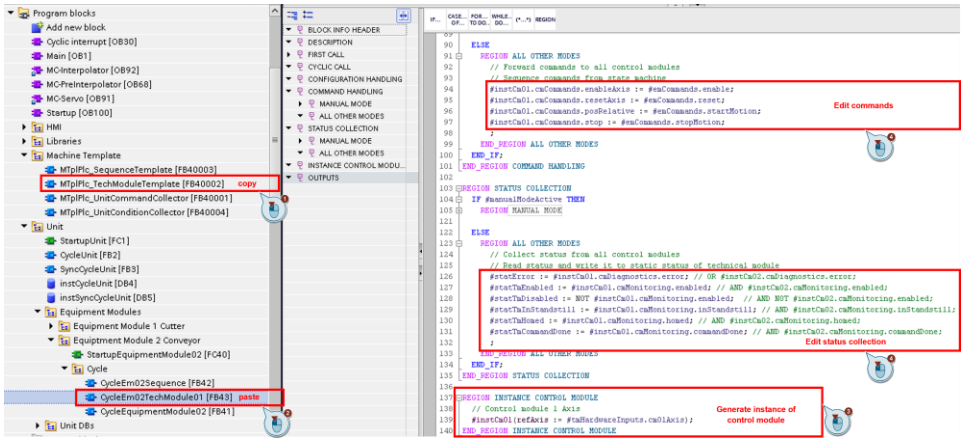
If the adaptation of the already in the project integrated modules is completed and even more modules are needed for the machine project, then these modules can be created from the templates. This is described in detail below.

3.4.1 Implementation of a Technical Module

The expansion of the machine project is illustrated by the example of the hairpin machine from Figure 3-1. The adaptation of *Equipment Module 1* has been

completed and *Equipment Module 2* is now created. To do this, a separate folder *Equipment Module 2 Conveyor* is first created. In this folder, the Technical module *CycleEm02TechModule01* is generated. Therefore, the template *MTplPlc_TechModuleTemplate* must be copied into the folder and accordingly renamed, see figure 3-29.

Figure 3-29: FB *CycleEm02TechModule01*



First, the inputs and their data type have to be checked and adapted to the corresponding Technical Module. Next, the Control Modules, which are used in the Technical Module, have to be instantiated. In the conveyor example the Control Module *CycleEm01Tm01ControlModulePosAxis* can be used as a template, which is already used in the Equipment Module Cutter. The Control Module must be adapted according to the function which should be performed.

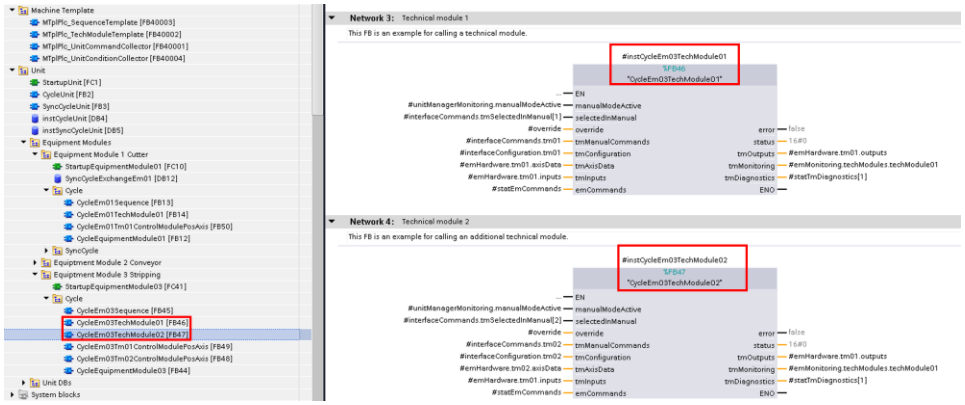
Furthermore, the user must ensure that all commands are forwarded to the Control Modules and their feedback signals are collected. This can be seen in figure 3-29 on the example of one Control Module. If more then one Control Module is used, the commands of all Control Modules have to be connected and all status have to be collected. In addition, the technology-specific functions such as a cam calculation can be implemented by the user.

3.4.2 Implementation of a Equipment Module

When all Technical Module are implemented, the FB for the sequence has to be added. The implementation of the sequence is done by using the template *MTplPlc_SequenceTemplate*, described in chapter 3.3.2. As already described, the state machine has to be adapted accoring to the requirements of the new Equipment Module.

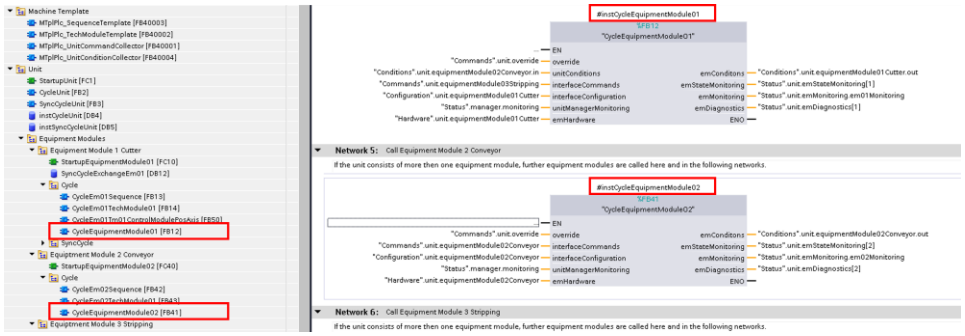
In the next step the FB for the Equipment Module can be added. This is illustrated in Figure 3-30 with the example of Equipment Module 3 Stripping. In the *EquipmentModuleEm03* FB, first the sequence and then all Technical Modules are called as multi-instances. It is important to ensure that all input and output interfaces are correctly connected. For example, the Technical Modules of this Equipment Module Stripping do not have any hardware outputs, so the output *tmHardwareOutputs* have to be deleted.

Figure 3-30: FB *CycleEquipmentModule03*



The last step of implementing an Equipment Module is shown for the *CycleEquipmentModule03* FB. It is called in FB *CycleUnit*, see figure 3-31. The interface connection to the Unit DBs is finally checked. This completes the adaptation of the template to a real machine project with several Equipment and Technical Modules.

Figure 3-31: FB *CycleUnit* Extension



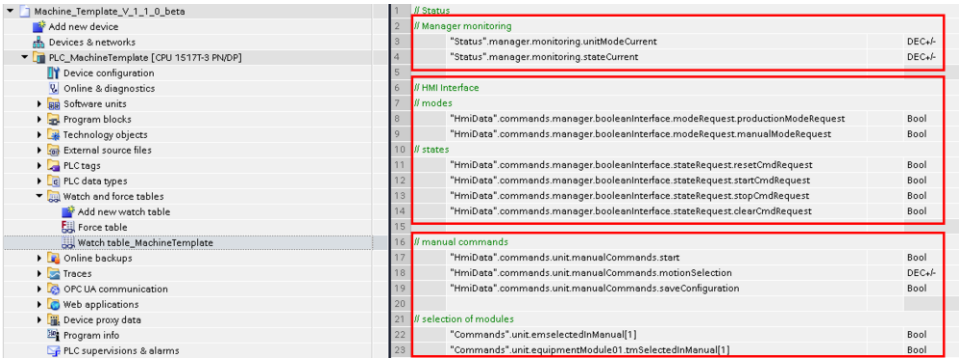
3.5 Project testing

The machine template can be tested using a watch table. This is possible if the TOs used in the project are selected as virtual and in simulation. PLC Sim or PLC Sim Advanced can be used for simulation. To operate the machine template the watch table *WatchTable_MachineTemplate* from Figure 3-32 is created.

This enables the user to control the Unit Mode State Manager. In the watch table *remoteInterface[1]* is created. In the upper part of the table, the status of the Unit Mode State Manager is shown. When the virtual PLC is switched to run, the Unit Mode State Manager begins in the operating mode *Manual* with the state *Stopped*.

The Mode State Manager can be controlled via the commands from the boolean interface. For example, the Boolean command *resetCmdRequest* moves the Unit Mode State Manager via the state *Resetting* to the state *Idle*. Now the module that wants to be moved can be selected under *module selection*. In *Manual* mode the operation is done via the *manual commands*, e.g. *start*, *safeConfiguration*. The command *motionSelection* defines whether the axis is homed or jogged forward or backward.

Figure 3-32: Watch table Machine Template



By using the Boolean command *productionModeRequest*, the operating mode can be changed from *Manual* to *Production*. If the Mode State Manager is in the state *Idle*, the user can control the Unit Mode State Manager with the boolean command *startCmdRequest* to move to the state *Execute* via the state *Starting*. Now the programmed process from the state *Execute* is started. By using the boolean command *stopCmdRequest*, this process can be interrupted again by changing the state to *Stopping*.

If the control via the integer commands is desired, this must first be adapted in the *StartupUnit* FB.

The control of the Machine Template can also be done via the HMI.

4 Program blocks

4.1.1 MTplPlc_SequenceTemplate (FB)

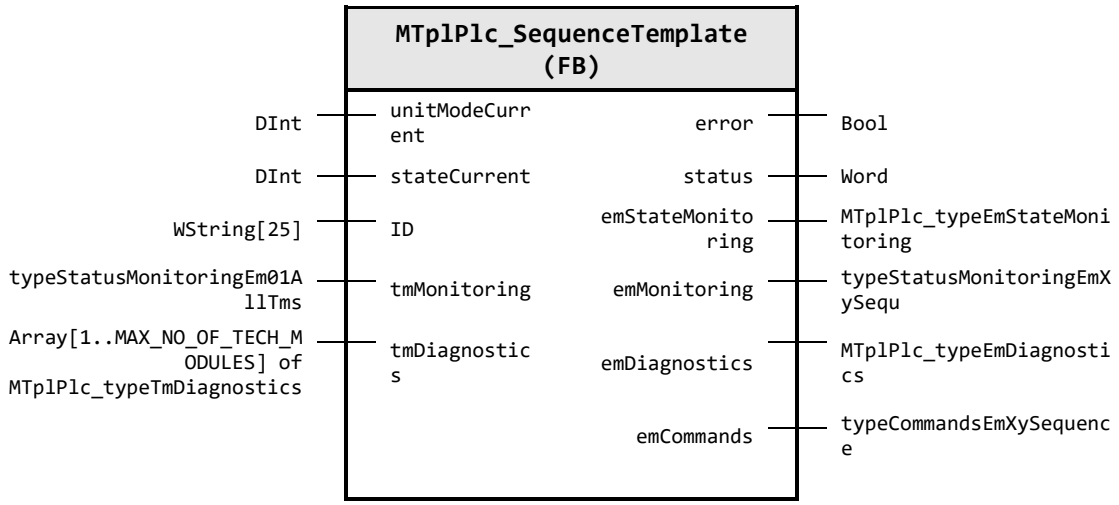
Author: APC ERLF

Short description

The FB MTplPlc_SequenceTemplate is a template for the sequence of an Equipment Module. It contains a state machine with all used modes and states.

Interface description

Block interface



Input parameter

Identifier	Data type	Default value	Description
unitModeCurrent	DInt	0	Current unit mode
stateCurrent	DInt	0	Current unit state
ID	WString[25]	---	ID for Equipment Module
tmMonitoring	typeStatusMonitoringEm01AllTms	---	Monitoring of functional status from Technical Modules of Equipment Module xy
tmDiagnostics	Array[1..MAX_NO_OF_TECH_MODULES] of MTplPlc_typeTmDiagnostics	---	Status of Technical Module

Output parameter

Identifier	Data type	Description
error	Bool	TRUE: An error occurred during the execution of the FB
status	Word	16#0000 - 16#7FFF: Status of the FB, 16#8000 - 16#FFFF: Error identification
emStateMonitoring	MTplPlc_typeEmStateMonitoring	Monitoring signals of Equipment Module
emMonitoring	typeStatusMonitoringEmXySequ	Monitoring signals of Equipment Module

Identifier	Data type	Description
emDiagnostics	MTplPlc_typeEmDiagnostics	Diagnostics of Equipment Module
emCommands	typeCommandsEmXySequence	Commands of Equipment Module for controlling the Technical Modules

Status & Error codes

Code / Value	Identifier / Description
16#8000	ERROR_ABORT_FROM_TECH_MODULE Error Abort from Technical Module
16#8500	ERROR_HOLD_NO_MATERIAL Error Hold due to missing material

User defined datatype(s)

MTplPlc_typeTmDiagnostics (UDT)

Diagnostics of Technical Module provided for sequence.

Identifier	Data type	Default value	Description
error	Bool	FALSE	TRUE: Error
ID	WString[25]	---	ID for technical module
status	Word	16#0000	Status of technical module
statusSubfunction	Word	16#0000	Status or return value of called control module
IdCmFirstError	WString[25]	---	ID of control module with first error
cmDiagnostics	Array[1..MAX_NO_OF_CONTROL_MODULES] of MTplPlc_typeCmDiagnostics	---	Diagnostics of technical module

MTplPlc_typeEmStateMonitoring (UDT)

Shows the Equipment Module status with *stateComplete*, *abort* and *hold*.

Identifier	Data type	Default value	Description
stateComplete	Bool	FALSE	TRUE: State complete
abort	Bool	FALSE	TRUE: Command abort due to error in technical module
hold	Bool	FALSE	TRUE: Command hold due to lack of meterial at input of equipment module

MTplPlc_typeEmDiagnostics (UDT)

Equipment Module diagnostics

Identifier	Data type	Default value	Description
error	Bool	FALSE	TRUE: Error
ID	WString[25]	---	ID of equipment module
status	Word	16#0000	Status of technical module
errorState	DInt	0	State in where the error occurred
errorSubstate	DInt	0	Substate of the main state where the error occurred
IdTmFirstError	WString[25]	---	ID of control module with first error
tmDiagnostics	Array[1..MAX_NO_OF_TECH_MODULES] of MTplPlc_typeTmDiagnostics	---	Diagnostics of technical module

Functional description

There is exactly one sequence for each Equipment Module and one or more Technical Modules. Thus, the sequence is intended to separate the user program of an Equipment Module from the actual Technical Modules. Like all function blocks from the template *MTplPlc*, this block includes the regions *FIRST CALL*, *CYCLIC CALL* and *OUTPUTS*.

The first region *FIRST CALL* initializes the function block. *CYCLIC CALL* cyclically collects the feedback signals from the Technical Modules and sums them up to static variables representing the status of the whole Equipment Module. Additionally, in the event of an error, it is evaluated in which Technical Module the error occurred first.

In the region *OUTPUTS*, the static variables are copied to the output, the error evaluation takes place and is copied to the Program_alarm. The main region of the function block is the *STATE MACHINE*. This region contains all used modes with their used states. In the template, regions for the operating modes *Production*, *Maintenance* and *Manual* are already created. Each of them contain all states. However, the states do not contain any programs, but only hints what could be programming. The variables for *emMonitoring*, which are required for the feedback to the unit, are already prepared to make the programming easier for to user.

The *MTplPLC_SequenceTemplate* FB should be copied into the respective Equipment Module. Then the operating modes within the state machine have to be programmed specifically for the Equipment Module by the user. For the operating mode *Production*, the production sequence, for example, is programmed in the state *Execute*. The operating mode *Manual* works different because there is no manual sequence which is to be programmed in the state *Execute*. So instead, the state *Execute* remains empty and the Technical Modules receive their commands directly from HMI via the interface *manualCommands*.

For the state change condition and abort criteria of the substates in the state machine, the inputs *tmMonitoring* and *tmDiagnostics* are used. The array of the UDT *typeMonitoringUnit01EMXy* is specific for the Equipment Module and has to be adapted by the user in the folder *Application* in the PLC data types. The array of the UDT *MTplPlc_typeTmDiagnostics* is standardized and does not have to be adapted. Both inputs collect the feedback from all Technical Modules.

The monitoring inputs of the Technical Modules include forwarding conditions such as enabled, homed, commandDone or a start condition. The inputs from *tmDiagnostics* include the abort criterion of the state, i.e. the error status of the Technical Modules with its discription. At the output of the block, on the one hand,

the status of the Equipment Module for the superimposed unit and on the other hand the commands for the subordinate level of the Technical Modules is provided.

emMonitoring shows the result of the individual state for the unit. Here either a *stateComplete* signal is set if the state is successfully processed or an *abort* or *hold* signal is set in case of an error. This is supplemented by the output *emDiagnostics* where the error case is described in more detail. The output *emCommands* contains all relevant commands for the Technical Modules. The variables are strongly dependent on the specific Equipment Modules. It is therefore an application-specific type, which has to be adapted by the user during project development, see chapter 3.2.2.

Change log

Version & Date	Change description
01.00.00	**APC ERLF**
31.01.2022	First released version
01.00.01	**APC ERLF**
31.03.2022	Bugfix outputs hold and abort were mixed up
01.00.02	**APC ERLF**
31.03.2022	Bugfix abort and hold are triggered by rising edge of statError & reset by state ABORTING

4.1.2 MTplPlc_TechModuleTemplate (FB)

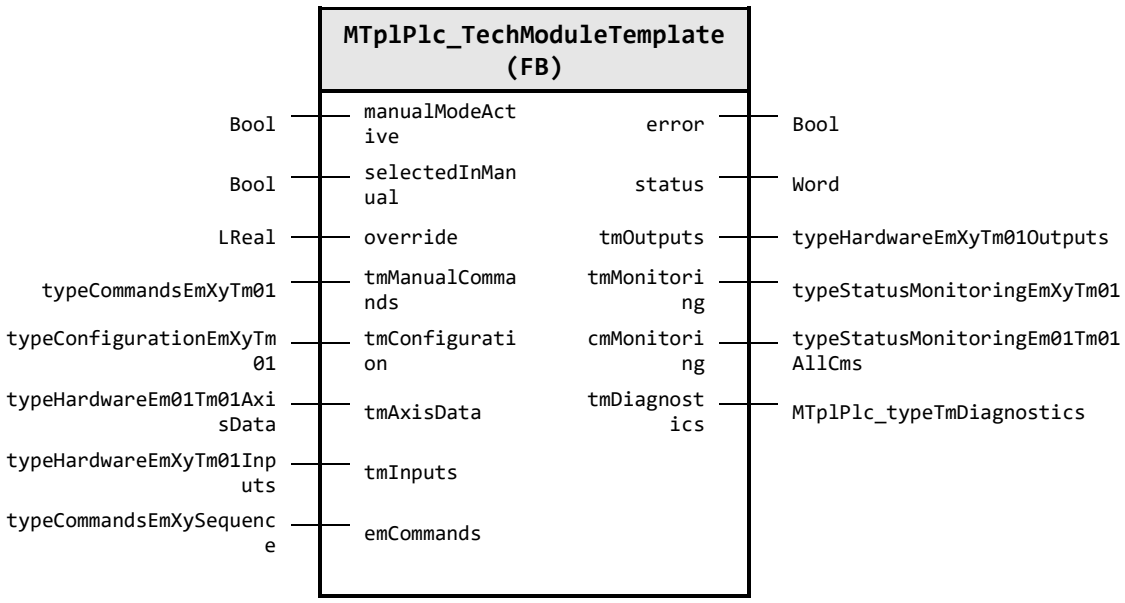
Author: APC ERLF

Short description

The FB TechModuleTemplate is a template for a Technical Module to control one or more Control Modules like axes or actors.

Interface description

Block interface



Input parameter

Identifier	Data type	Default value	Description
manualModeActive	Bool	FALSE	TRUE: Manual mode is selected
selectedInManual	Bool	FALSE	TRUE: Technical module is selected in manual mode
override	LReal	100.0	Process speed in percent
tmManualCommands	typeCommandsEmXyTm01	---	Manual commands from HMI
tmConfiguration	typeConfigurationEmXyTm01	---	Configuration equipment module xy, technical module xy
tmAxisData	typeHardwareEm01Tm01AxisData	---	Hardware unit 1, equipment module 1, technical module 1
tmInputs	typeHardwareEmXyTm01Inputs	---	Hardware equipment module xy, technical module xy
emCommands	typeCommandsEmXySequence	---	Commands of equipment module xy

Output parameter

Identifier	Data type	Description
error	Bool	TRUE: An error occurred during the execution of the FB

Identifier	Data type	Description
status	Word	16#0000 - 16#7FFF: Status of the FB, 16#8000 - 16#FFFF: Error identification
tmOutputs	typeHardwareEmXyTm01Outputs	Hardware equipment module xy, technical module xy
tmMonitoring	typeStatusMonitoringEmXyTm01	Monitoring of functional status from technical module
cmMonitoring	typeStatusMonitoringEm01Tm01AllCms	Monitoring of functional status from technical module
tmDiagnostics	MTplPlc_typeTmDiagnostics	Status of technical module

Status & Error codes

Code / Value	Identifier / Description
16#9000	ERROR_FROM_CONTROL_MODULE Error from control module 1

User defined datatype(s)

MTplPlc_typeTmDiagnostics (UDT)

Diagnostics of Technical Module provided for sequence.

Identifier	Data type	Default value	Description
error	Bool	FALSE	TRUE: Error
ID	WString[25]	---	ID for technical module
status	Word	16#0000	Status of technical module
statusSubfunction	Word	16#0000	Status or return value of called control module
IdCmFirstError	WString[25]	---	ID of control module with first error
cmDiagnostics	Array[1..MAX_NO_OF_CONTROL_MODULES] of MTplPlc_typeCmDiagnostics	---	Diagnostics of technical module

Functional description

This FB includes the region *FIRST CALL* for initialization of the function block variables and the region *CYCLIC CALL* for writing the input signals to static values. The next region is called *CONFIGURATION HANDLING*, where the configuration parameters from the HMI can be written to the static configuration in case the *saveConfiguration* is set in the interface commands. In the region *INSTANCE CONTROL MODULE* the instances of the used Control Modules are called. In the last region *OUTPUTS* the error is set or reset and the Control Module with the first error is evaluated. Furthermore, the Program_alarm is called and the static variables are copied to the outputs.

The main regions of the FB are the *COMMAND HANDLING* and *STATUS COLLECTION*. In the *COMMAND HANDLING* the commands from the sequence and the interface are written to the subordinate control modules. In *STATUS COLLECTION* the feedback of the used control modules is collected.

At the inputs of the block, the commands from the Equipment Module sequence as well as the commands from the remote interface are connected. The commands of the Equipment Module sequence are relevant for all operating modes. This is used,

for example, for enabling the Control Modules. The *manualCommands* are only relevant for the operating mode *Manual*. Via this interface the commands, like jog and pos, are provided. A boolean variable *selectedInManual* is connected to the inputs to select the Technical Module and an array of bool within *manualCommands* to select the Control Modules. This is required because in *Manual* mode it may also be necessary not to move the complete Equipment Module, but, for example, only individual Technical Modules or even individual Control Modules.

In addition to the commands, the configuration of the Technical Module is also connected to the inputs. This is pre-configured in the FB *StartupEquipmentModuleXy* and can be adapted via the HMI during runtime. The axis data and hardware inputs, needed in the FB for the associated Control Modules, are connected to the block via *tmAxisData* and *tmInputs*.

At the output, the status of the Technical Module, e.g. monitoring and diagnostics of the Technical Module and the Control Module, and outputs, e.g. outputcams, are provided.

As soon as the Technical Module requires a call in sync cylce, an interface must be provided in this FB of the Technical Module in order to forward commands to the sync cycle call of the Technical Module and receive its feedback. These inputs and outputs must be connected accordingly in the regions *COMMAND HANDLING*, *STATUS COLLECTION* as well as in the *OUTPUTS*.

Change log

Version & Date	Change description
01.00.00	**APC ERLF**
31.03.2022	First released version
01.01.00	**APC ERLF**
31.08.2022	Name of hardware inputs changed

4.1.3 MTplPlc_UnitCommandCollector (FB)

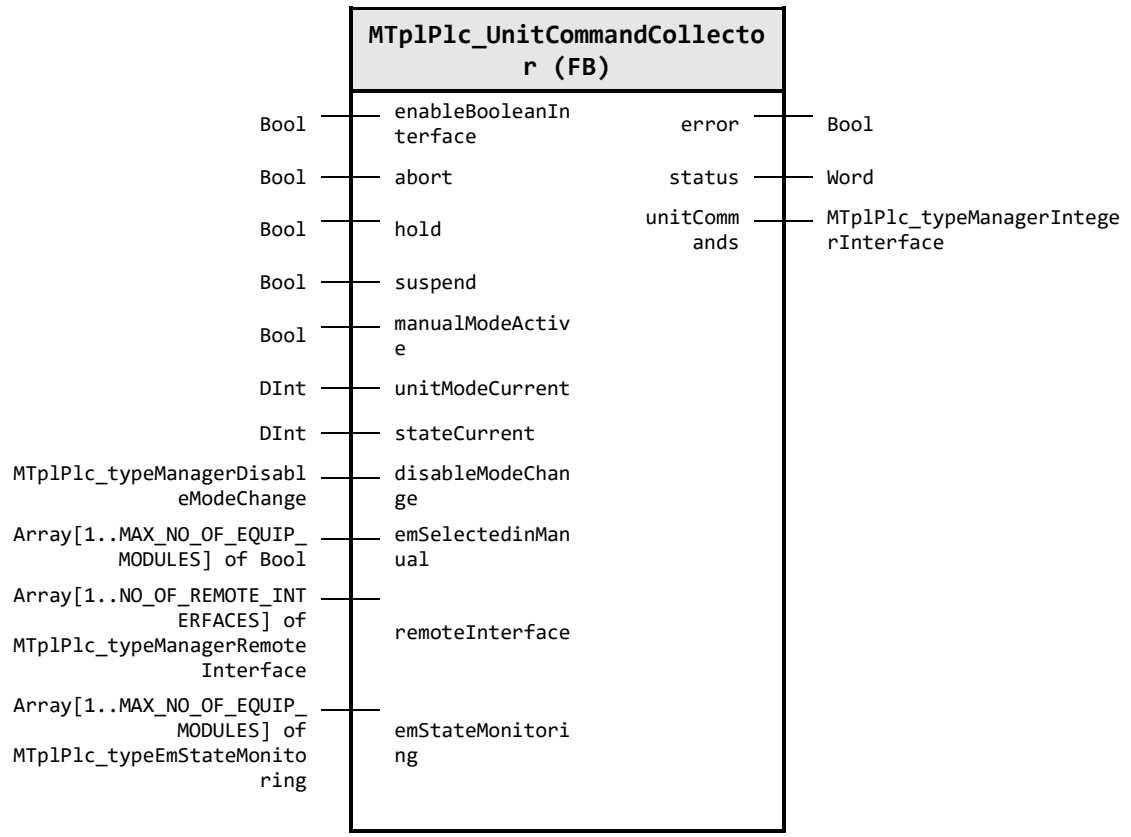
Author: APC ERLF

Short description

The FB collects commands of multiple remote interfaces and the process and provides them for the Unit Mode State Manager.

Interface description

Block interface



Input parameter

Identifier	Data type	Default value	Description
enableBooleanInterface	Bool	FALSE	TRUE: Enable boolean interface
abort	Bool	FALSE	TRUE: Command request to abort
hold	Bool	FALSE	TRUE: Command request to hold
suspend	Bool	FALSE	TRUE: Command request to suspend
manualModeActive	Bool	FALSE	TRUE: Unit mode Manual is currently active
unitModeCurrent	DInt	0	Current unit mode
stateCurrent	DInt	0	Current state of the unit mode state manager

Identifier	Data type	Default value	Description
disableModeChange	MTplPlc_typeManagerDisableModeChange	---	Defines in which state a mode change is possible, TRUE: disabled
emSelectedinManual	Array[1..MAX_NO_OF_EQUIP_MODULES] of Bool	---	TRUE: Equipment module is selected in manual mode
remoteInterface	Array[1..NO_OF_REMOTE_INTERFACES] of MTplPlc_typeManagerRemoteInterface	---	Commands for controlling the manager as Input
emStateMonitoring	Array[1..MAX_NO_OF_EQUIP_MODULES] of MTplPlc_typeEmStateMonitoring	---	Array of equipment module status

Output parameter

Identifier	Data type	Description
error	Bool	TRUE: An error occurred during the execution of the FB
status	Word	16#0000 - 16#7FFF: Status of the FB, 16#8000 - 16#FFFF: Error identification
unitCommands	MTplPlc_typeManagerIntegerInterface	Commands for controlling the manager as Input

User defined datatype(s)

MTplPlc_typeManagerDisableModeChange (UDT)

Disable mode change of manager

Identifier	Data type	Default value	Description
aborted	Bool	FALSE	FALSE: Mode change in "Aborted" is possible
stopped	Bool	FALSE	FALSE: Mode change in "Stopped" is possible
idle	Bool	FALSE	FALSE: Mode change in "Idle" is possible
suspended	Bool	FALSE	FALSE: Mode change in "Suspended" is possible
held	Bool	FALSE	FALSE: Mode change in "Held" is possible
complete	Bool	FALSE	FALSE: Mode change in "Complete" is possible

MTplPlc_typeManagerRemoteInterface (UDT)

Commands for controlling the manager as Input

Identifier	Data type	Default value	Description
integerInterface	MTplPlc_typeManagerIntegerInterface	---	Manager commands integer as interface to remote interfaces
booleanInterface	MTplPlc_typeManagerBooleanInterface	---	Manager commands boolean as interface to remote interfaces

MTplPlc_typeEmStateMonitoring (UDT)

Shows the Equipment Module status with *stateComplete*, *abort* and *hold*.

Identifier	Data type	Default value	Description
stateComplete	Bool	FALSE	TRUE: State complete
abort	Bool	FALSE	TRUE: Command abort due to error in technical module
hold	Bool	FALSE	TRUE: Command hold due to lack of meterial at input of equipment module

MTplPlc_typeManagerIntegerInterface (UDT)

Manager commands integer as interface to remote interfaces

Identifier	Data type	Default value	Description
unitMode	DInt	0	Requested unit mode if enableBooleanInterface = FALSE
unitModeChangeRequest	Bool	FALSE	TRUE: Request unit mode if enableBooleanInterface = FALSE
cntrlCmd	DInt	0	Request control command if enableBooleanInterface = FALSE
cmdChangeRequest	Bool	FALSE	TRUE: Enable change into requested state if enableBooleanInterface = FALSE
SC	Bool	FALSE	State change from FALSE to TRUE (rising edge) triggers state complete signal

Functional description

In the Machine Template the OMAC-compliant Mode State Manager LPMLV30_UnitModeStateManager from the library *LPMLV30* is used in the Unit level. In order to control the Manager with a variety of interfaces and to get the feedback of the Equipment Modules, this function block is used. It includes the region *FIRST CALL*, which initializes the FB. In the region *OUTPUTS* the static variables for the unit commands are copied to outputs and the errors are evaluated.

The main part of the function block is contained in the following regions. In the first region called *SUMMATION OF PROCESS* the feedback of all equipment modules is collected and written to the internal static state requests *stateComplete*, *abort* and *hold* with the array index 0. In the region *MAPPING OF INTERFACES* the inputs *abort*, *suspend* and *hold* are first connected to those of the process.

Subsequently, either the boolean or integer commands of the connected interfaces are written to the static mode and state requests with the respective array indexes of the interfaces. These arrays are then summed up with an appropriate prioritization. In addition, so-called change requests for mode and state changes are set, depending on the current mode and state and the user-defined configuration of the manager.

The user has to be aware that only the boolean or the integer interface is used and not a mixture of them from different remote interfaces.

Change log

Version & Date	Change description
01.00.00 31.01.2022	**APC ERLF** First released version
01.00.01 31.03.2022	**APC ERLF** Bugfix stateComplete checked for every selected EM in Manual tempSelectedModuleCounter added
01.00.02 31.03.2022	**APC ERLF** Bugfix detecting state change request, level of abort added (needed, in case abort is still active)
01.01.00 31.03.2022	**APC ERLF** Bugfix detecting mode change request after mode change was not possible (due to unrestricted state) mode change request is compared to current unit mode
01.01.01 18.10.2022	**APC ERLF** Bugfix detecting state change request via integer changed from STATE to CMD

4.1.4 MTplPlc_UnitConditionCollectorTemplate (FB)

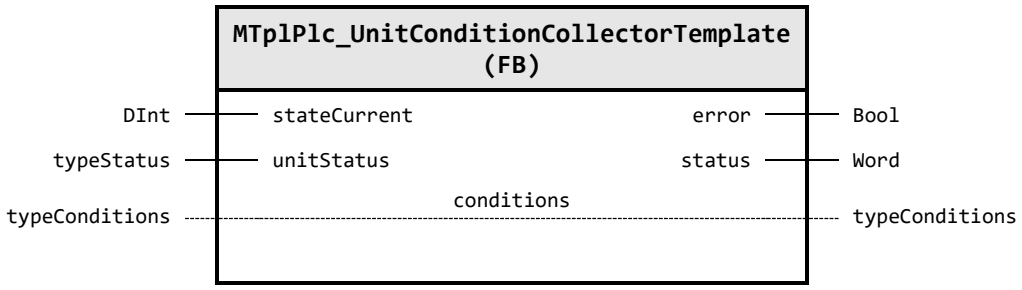
Author: APC ERLF

Short description

The FB collects the conditions of all Equipment Modules, processes them and provides unit conditions to the Equipment Modules. This FB is the interface between the Equipment Modules.

Interface description

Block interface



Input parameter

Identifier	Data type	Default value	Description
stateCurrent	DInt	0	Current state of the unit mode state manager
unitStatus	typeStatus	---	Status of whole unit

Output parameter

Identifier	Data type	Description
error	Bool	TRUE: An error occurred during the execution of the FB
status	Word	16#0000 - 16#7FFF: Status of the FB, 16#8000 - 16#FFFF: Error identification

In/Out parameter

Identifier	Data type	Description
conditions	typeConditions	Conditions

Functional description

The template FB represents the interface between the Equipment Modules and the Unit. It has to be adapted by the user in the Unit layer. All conditions of the Equipment Modules are collected in the FB, processed and distributed to the Equipment Modules. This is necessary if Equipment Modules require conditions from other Equipment Modules. This communication is not implemented inbetween the Equipment Modules. It is implemented as a standard interface between the Unit level and each Equipment Module. This offers the possibility to process all Equipment Module conditions which are exchanged in one FB.

This simplifies the condition interface via UDTs. One example is the exchange of release conditions from one Equipment Module to the other. Therefore, the Equipment Modules sends a Module *finished* condition to the Unit level. In this FB the release conditions from all Equipment Modules is evaluated. Afterwards the *release* condition from unit level is send back to the Equipment Module.

Another example is the use of Equipment Modules with SIMATIC Interpreter. The interface between the Equipment Modules and the Interpreter Control is carried out via the Unit Condition Collector FB. Depending on the current state, the Equipment Modules send a start condition to the Unit Condition Collector FB. This forwards the condition to the Interpreter Control.

Change log

Version & Date	Change description
01.00.01 22.12.2022	**APC ERLF** First released version

5 PLC data types

5.1 Commands

5.1.1 MTplPlc_typeManagerBooleanInterface (UDT)

Description

Manager commands boolean as interface to remote interfaces

Parameter description

Identifier	Data type	Default value	Description
modeRequest	MTplPlc_typeManagerModeBoolean	---	Commands from remote interfaces requesting mode change
stateRequest	MTplPlc_typeManagerStateBoolean	---	Commands from remote interfaces for state change

5.1.2 MTplPlc_typeManagerCommands (UDT)

Description

Commands for controlling the manager from remote interfaces

Parameter description

Identifier	Data type	Default value	Description
remoteInterface	Array[1..NO_OF_REMOTE_INTERFACES] of MTplPlc_typeManagerRemoteInterface	---	Commands for controlling the manager as Input

5.1.3 MTplPlc_typeManagerIntegerInterface (UDT)

Description

Manager commands integer as interface to remote interfaces

Parameter description

Identifier	Data type	Default value	Description
unitMode	DInt	0	Requested unit mode if enableBooleanInterface = FALSE
unitModeChangeRequest	Bool	FALSE	TRUE: Request unit mode if enableBooleanInterface = FALSE
cntrlCmd	DInt	0	Request control command if enableBooleanInterface = FALSE
cmdChangeRequest	Bool	FALSE	TRUE: Enable change into requested state if enableBooleanInterface = FALSE
SC	Bool	FALSE	State change from FALSE to TRUE (rising edge) triggers state complete signal

5.1.4 MTplPlc_typeManagerModeBoolean (UDT)

Description

Commands from remote interfaces requesting mode change

Parameter description

Identifier	Data type	Default value	Description
productionModeRequest	Bool	false	TRUE: Request change to unit mode Production if enableBooleanInterface = TRUE
maintenanceModeRequest	Bool	false	TRUE: Request change to unit mode Maintenance if enableBooleanInterface = TRUE
manualModeRequest	Bool	false	TRUE: Request change to unit mode Manual if enableBooleanInterface = TRUE
userMode01Request	Bool	false	TRUE: Request change to user-defined unit mode 01 if enableBooleanInterface = TRUE
userMode02Request	Bool	false	TRUE: Request change to user-defined unit mode 02 if enableBooleanInterface = TRUE
userMode03Request	Bool	false	TRUE: Request change to user-defined unit mode 03 if enableBooleanInterface = TRUE
userMode04Request	Bool	false	TRUE: Request change to user-defined unit mode 04 if enableBooleanInterface = TRUE
userMode05Request	Bool	false	TRUE: Request change to user-defined unit mode 05 if enableBooleanInterface = TRUE
userMode06Request	Bool	false	TRUE: Request change to user-defined unit mode 06 if enableBooleanInterface = TRUE
userMode07Request	Bool	false	TRUE: Request change to user-defined unit mode 07 if enableBooleanInterface = TRUE
userMode08Request	Bool	false	TRUE: Request change to user-defined unit mode 08 if enableBooleanInterface = TRUE

5.1.5 MTplPlc_typeManagerRemoteInterface (UDT)

Description

Commands for controlling the manager as Input

Parameter description

Identifier	Data type	Default value	Description
integerInterface	MTplPlc_typeManagerIntegerInterface	---	Manager commands integer as interface to remote interfaces
booleanInterface	MTplPlc_typeManagerBooleanInterface	---	Manager commands boolean as interface to remote interfaces

5.1.6 MTplPlc_typeManagerStateBoolean (UDT)

Description

Commands from remote interfaces for state change

Parameter description

Identifier	Data type	Default value	Description
resetCmdRequest	Bool	FALSE	TRUE: Request control command Reset if enableBooleanInterface = TRUE
startCmdRequest	Bool	FALSE	TRUE: Request control command Start if enableBooleanInterface = TRUE
stopCmdRequest	Bool	FALSE	TRUE: Request control command Stop if enableBooleanInterface = TRUE
holdCmdRequest	Bool	FALSE	TRUE: Request control command Hold if enableBooleanInterface = TRUE
unholdCmdRequest	Bool	FALSE	TRUE: Request control command Unhold if enableBooleanInterface = TRUE
suspendCmdRequest	Bool	FALSE	TRUE: Request control command Suspend if enableBooleanInterface = TRUE
unsuspendCmdRequest	Bool	FALSE	TRUE: Request control command Unsuspend if enableBooleanInterface = TRUE
abortCmdRequest	Bool	FALSE	TRUE: Request control command Abort if enableBooleanInterface = TRUE
clearCmdRequest	Bool	FALSE	TRUE: Request control command Clear if enableBooleanInterface = TRUE
completeCmdRequest	Bool	FALSE	TRUE: Request control command Complete if enableBooleanInterface = TRUE

5.2 Configuration

5.2.1 MTplPlc_typeManagerConfiguration (UDT)

Description

Configuration of the manager as Input

Parameter description

Identifier	Data type	Default value	Description
enableBooleanInterface	Bool	false	TRUE: Enable boolean interface
disableModeChange	MTplPlc_typeManagerDisableModeChange	---	Defines in which state a mode change is possible, TRUE: disabled
disableModesStates	typeLPMLV30_Configuration	---	

5.2.2 MTplPlc_typeManagerDisableModeChange (UDT)

Description

Disable mode change of manager

Parameter description

Identifier	Data type	Default value	Description
aborted	Bool	FALSE	FALSE: Mode change in "Aborted" is possible
stopped	Bool	FALSE	FALSE: Mode change in "Stopped" is possible
idle	Bool	FALSE	FALSE: Mode change in "Idle" is possible
suspended	Bool	FALSE	FALSE: Mode change in "Suspended" is possible
held	Bool	FALSE	FALSE: Mode change in "Held" is possible
complete	Bool	FALSE	FALSE: Mode change in "Complete" is possible

5.3 Status

5.3.1 MTplPlc_typeCmDiagnostics (UDT)

Description

Diagnostics of Control Module provided for sequence.

Parameter description

Identifier	Data type	Default value	Description
error	Bool	FALSE	TRUE: Error
ID	WString[25]	---	ID for technical module
status	Word	16#0000	Status of technical module
statusSubfunction	Word	16#0000	Status or return value of called control module

5.3.2 MTplPlc_typeEmDiagnostics (UDT)

Description

Equipment Module diagnostics

Parameter description

Identifier	Data type	Default value	Description
error	Bool	FALSE	TRUE: Error
ID	WString[25]	---	ID of equipment module
status	Word	16#0000	Status of technical module
errorState	DInt	0	State in where the error occurred
errorSubstate	DInt	0	Substate of the main state where the error occurred
IdTmFirstError	WString[25]	---	ID of control module with first error
tmDiagnostics	Array[1..MAX_NO_OF_TECH_MODULES] of MTplPlc_typeTmDiagnostics	---	Diagnostics of technical module

5.3.3 MTplPlc_typeEmStateMonitoring (UDT)

Description

Shows the Equipment Module status with *stateComplete*, *abort* and *hold*.

Parameter description

Identifier	Data type	Default value	Description
stateComplete	Bool	FALSE	TRUE: State complete
abort	Bool	FALSE	TRUE: Command abort due to error in technical module
hold	Bool	FALSE	TRUE: Command hold due to lack of meterial at input of equipment module

5.3.4 MTplPlc_typeManagerMonitoring (UDT)

Description

Monitoring of manager mode and state provided for MES and Equipment Modules as output.

Parameter description

Identifier	Data type	Default value	Description
unitModeCurrent	DInt	0	Current unit mode
stateCurrent	DInt	0	Current state of the unit mode state manager
productionModeActive	Bool	FALSE	TRUE: Unit mode Production is currently active
maintenanceModeActive	Bool	FALSE	TRUE: Unit mode Maintenance is currently active
manualModeActive	Bool	FALSE	TRUE: Unit mode Manual is currently active
userMode01Active	Bool	FALSE	TRUE: User-defined unit mode 01 is currently active
userMode02Active	Bool	FALSE	TRUE: User-defined unit mode 02 is currently active
userMode03Active	Bool	FALSE	TRUE: User-defined unit mode 03 is currently active
userMode04Active	Bool	FALSE	TRUE: User-defined unit mode 04 is currently active
userMode05Active	Bool	FALSE	TRUE: User-defined unit mode 05 is currently active
userMode06Active	Bool	FALSE	TRUE: User-defined unit mode 06 is currently active
userMode07Active	Bool	FALSE	TRUE: User-defined unit mode 07 is currently active
userMode08Active	Bool	FALSE	TRUE: User-defined unit mode 08 is currently active
clearingStateActive	Bool	FALSE	TRUE: State Clearing is currently active
stoppedStateActive	Bool	FALSE	TRUE: State Stopped is currently active
startingStateActive	Bool	FALSE	TRUE: State Starting is currently active
idleStateActive	Bool	FALSE	TRUE: State Idle is currently active
suspendedStateActive	Bool	FALSE	TRUE: State Suspended is currently active
executeStateActive	Bool	FALSE	TRUE: State Execute is currently active
stoppingStateActive	Bool	FALSE	TRUE: State Stopping is currently active
abortingStateActive	Bool	FALSE	TRUE: State Aborting is currently active
abortedStateActive	Bool	FALSE	TRUE: State Aborted is currently active
holdingStateActive	Bool	FALSE	TRUE: State Holding is currently active
heldStateActive	Bool	FALSE	TRUE: State Held is currently active
unholdingStateActive	Bool	FALSE	TRUE: State Unholding is currently active
suspendingStateActive	Bool	FALSE	TRUE: State Suspending is currently active
unsuspendingStateActive	Bool	FALSE	TRUE: State Unsuspending is currently active
resettingStateActive	Bool	FALSE	TRUE: State Resetting is currently active
completingStateActive	Bool	FALSE	TRUE: State Completing is currently active

Identifier	Data type	Default value	Description
completeStateActive	Bool	FALSE	TRUE: State Complete is currently active
StatesDisabled	DInt	0	Disabled states in current unit mode

5.3.5 MTplPlc_typeManagerStatus (UDT)

Description

Shows the Manager status.

Parameter description

Identifier	Data type	Default value	Description
monitoring	MTplPlc_typeManagerMonitoring	---	Monitoring of manager mode and state provided for MES and Equipment Modules as output.
diagnostics	typeLPMLV30_Diagnostics	---	

5.3.6 MTplPlc_typeTmDiagnostics (UDT)

Description

Diagnostics of Technical Module provided for sequence.

Parameter description

Identifier	Data type	Default value	Description
error	Bool	FALSE	TRUE: Error
ID	WString[25]	---	ID for technical module
status	Word	16#0000	Status of technical module
statusSubfunction	Word	16#0000	Status or return value of called control module
IdCmFirstError	WString[25]	---	ID of control module with first error
cmDiagnostics	Array[1..MAX_NO_OF_CONTROL_MODULES] of MTplPlc_typeCmDiagnostics	---	Diagnostics of technical module

6 PLC tags & constants

6.1.1 MTplPlc_Constants

Constant identifier, values and description

Identifier & Value	Description
MAX_NO_OF_CONTROL_MODULES 3	Maximum number of control modules within one technical module
MAX_NO_OF_EQUIP_MODULES 1	Maximum number of equipment modules within one unit
MAX_NO_OF_TECH_MODULES 3	Maximum number of technical modules within one equipment module
NO_OF_REMOTE_INTERFACES 2	Number of interfaces used in the project

7 Appendix

7.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.
The Industry Online Support is the central address for information about our products, solutions and services.
Product information, manuals, downloads, FAQs, application examples and videos, all information is accessible with just a few mouse clicks:
https://support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers - ranging from basic support to individual support contracts. Please send queries to Technical Support via Web form:
https://www.siemens.com/industry/supportrequest

SITRAIN - Training for Industry

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.
For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:
https://www.siemens.com/sitrain

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contract's

You can find detailed information on our range of services in the service catalog web page:
https://support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the “Siemens Industry Online Support” app. The app is available for Apple iOS and Android:
https://support.industry.siemens.com/cs/ww/en/sc/2067

7.2 Links and Literature

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the entry page of the application example https://support.industry.siemens.com/cs/ww/en/view/109804620
\3\	Programming Guidelines and Programming Style guide for SIMATIC S7-1200 and S7-1500 https://support.industry.siemens.com/cs/ww/en/view/81318674
\4\	Library with PLC data types (LPD) for STEP 7 (TIA Portal) and SIMATIC S7-1200 / S7-1500 https://support.industry.siemens.com/cs/ww/en/view/109482396
\5\	Guideline on Library Handling in Tia Portal https://support.industry.siemens.com/cs/ww/en/view/109747503
\6\	Libraries in the TIA Portal https://support.industry.siemens.com/cs/ww/en/view/109738702
\7\	Engineering and automation of batch processes with PCS 7 along ISA-88 models https://support.industry.siemens.com/cs/de/de/view/109784331
\8\	SIMATIC/SIMOTION OMAC PackML V3 Machine and Unit States https://support.industry.siemens.com/cs/document/49970441
\9\	OMAC PackML https://www.omac.org/packml
\10\	SIMATIC Alarm Handling for SINAMICS drives https://support.industry.siemens.com/cs/document/109761931

7.3 FAQ / Frequently Asked Questions

7.4 Change log

Version & Date	Change description
V1.1.1 03/2023	**NEW:** MTplPlc_SequenceTemplate / V01.00.02 MTplPlc_TechModuleTemplate / V01.01.00 MTplPlc_UnitCommandCollector / V01.01.01 MTplPlc_UnitConditionCollectorTemplate / V01.00.01