

# BONFIGLIOLI AXIS MANAGER

TwinCAT

Library guide







## TABLE OF CONTENTS

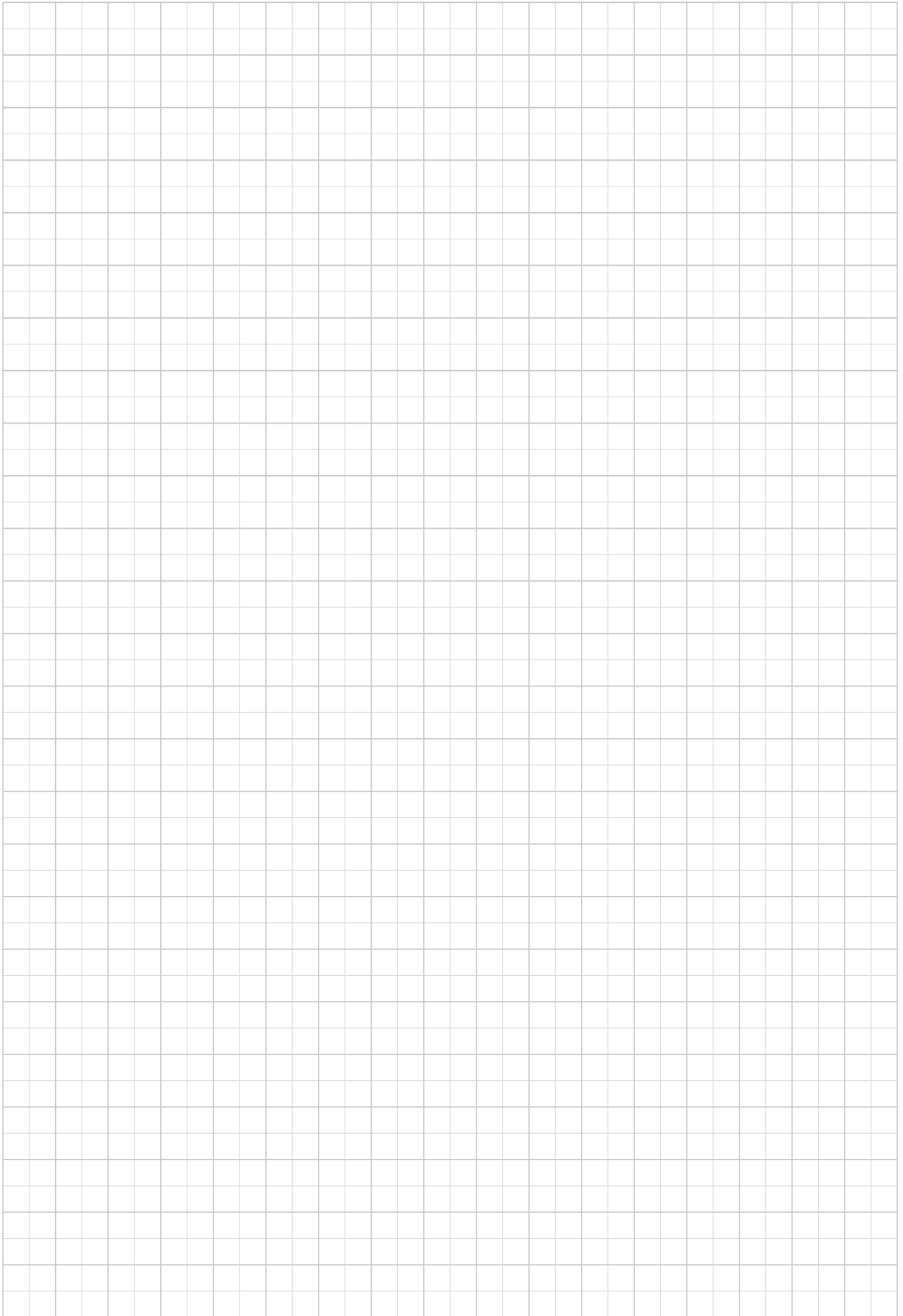
<b>1</b>	<b>General information about the Documentation .....</b>	<b>7</b>
<b>1.1</b>	<b>This document .....</b>	<b>7</b>
<b>1.2</b>	<b>Prerequisites .....</b>	<b>7</b>
<b>1.3</b>	<b>Warranty and liability .....</b>	<b>7</b>
<b>1.4</b>	<b>Copyright .....</b>	<b>8</b>
<b>1.5</b>	<b>Storage .....</b>	<b>8</b>
<b>2</b>	<b>General safety instructions and information on use .....</b>	<b>9</b>
<b>2.1</b>	<b>Warning information and symbols used in the user manual .....</b>	<b>9</b>
2.1.1	Hazard classes .....	9
2.1.2	Hazard symbols .....	9
2.1.3	Prohibition signs .....	9
2.1.4	Personal safety equipment .....	10
2.1.5	Recycling .....	10
2.1.6	Grounding symbol .....	10
2.1.7	ESD symbol .....	10
2.1.8	Information sign .....	10
<b>3</b>	<b>Introduction .....</b>	<b>11</b>
<b>3.1</b>	<b>Information about the product .....</b>	<b>11</b>
<b>4</b>	<b>Library Overview .....</b>	<b>12</b>
<b>4.1</b>	<b>Example Project .....</b>	<b>12</b>
<b>4.2</b>	<b>Overview of the POUs .....</b>	<b>13</b>
<b>4.3</b>	<b>Overview of the Enumerations .....</b>	<b>14</b>
<b>4.4</b>	<b>Overview of the Structures .....</b>	<b>15</b>
<b>4.5</b>	<b>Overview of the Alias .....</b>	<b>16</b>
<b>4.6</b>	<b>Overview of the Unions .....</b>	<b>16</b>
<b>4.7</b>	<b>Overview of the Dependencies .....</b>	<b>16</b>
<b>5</b>	<b>Overview of Axis Manager .....</b>	<b>17</b>
<b>5.1</b>	<b>State diagram .....</b>	<b>18</b>
<b>5.2</b>	<b>AxisMgr base model .....</b>	<b>19</b>
5.2.1	General architecture .....	19
5.2.2	Base interface .....	20
5.2.2.1	Methods and properties .....	21
5.2.3	Common behavior .....	21
5.2.3.1	From Unknown to Disabled .....	21
5.2.3.2	Disabled, Standstill and EmergencyStop .....	22
5.2.3.3	Launching commands: <i>i_iCmd</i> , <i>i_xLaunchCmd</i> , <i>q_xCmdDone</i> and <i>q_xCmdBusy</i> .....	24
5.2.3.4	ErrorStop: possible causes, <i>q_xFault</i> and <i>i_xReset</i> .....	25
5.2.4	Fieldbus management .....	27
5.2.4.1	CANOpen/EtherCAT AxisMgr .....	28
<b>5.3</b>	<b>Overview of AxisMgr Functions .....</b>	<b>30</b>
5.3.1	Input/Output data .....	32
5.3.1.1	<i>i_stData</i> structure .....	32
5.3.1.2	Values conversion factor, user units definition .....	32



5.3.2	AxisMgr_SM supported functions .....	34
5.3.2.1	Stop function (AxisMgr_SM) .....	36
5.3.2.2	Halt SuperImposed function (AxisMgr_SM) .....	36
5.3.2.3	SetPos function (AxisMgr_SM) .....	37
5.3.2.4	Velocity function (AxisMgr_SM) .....	38
5.3.2.5	Positioning function (AxisMgr_SM) .....	39
5.3.2.6	Gearing function (AxisMgr_SM) .....	40
5.3.2.7	Camming function (AxisMgr_SM) .....	41
5.3.2.1	SuperImposed function (AxisMgr_SM) .....	42
5.3.2.2	Jog function (AxisMgr_SM) .....	43
5.3.2.3	Set Controller Mode function (AxisMgr_SM) .....	44
5.3.2.4	Set Torque function (AxisMgr_SM) .....	45
5.3.3	AxisMgr "Light" supported functions .....	46
5.3.3.1	Stop function (AxisMgr "Light") .....	47
5.3.3.2	Homing function (AxisMgr "Light") .....	49
5.3.3.3	Profile Velocity function (AxisMgr "Light") .....	50
5.3.3.4	Velocity Mode function (AxisMgr "Light") .....	51
5.3.3.5	Profile Position function (AxisMgr "Light") .....	52
5.3.3.6	Profile Torque function (AxisMgr "Light") .....	54
5.3.4	Common functions .....	55
5.3.4.1	TouchProbe function (AxisMgr SM and "Light") .....	55
<b>6</b>	<b>Overview of Axis Manager Groups .....</b>	<b>57</b>
<b>7</b>	<b>Program Organization Units (POU) .....</b>	<b>59</b>
<b>7.1</b>	<b>Function Blocks .....</b>	<b>60</b>
7.1.1	AxisMgr_SM .....	61
7.1.2	AxisMgr_ANG_CAN_ETC .....	64
7.1.3	AxisMgr_AGL_CAN_ETC .....	68
7.1.4	AxisMgr_ACUx10_CAN_ETC .....	71
7.1.5	AxisMgr_AXV_CAN_ETC .....	75
7.1.6	AxisMgr_AXM_CAN_ETC .....	79
7.1.7	AxisMgr_Group .....	83
7.1.8	FB_GenericAcces_CAN_ETC .....	86
<b>7.2</b>	<b>Functions .....</b>	<b>88</b>
7.2.1	GetAppWarningMessage .....	89
7.2.2	GetFaultMessage_ACTIVE_AGILE .....	89
7.2.3	GetFaultMessage_AXIA .....	90
7.2.4	GetWarningMessage .....	90
7.2.5	GetWarningDiagnostic_Axia .....	91
<b>8</b>	<b>Data Unit Types .....</b>	<b>92</b>
<b>8.1</b>	<b>Enumerations .....</b>	<b>93</b>
8.1.1	AM_AxisMgrStatusEnum .....	94
8.1.2	AM_ChangeProfileBehaviorEnum .....	95
8.1.3	AM_ControllerMode .....	95
8.1.4	AM_DataTypeEnum .....	96
8.1.5	AM_DriveModelEnum .....	96
8.1.6	AM_EndOfPositionBehaviorEnum .....	97
8.1.7	AM_ErrorSourceEnum .....	97
8.1.8	AM_FBErrorEnum .....	98
8.1.9	AM_FieldbusEnum .....	98
8.1.10	AM_HomingMethodsEnum .....	99
8.1.11	AM_NewCommandFailedBehaviorEnum .....	101
8.1.12	AM_PowerOffOptionEnum .....	102
8.1.13	AM_PosTypeEnum .....	103
8.1.14	AM_TouchProbeModeEnum .....	103
8.1.15	AM_UserErrorEnum .....	104
8.1.16	EN_State_CAN_ETC .....	105



<b>8.2</b>	<b>Structures .....</b>	<b>106</b>
8.2.1	AM_CamDataType.....	107
8.2.2	AM_CommandDataType_ACTIVE.....	108
8.2.3	AM_CommandDataType_AGL.....	109
8.2.4	AM_CommandDataType_Axia.....	109
8.2.5	AM_CommandDataType_SM .....	110
8.2.6	AM_ErrorDiagnosticType.....	111
8.2.7	AM_GearDataType .....	111
8.2.8	AM_HomingDataType .....	112
8.2.9	AM_InitDataType .....	113
8.2.10	AM_InitParameterType .....	114
8.2.11	AM_JogDataType .....	114
8.2.12	AM_PositioningDataType .....	115
8.2.13	AM_PowerDataType .....	116
8.2.14	AM_ProfilePositionDataType_ACTIVE .....	117
8.2.15	AM_ProfilePositionDataType_Axia.....	120
8.2.16	AM_ProfileTorqueDataType_Axia.....	122
8.2.17	AM_ProfileVelocityDataType_ACTIVE .....	123
8.2.18	AM_ProfileVelocityDataType_Axia.....	124
8.2.19	AM_QuickStopDataType.....	124
8.2.20	AM_SetPosDataType .....	125
8.2.21	AM_SetTorqueDataType .....	125
8.2.22	AM_StopDataType.....	126
8.2.23	AM_SuperImposedDataType .....	127
8.2.24	AM_TouchProbeDataType.....	128
8.2.25	AM_TouchProbeStatusType.....	128
8.2.26	AM_TPChannelStatusType .....	129
8.2.27	AM_TPChannelType.....	130
8.2.28	AM_VelocityModeDataType_ACTIVE .....	131
8.2.29	AM_VelocityModeDataType_AGL .....	132
8.2.30	AM_VelocityModeDataType_Axia .....	133
8.2.31	AM_VeloDataType_SM.....	133
8.2.32	AM_WarningDiagnosticType_ACTIVE .....	134
8.2.33	AM_WarningDiagnosticType_Axia.....	135
8.2.34	AM_WarningGroupType_Axia .....	136
8.2.35	ST_ParamType_Common.....	137
<b>9</b>	<b>Global Variables .....</b>	<b>139</b>
<b>9.1</b>	<b>GVL .....</b>	<b>140</b>
<b>10</b>	<b>Parameters .....</b>	<b>141</b>
<b>10.1</b>	<b>_AM_K.....</b>	<b>142</b>
<b>11</b>	<b>Notes .....</b>	<b>143</b>
<b>12</b>	<b>Revision Index (R) .....</b>	<b>144</b>





# 1 General information about the Documentation

For better clarity, the documentation of the library is structured according to the customer-specific requirements.

This documentation was written in English language. The English documentation is the original one. Other language versions are translated.

## 1.1 This document

The user manual contains important information about the implementation of the *Bonfiglioli Axis Manager* Library and use of its functionalities. Compliance with this user manual contributes to avoiding risks, minimizing repair cost and downtimes.

For this reason, make sure to read the user manual carefully.



### WARNING

Bonfiglioli S.p.A. shall not be held liable for any damage caused by any non-compliance with the documentation.



In case any problems occur which are not covered by the documentation sufficiently, please contact the manufacturer.

## 1.2 Prerequisites

In order to fully understand the content of this document, the reader must be familiar with the following topics:

- TwinCAT and IEC 61131-3 programming languages;
- Bonfiglioli's inverter series;
- Fieldbus technologies;
- Object Oriented Programming;
- DSP CiA402 specifications and motion kinematics.

## 1.3 Warranty and liability

Bonfiglioli S.p.A. would like to point out that the contents of this user manual do not form part of any previous or existing agreement, assurance or legal relationship. Neither are they intended to supplement or replace such agreements, assurances or legal relationships. Any obligations of the manufacturer shall solely be based on the relevant purchase agreement which also includes the complete and solely valid warranty stipulations. These contractual warranty provisions are neither extended nor limited by the specifications contained in this documentation.

The manufacturer reserves the right to correct or amend the specifications, product information and omissions in these operating instructions without notice. The manufacturer shall not be liable for any damage, injuries or costs which may be caused by the aforementioned reasons.

In addition to that, Bonfiglioli S.p.A. excludes any warranty/liability claims for any personal and/or material damage if such damage is due to one or more of the following causes:

- inappropriate use of the library,
- non-compliance with the instructions, warnings and prohibitions contained in the documentation,
- unauthorized modifications of library,
- insufficient monitoring of parts of the machine/plant which are subject to wear,
- repair work at the machine/plant not carried out properly or in time,
- catastrophes by external impact and Force Majeure.



---

## 1.4 Copyright

In accordance with applicable law against unfair competition, this user manual is a certificate. Any copyrights relating to it shall remain with

### **Bonfiglioli S.p.A.**

Via Cav. Clementino Bonfiglioli 1 - 40012 Calderara di Reno (BO)

Tel. +39 (0)51 6473111

This user manual is intended for the user. Any disclosure or copying of this document, exploitation, and communication of its contents (as hardcopy or electronically) shall be forbidden, unless permitted expressly.

Any non-compliance will constitute an offense against the copyright law dated 09 September 1965, the law against unfair competition and the Civil Code and may result in claims for damages. All rights relating to patent, utility model or design registration reserved.

## 1.5 Storage

The documentation forms an integral part of the library.



## 2 General safety instructions and information on use

The chapter "General safety instructions and information on use" contains general safety instructions for the Operator and the Operating Staff. At the beginning of certain main chapters, some safety instructions are included which apply to all work described in the relevant chapter. Special work-specific safety instructions are provided before each safety-relevant work step.

### 2.1 Warning information and symbols used in the user manual

#### 2.1.1 Hazard classes

The following hazard identifications and symbols are used to mark particularly important information:



##### **DANGER**

Identification of immediate threat holding a **high** risk of death or serious injury if not avoided.



##### **WARNING**

Identification of immediate threat holding a **medium** risk of death or serious injury if not avoided.







##### **CAUTION**

Identification of immediate threat holding a **low** risk of minor or moderate physical injury if not avoided.


##### **NOTE**

Identification of a threat holding a risk of material damage if not avoided.

#### 2.1.2 Hazard symbols


Symbol	Meaning	Symbol	Meaning
	General hazard		Suspended load
	Electrical voltage		Hot surfaces

#### 2.1.3 Prohibition signs


Symbol	Meaning
	No switching; it is forbidden to switch the machine/plant, assembly on




### 2.1.4 Personal safety equipment

Symbol	Meaning
	Wear body protection


### 2.1.5 Recycling

Symbol	Meaning
	Recycling, to avoid waste, collect all materials for reuse


### 2.1.6 Grounding symbol

Symbol	Meaning
	Ground connection

### 2.1.7 ESD symbol

Symbol	Meaning
	ESD: Electrostatic Discharge (can damage components and assemblies)

### 2.1.8 Information sign

Symbol	Meaning
	Tips and information making using the library easier.



### 3 Introduction

The library provides function and function blocks as tools to manage and control the Bonfiglioli's inverter and drive series.

#### 3.1 Information about the product

Before to attempt a solution (machine or process) for a specific application using the POU in the library, it is required to consider the usage of right procedure which includes, besides the rest, risks analysis, functional safety, components compatibility, test and validation of the system related to this library.



#### **WARNING**

##### **WRONG USAGE OF POUs**

- ➔ Do a safety analysis for the application and for the installed devices.
- ➔ Verify that the POUs are compatible with the devices in the system and that they do not have unwanted behavior on the system work.
- ➔ Use appropriate parameters, limit the values in particular, and observe the usage of the machine and the stop behavior.
- ➔ Verify that all the sensor and the actuators are compatible with the selected POU.
- ➔ Test deeply all the functions during the validation and the commissioning in each operating mode.
- ➔ Provide independent methods for the critical control functions (emergency stop, etc...) after an analysis about safety and the related rules.

**The inobservance of these instructions can lead to death, injuries or damage to the equipment.**



#### **WARNING**

##### **UNEXPECTED OPERATION OF THE EQUIPMENT**

Update the application software according to the necessities, paying attention mostly to the modification of the I/O addresses every time that the hardware configuration changes. Pay also attention performing online changes in the software, variable address areas and pointers in use may have unexpected consequences.



## 4 Library Overview

The library provides functions and function blocks as tools to manage and control the Bonfiglioli's inverter and drive series. This library has been implemented with **TwinCAT TcXaeShell version 15.0.28307.1300 D15.9**. Compatibility with other development system or TcXaeShell versions is not guaranteed.

Features	Value
Library title	Bonfiglioli Axis Manager
Company	Bonfiglioli
Author	Motion Control Development
Category	Application
Default Name Space	B_AxisMgr

### 4.1 Example Project

The specific application requirements may be different from those assumed for any related examples, templates or architectures described herein. In that case, it is needed to adapt the information provided in this and other related documents to the particular needs. To do so, the user need to consult the specific product documentation of the hardware and/or software components that you may add or substitute for any information specified in this documentation. Pay particular attention and conform to any safety information, different electrical requirements and normative standards that would apply to the user's adaptation.



#### **WARNING**

##### **REGULATORY INCOMPATIBILITY**

Be sure that all equipment applied and systems designed comply with all application local, regional and national regulations and standards.  
Failure to follow these instructions can result in death, serious injury, or equipment damage.



## 4.2 Overview of the POU

The library contains the following function block and functions:

Type of POU	Fieldbus/Function	Drive	Function Block / Function	Description
Motion AxisMgr	CANOpen EtherCAT	ANG ACUx10 AXV AXM iBMD	<a href="#">AxisMgr_SM</a>	This FB offers an easy and intuitive interface to manage the control of an axis in CSx mode (refer to the CiA DS402 specifications for further information). The FB uses the motion blocks provided by the Tc2_MC2, putting them together and managing the right sequence of commands. Refer to Beckhoff's documentations for further information. The specific drive diagnostic provided by the FB help the integration of Bonfiglioli's drive in the user program.
Drive and fieldbus dependent AxisMgr "Light" (NC axis not used)	CANOpen EtherCAT	ANG	<a href="#">AxisMgr_ANG_CAN_ETC</a>	These FBs offer an easy and intuitive interface to manage the control of a specific inverter series from Bonfiglioli. Each FB is designed to work in combination with a specific fieldbus protocol (CANOpen, EtherCAT, PROFINet...). Unlike AxisMgr_SM, this FB does not work with CSx modes and TwinCAT NC, instead, it gives the possibility to easily manage the other operation modes (where supported) such as Profile Position Mode, Homing Mode, Profile Velocity Mode...
		AGL	<a href="#">AxisMgr_AGL_CAN_ETC</a>	
		ACUx10	<a href="#">AxisMgr_ACUx10_CAN_ETC</a>	
		AXV	<a href="#">AxisMgr_AXV_CAN_ETC</a>	
		AXM	<a href="#">AxisMgr_AXM_CAN_ETC</a>	
Group	<a href="#">AxisMgr_Group</a>			This FB is very useful when it comes to control several axes together. The user can configure the FB adding as many AxisMgrs as he/she wants, so then the commands given to the AxisMgr_Group are spread to each AxisMgr at the same time. The AxisMgr_Group collect the diagnostic of all the AxisMgrs in a single point.
Utility	CANOpen EtherCAT	<a href="#">FB_GenericAcces_CAN_ETC</a>		This FB is designed to give an easy way to access objects and parameter via SDO. The user can configure it adding the parameter that has to be exchanged, then the FB will manage the SDOs automatically, putting them in a queue. The user has not to think about the right sequence to send/receive several SDO.



	ANG ACUx10 AGL	<a href="#">GetAppWarningMessage</a>	It returns the Application Warning message according to the specified App Warning ID (only for Active/Agile inverter series).
		<a href="#">GetFaultMessage ACTIVE AGILE</a>	It returns the fault message according to the specified Fault ID (only for Active/Agile inverter series).
		<a href="#">GetWarningMessage</a>	It returns the Warning message according to the specified Warning ID (only for Active/Agile inverter series).
	AXV AXM	<a href="#">GetFaultMessage AXIA</a>	It returns the fault message according to the specified Fault ID (only for Axia inverter series).
		<a href="#">GetWarningDiagnostic Axia</a>	It returns the warnings message (divided into groups) according to the Warnings IDs (only for Axia inverter series).

### 4.3 Overview of the Enumerations

Enum	Description
<a href="#">AM_AxisMgrStatusEnum</a>	It contains the all the possible states that an axis can reach
<a href="#">AM_ChangeProfileBehaviorEnum</a>	It defines the behavior when a new setpoint is triggered in Profile Position Mode.
<a href="#">AM_ControllerMode</a>	It contains the allowed Mode of Operation (AxisMgr_SM)
<a href="#">AM_DataTypeEnum</a>	It defines the available data types
<a href="#">AM_DriveModelEnum</a>	It contains the drive model
<a href="#">AM_EndOfPositionBehaviorEnum</a>	It defines the behavior when a positioning ends its execution
<a href="#">AM_ErrorSourceEnum</a>	It contains the possible sources of error
<a href="#">AM_FBErrorEnum</a>	It contains the possible error causes when an FB error is occurred (only relevant in AxisMgr)
<a href="#">AM_FieldbusEnum</a>	It contains the information
<a href="#">AM_HomingMethodsEnum</a>	It contains all the possible Homing methods
<a href="#">AM_NewCommandFailedBehaviorEnum</a>	It defines the behavior of AxisMgr when a new command fails before to get in execution
<a href="#">AM_PowerOffOptionEnum</a>	It defines the way the AxisMgr disables the drive.
<a href="#">AM_PosTypeEnum</a>	It contains the possible positioning types.
<a href="#">AM_TouchProbeModeEnum</a>	It contains the possible mode of operation of the Touch Probe.
<a href="#">AM_UserErrorEnum</a>	It contains the possible causes in the case of a User_Error
<a href="#">EN_State_CAN_ETC</a>	It contains the possible states that the CANOpen/EtherCAT slave can reach



## 4.4 Overview of the Structures

Structure	Description
<a href="#">AM_CamDataType</a>	It contains the data relevant for the CamIn function
<a href="#">AM_CommandDataType_ACTIVE</a>	It contains the data relevant for the ACTIVE inverter series
<a href="#">AM_CommandDataType_AGL</a>	It contains the data relevant for the AGILE inverter series
<a href="#">AM_CommandDataType_Axia</a>	It contains the data relevant for the AXIA inverter series
<a href="#">AM_CommandDataType_SM</a>	It contains all the data structures relevant for the commands of AxisMgr_SM
<a href="#">AM_ErrorDiagnosticType</a>	It contains the information about an error
<a href="#">AM_GearDataType</a>	It contains the data relevant for the GearIn function
<a href="#">AM_HomingDataType</a>	It contains the data relevant for the Home function
<a href="#">AM_InitDataType</a>	It contains the data relevant for the Initialization
<a href="#">AM_InitParameterType</a>	It contains the parameter information that has to be written during the initialization
<a href="#">AM_JogDataType</a>	It contains the data relevant for the Jog function
<a href="#">AM_PositioningDataType</a>	It contains the data relevant for the Positioning function
<a href="#">AM_PowerDataType</a>	It contains the data relevant for Power function
<a href="#">AM_ProfilePositionDataType_ACTIVE</a>	It contains the data relevant for the Profile Position Mode function (ACTIVE)
<a href="#">AM_ProfilePositionDataType_Axia</a>	It contains the data relevant for the Profile Position Mode function (AXIA)
<a href="#">AM_ProfileTorqueDataType_Axia</a>	It contains the data relevant for the Profile Torque Mode function (AXIA)
<a href="#">AM_ProfileVelocityDataType_ACTIVE</a>	It contains the data relevant for the Profile Velocity Mode function (ACTIVE)
<a href="#">AM_ProfileVelocityDataType_Axia</a>	It contains the data relevant for the Profile Velocity Mode function (AXIA)
<a href="#">AM_QuickStopDataType</a>	It contains the data relevant for the QuickStop function
<a href="#">AM_SetPosDataType</a>	It contains the data relevant for the SetPos function
<a href="#">AM_SetTorqueDataType</a>	It contains the data relevant for the Set Torque function
<a href="#">AM_StopDataType</a>	It contains the data relevant for the Stop function
<a href="#">AM_SuperImposedDataType</a>	It contains the data relevant for the SuperImposed function
<a href="#">AM_TouchProbeDataType</a>	It contains the data structures of the Touch Probe channels
<a href="#">AM_TouchProbeStatusType</a>	It contains the status structures of the Touch Probe channels
<a href="#">AM_TPChannelStatusType</a>	It contains the output status information of one Touch Probe channel
<a href="#">AM_TPChannelType</a>	It contains the input data information of one Touch Probe channel
<a href="#">AM_VelocityModeDataType_ACTIVE</a>	It contains the data relevant for the Velocity Mode function (ACTIVE)
<a href="#">AM_VelocityModeDataType_AGL</a>	It contains the data relevant for the Velocity Mode function (AGILE)
<a href="#">AM_VelocityModeDataType_Axia</a>	It contains the data relevant for the Velocity Mode function (AXIA)
<a href="#">AM_VeloDataType_SM</a>	It contains the data relevant for the Velocity function (AxisMgr_SM)
<a href="#">AM_WarningDiagnosticType_ACTIVE</a>	It contains the information about a warning (ACTIVE)
<a href="#">AM_WarningDiagnosticType_Axia</a>	It groups the warning group informations (AXIA)
<a href="#">AM_WarningGroupType_Axia</a>	It contains the warning information about one single group
<a href="#">ST_ParamType_Common</a>	It contains the input/output data of one generic parameter for the acyclic communication



## 4.5 Overview of the Alias

Alias	Description
AM_Diagnostic	It represents an array of <a href="#">AM_ErrorDiagnosticType</a>

## 4.6 Overview of the Unions

Union	Description
RW_Data	It groups all the data type which can be read/written in a parameter generic access.

## 4.7 Overview of the Dependencies

Library Name	Placeholder	Company	Version
Tc3_BA2_Common	Tc3_BA2_Common	Beckhoff Automation GmbH	2.1.14.0
Util	Util	System	3.5.11.0
Tc2_MC2	Tc2_MC2	Beckhoff Automation GmbH	3.3.46.0
Tc2_MC2_Camming	Tc2_MC2_Camming	Beckhoff Automation GmbH	3.3.15.0
Tc2_Standard	Tc2_Standard	Beckhoff Automation GmbH	3.3.3.0
Tc2_System	Tc2_System	Beckhoff Automation GmbH	3.4.25.0
Tc3_Module	Tc3_Module	Beckhoff Automation GmbH	3.3.21.0
Tc2_EtherCAT		Beckhoff Automation GmbH	3.3.19.0

### NOTE

- The library has been tested with the dependency of version as listed above. Compatibility with other library version is not guaranteed
- Once added in the project, it may be possible that some dependency placeholders are resolved with a different version of the same library. In most occasions, this does not represent a problem. Nevertheless, it may happen that, mostly when there is a big version number gap, errors are reported when building the library. In this case, consider to manually replace the affected library's placeholder according to the table above.
- It may happen that the target device is not able to resolve some placeholder. In this case, once downloaded and installed the library, it may be necessary to manually assign the placeholder where it is missing.



## 5 Overview of Axis Manager

The main goal of the Axis Manager, described in this document, is to help the user to easily implement an effective control of an axis using Bonfiglioli's inverter and drive solution. The FBs provided by the library uses a field bus to give commands and to check the status of the inverter (control must be set to *State Machine*). The FB are designed in order to be as more compliant with CiA402 as possible, but they are specifically developed to work in combination with Bonfiglioli's inverter only.

The Axis Managers provided in the library are designed to work mainly with the motion profile defined by the CiA402 specifications (for those that are supported by each kind of inverter). Depending on the inverter model and on the scope of the application, Axis Managers are divided by three main types:

- **AxisMgr\_SM**: this FB is developed to work mainly with CiA402 Cyclic Synchronous Position Mode 8. This manager must be used in combination with TwinCAT NC axis.  
[AxisMgr\\_SM](#) uses PLCOpen Motion blocks in its inside, giving the right sequence of command to them and collecting the diagnostic in one single point. Please refer to PLCOpen and Tc2\_MC2 documentations for further informations.  
It is suitable for real and virtual axes. In both cases, the target controller should have real-time capabilities. In case of real axis, motion control trajectories are computed by the motion controller, then sent to the inverter point-by-point at each cycle time. The controller that runs this kind of FB needs a real-time system and a deterministic fieldbus as well. Those devices which do not satisfy these requirements may fail. [AxisMgr\\_SM](#) is therefore essential when it comes to synchronize multiple axes together (e.g.: Gearing, Camming...), in order to perform more complex movements.
- **AxisMgr\_<inverter>\_<fieldbus> ("Light")**: these kind of Axis Manager ("Light") are developed specifically depending on the inverter model and the fieldbus used. They support many operating modes, both standard (i.e.: defined by CiA402) and manufacturer specific. Kinematics are then managed by the inverter itself; the controller, thanks to these FBs, only gives commands in the proper way. In this case the controller could be less performing, since it does not carry out trajectories calculation, and there is no need of a deterministic field bus as well. Nevertheless, an active license of the fieldbus used should be active on the controller.
- **AxisMgr\_Group\_<other function>**: these FB can be used to manage a group of Axis Managers (both "SM" and "Light") instead of controlling each axis individually. It comes very useful in order to give commands and receive the diagnostic to/from several axes just from one single point. It is possible to define as many groups as the user wants, and to assign which Axis Managers should take part of. Each Axis Manager assigned to a group publish its diagnostic to the group diagnostic, so then the user can have a higher overview of the system.  
Based on the same logic of AxisMgr\_Group, function-oriented groups have been created. These group can support specific commands that relate to the whole group.





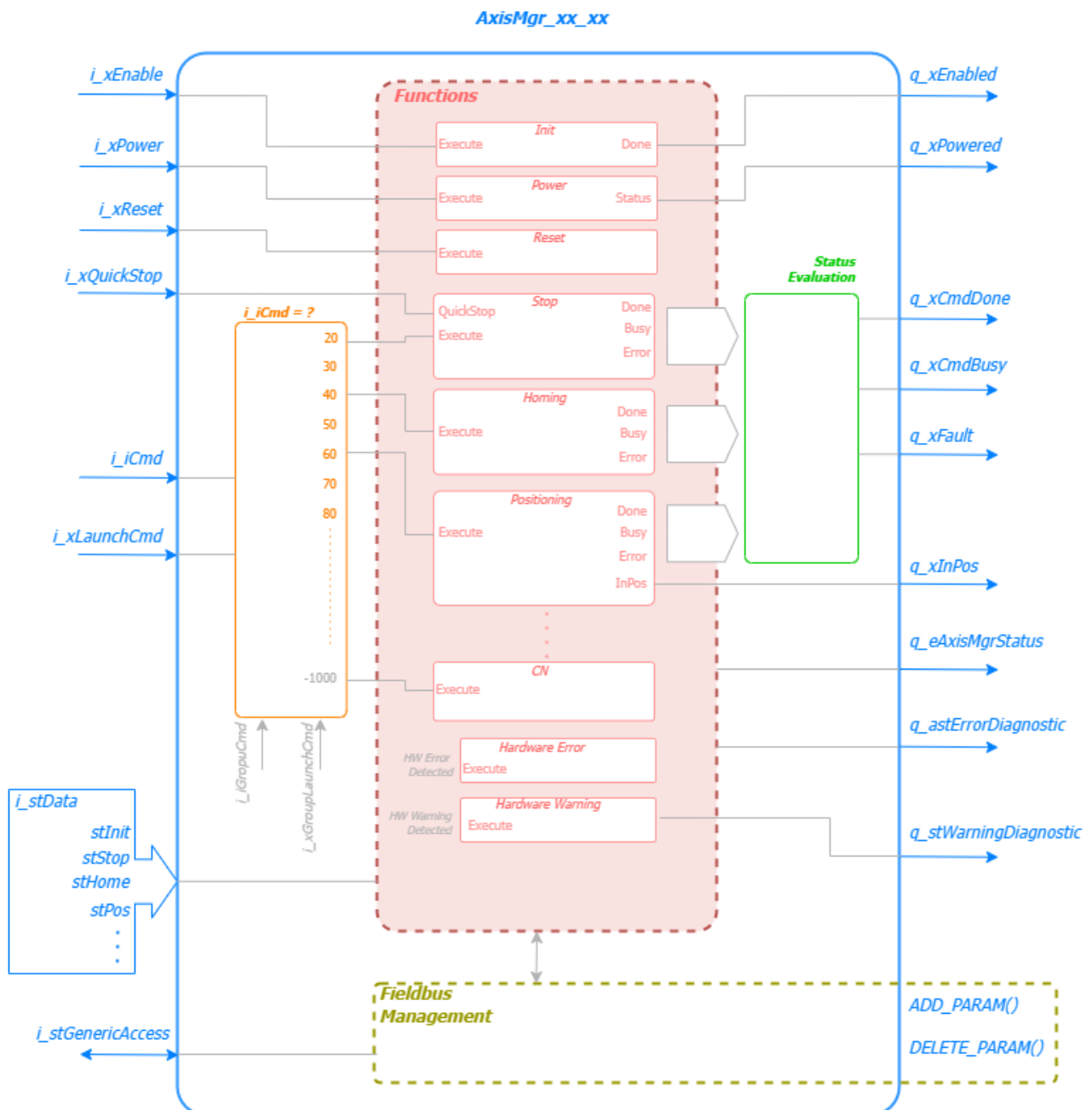
## 5.2 AxisMgr base model

All the Axis Managers share a common interface and behavior, as well as transitions between states. Axis Manager is the engine that runs several **Functions** and it manages the fieldbus as well. These functions are the basic bricks responsible of a movement or functionality that is triggered by the Axis Manager interface.

### 5.2.1 General architecture

Depending on the type of the Axis Manager, the inverter model and the fieldbus used, different functions are supported, while some other are missing (e.g. *AxisMgr\_AGL\_<fieldbus>* does not support any Positioning-related function). This means that input and output interface and their behavior can change accordingly, as well as relationship between *i\_iCmd* and functions.

The image below shows an architecture example of an Axis Manager. It does not refer to a specific FB, it only gives a general overview. Please refer to the specific AxisMgr description in the related chapter of this document.





## 5.2.2 Base interface

A base input/output interface is provided by each AxisMgr. All the other input and output signals are specific for the FB and it will be described in the related chapter.

The base interface is then replicated on the AxisMgr Group interface. The same input triggered at the Group side will be spread to the input of each AxisMgr added in the Group.

Input	Data Type	Description
i_xEnable	BOOL	Set to TRUE to enable the execution of the FB. Execution of the initialization of the drive will start if not yet carried out. Once the initialization has been done, resetting of <i>i_xEnable</i> won't have any effect.
i_xPower	BOOL	Rising edge: power the axis. Falling edge: disable the axis.
i_xReset	BOOL	Rising edge: clear drive's fault and FB's diagnostic
i_xQuickStop	BOOL	Rising Edge: put the axis in <i>EmergencyStop</i> state and trigger the emergency ramp. As long as it remains TRUE, it prevents the axis to get powered.
i_iCmd	INT	Command selection input. It defines which functions should be executed. The possible values and the commands supported are listed in the FB description. The command will be taken into account only after a rising edge on <i>i_xLaunchCmd</i> .
i_xLaunchCmd	BOOL	Rising Edge: launch the command selected in <i>i_iCmd</i> .
i_stGenericAccess	<a href="#">ST_ParamType_Common</a>	Acyclic access to parameters. This structure is already defined to easy access acyclic data.

Output	Data Type	Description
q_xEnabled	BOOL	TRUE: FB is enabled ( <i>i_xEnable</i> = TRUE) and initialization has finished.
q_xPowered	BOOL	TRUE: the axis is powered (i.e.: in case of real axis, motor is energized)
q_xCmdBusy	BOOL	TRUE: the function indicated by <i>i_iCmd</i> is still busy.
q_xCmdDone	BOOL	TRUE: the function indicated by <i>i_iCmd</i> has been finished (Relevant only in commands that can reach a finish, e.g.: Positioning). It remains TRUE as long as <i>i_xLaunchCmd</i> stays TRUE. In the case <i>i_xLaunchCmd</i> is set back to FALSE before the function reaches its end, then <i>q_xCmdDone</i> will rise just for a task cycle.
q_xFault	BOOL	TRUE: a diagnostic error is pending in the <i>q_astErrorDiagnostic</i> list.
q_xOperational	BOOL	TRUE: fieldbus communication is running.
q_xRemoteControl	BOOL	TRUE: the drive is ready to receive commands from the PLC (i.e.: Remote bit on Status Word is set = Hardware/software limitation is released, StateMachine control is set, fault reaction not active). Always FALSE for <a href="#">AxisMgr_SM</a> .
q_eAxisStatus	<a href="#">AM_AxisMgrStatusEnum</a>	It returns the actual status of the axis
q_astErrorDiagnostic	AM_Diagnostic	it contains the errors diagnostics. The first entry in the list is always the most recent error event. The size of the array can be adjusted modifying <i>_AM_K.k_iDiagnostic</i> from <i>Library Manager-&gt;Library Parameters</i> . Keep it low to optimize the memory usage.
q_strAxisName	STRING	It returns the name of the FB instance. It is used in the Group diagnostic.



### 5.2.2.1 Methods and properties

#### Parameter\_access (Property)

By means of this property the user can access the interface methods of the internal fieldbus management (*FB\_GenericAccess\_<fieldbus>*) used by the AxisMgr. In addition to the *i\_stGenericAccess* (which is already managed), the user can add/delete user specific acyclic data of type [ST ParamType Common](#).

### 5.2.3 Common behavior

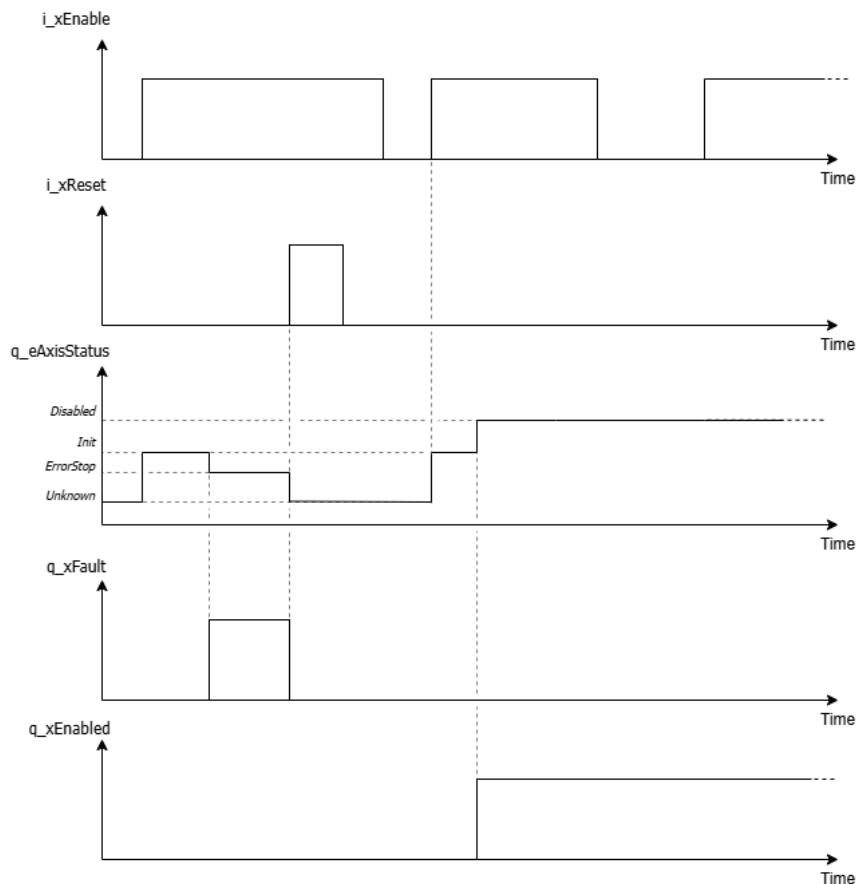
As mentioned in chapter 5.1, Axis Manager passes through the states due to a trigger that comes from the user program either via a Boolean input (*i\_xPower*, *i\_xReset*, *i\_xEnable*...) or via control command *i\_iCmd* and the *i\_xLaunchCmd*. Moreover, there are states (e.g.: *ErrorStop*) which are reached from an external condition (like an inverter fault), and states that are reached due to the successful completion of another (e.g. axis goes automatically in *Standstill* after a *Position* done).

Triggers that comes from the interface are usually evaluated on the edge (rising or falling or both) of the input, and not on the level.

#### 5.2.3.1 From *Unknown* to *Disabled*

As soon as *i\_xEnable* becomes *TRUE*, *Init* state will be reached. This will happen only if *Init* was never completed since the beginning of the user program, otherwise, a new rising edge of *i\_xEnable* won't occur in *Init* state again.

This also means that, if *Init* was not able to be completed (e.g.: invalid pointer to axis, communication faults...), AxisMgr will probably reach *ErrorStop*. After restoring the error (i.e.: *i\_xReset* -> *TRUE*), a new attempt should be performed by means of a new rising edge on *i\_xEnable*. Since Axis Manager reaches *Disabled*, *i\_xEnable* input does not have any effect.





## NOTE

In the case of real axis, Axis Manager will remain in *Init* state as long as the fieldbus communication is not yet running (i.e.:  $q\_xOperational = FALSE$ ). Depending on the fieldbus, this initial check may require specific monitoring to be activated (e.g.: for CANOpen requires the the Heartbeat telegram to be activated on the slave side).

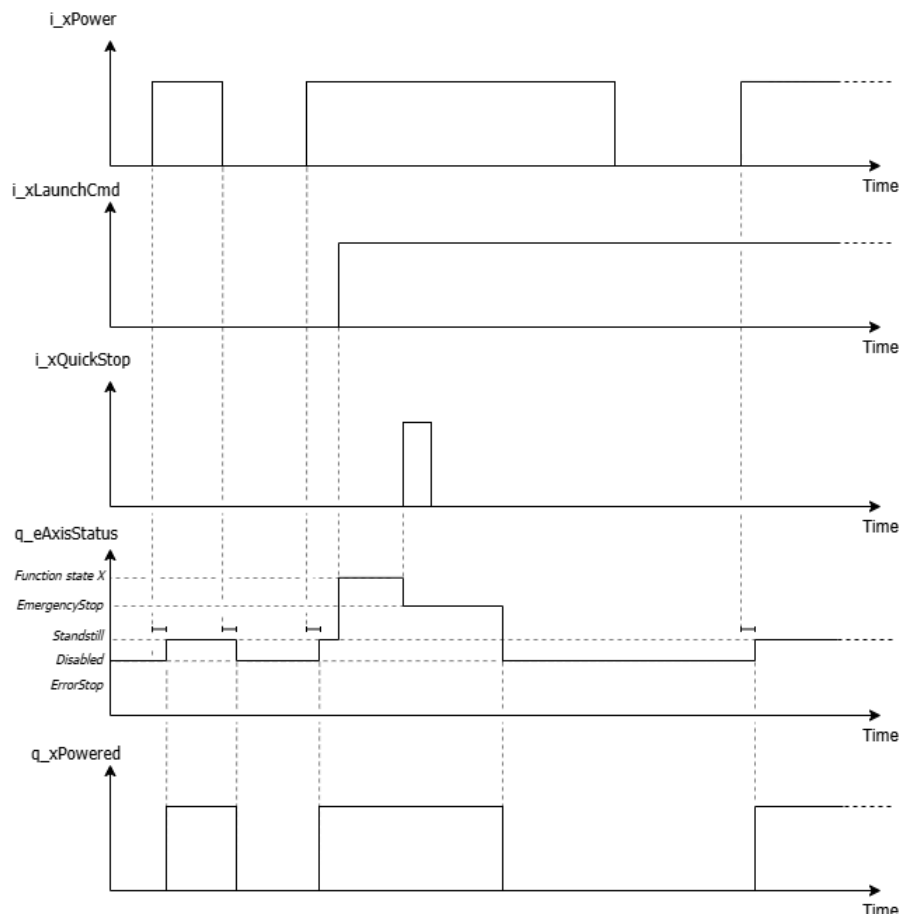
### 5.2.3.2 Disabled, Standstill and EmergencyStop

Setting the  $i\_xPower$  input, the axis reaches *Standstill* from *Disabled*, and it goes back to *Disabled* when it is reset. However, if the axis becomes *Disabled* due to other reasons (e.g. at the end of an *EmergencyStop*) and  $i\_xPower$  is still *TRUE*, then the axis does not get *Standstill* automatically, but can be powered again by means of a new rising edge.

$q\_xPowered$  indicates the actual status of the power stage of the drive. In the case of a real drive, it could take a while to get power-on/off since the  $i\_xPower$  input was set/reset.

When the axis is powered, if a rising edge on  $i\_xQuickStop$  is detected, the emergency ramp will be immediately performed. At the end of the Emergency ramp, the axis will be *Disabled* and *Disabled* state will occur. The value set in  $i\_stData.stQuickStop.rQuickStopDeceleration$  is relevant and will be used differently depending on the AxisMgr:

- **AxisMgr "Light":** The drive's emergency ramp will be triggered. The value set in  $i\_stData.stQuickStop.rQuickStopDeceleration$  will therefore be written into the right drive's parameter. If it is 0, the last value stored in the drive will take effect.
- **AxisMgr\_SM:** AxisMgr will perform a controlled ramp using the value set in  $i\_stData.stQuickStop.rQuickStopDeceleration$ . If it is 0, the last deceleration value used in a move command will take effect.





#### **WARNING**

##### **Unexpected behavior of QuickStop – AxisMgr “Light”**

- ➔ *i\_xQuickStop* input sets/resets the 3<sup>rd</sup> bit of the drive’s remote control word. Behavior of the drive, when a Quick Stop has been requested, may be selectable via *obj. 0x605A Quick Stop Option Code* (if supported). Value of *i\_stData.stQuickStop.rQuickStopDeceleration* is only relevant when this object is set to 2.
- ➔ At the end of a Quick Stop procedure, the drive may keep the motor powered and standstill until the *Holding time* has been expired (drive’s internal parameter, refer to the drive documentation).



The behavior of Axis Manager when powering off the drive can be selected depending on the AxisMgr used:

- ➔ **AxisMgr “Light”:** *i\_stData.i\_stPower.ePowerOffOptionCode* is provided. It defines the transition that should be performed when powering off the drive. Refer to [AM PowerOffOptionEnum](#).

The selection made can be crucial when managing holding brake in the proper way. In addition, the value set in *obj. 0x605B Shutdown Option Code* and *0x605C Disable Operation Option Code* (if supported) is also relevant.



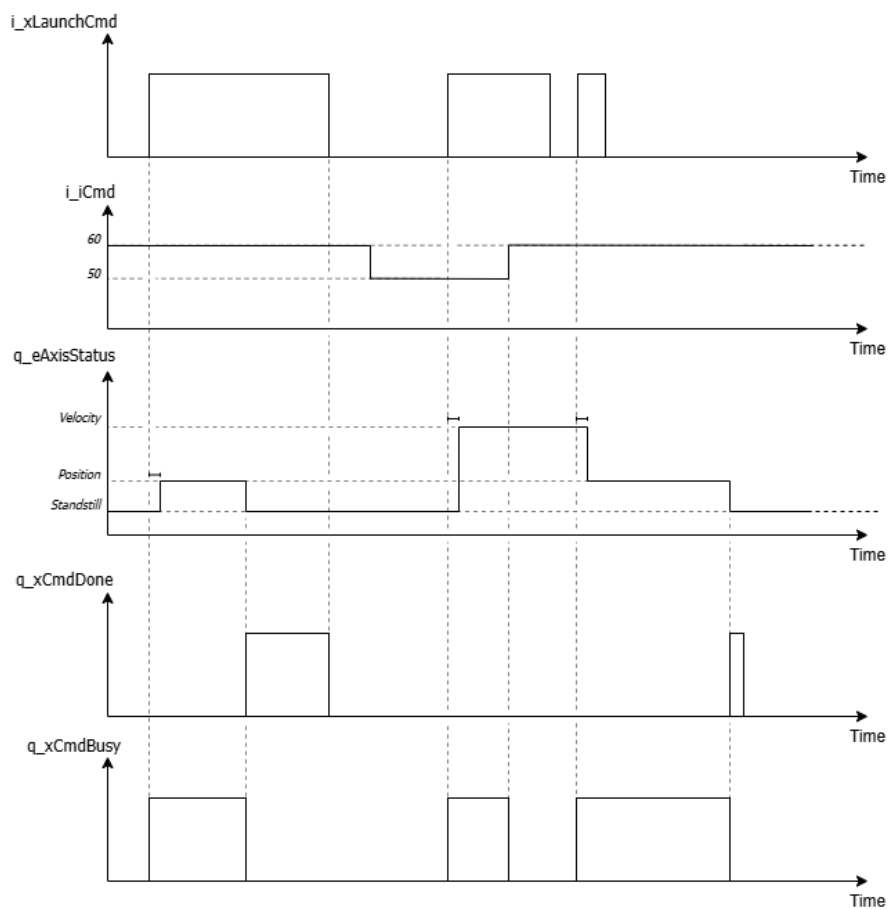
### 5.2.3.3 Launching commands: *i\_iCmd*, *i\_xLaunchCmd*, *q\_xCmdDone* and *q\_xCmdBusy*

The base interface provides a standard way for triggering **Functions**: *i\_iCmd* and *i\_xLaunchCmd* are responsible of the main actions of the axis. Some other AxisMgr-specific functions can be eventually provided and activated by means of a Boolean input (e.g.: *Jog* is activated via *i\_xJogModeActivate*). In this case the behavior will be described in the related chapter.

Functions are supported and can be triggered only in certain states (as shown in chapter 5.1), otherwise an error will occur.

Functions are associated to an unique command number. The function will start its execution as soon as *i\_iCmd* matches this number and a rising edge of *i\_xLaunchCmd* is detected. If none of the functions supported by the AxisMgr matches the number defined by *i\_iCmd*, then nothing will happen.

*q\_xCmdDone* and *q\_xCmdBusy* signal the actual state of the function that has been launched.



#### NOTE

- Depending on the Function, the axis may take some time to reach the execution, since the command was triggered. Nevertheless, *q\_xCmdBusy* is set immediately.
- In order to evaluate the Function results (*q\_xCmdDone* and *q\_xCmdBusy*), value *i\_iCmd* must stay to the same value. If the value changes during the execution, *q\_xCmdBusy* and *q\_xCmdDone* will be reset and/or never raise.
- *q\_xCmdDone* will set as soon as the function has been completed and remains *TRUE* as long as *i\_xLaunchCmd* is kept to *TRUE*. In the case *i\_xLaunchCmd* is reset before the end of the function, then *q\_xCmdDone* will be set for just one task cycle. This also means that, evaluation of *q\_xCmdDone* in a slower task may probably be lost in this situation.



#### 5.2.3.4 *ErrorStop*: possible causes, *q\_xFault* and *i\_xReset*

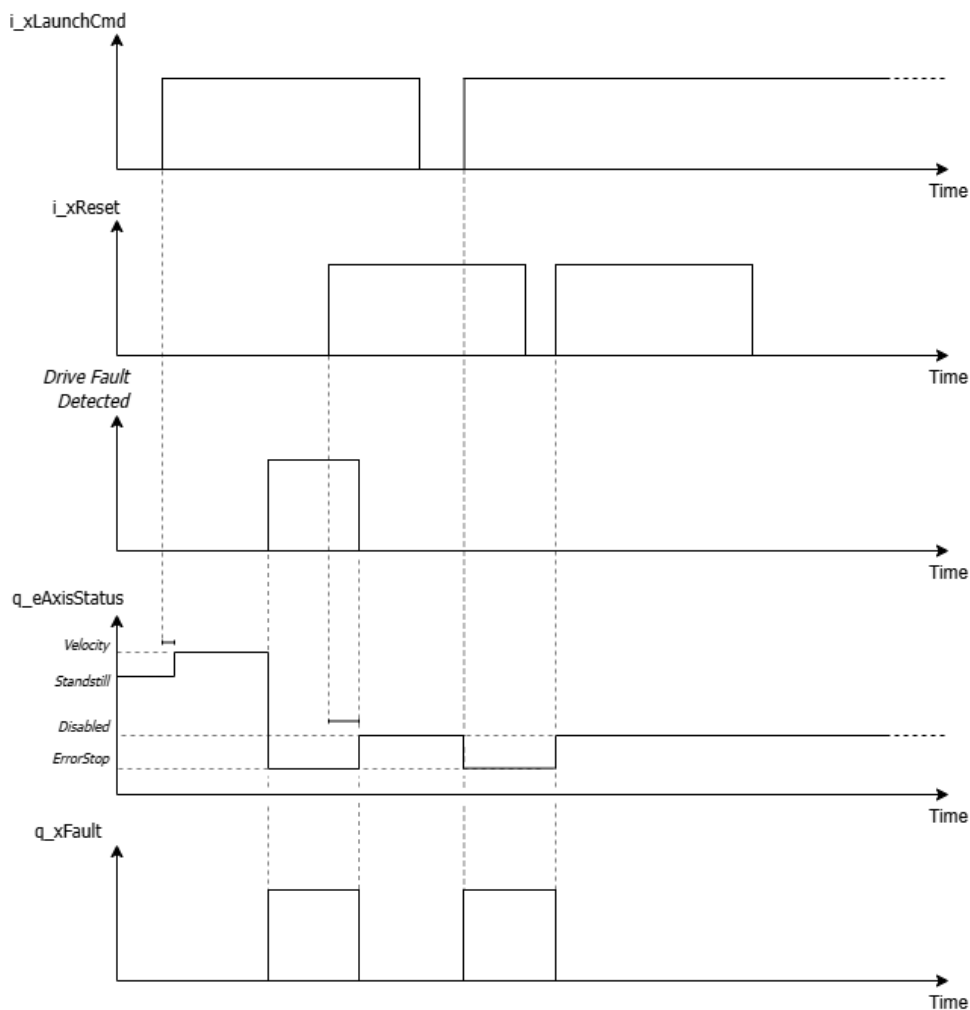
The *ErrorStop* state can be reached due to the following reasons:

- **Hardware error:** a drive error has been detected. AxisMgr will immediately report a fault (i.e.: *q\_xFault* -> *TRUE* and *q\_astErrorDiagnostic* will signals a drive error). The specific error code will be then read out from the drive as soon as it is available. This means that the error code may not be available in the same time as the fault was reported.
- **User error:** the user program has generated an error (e.g.: invalid input data, invalid command state...).
- **FB error:** the functions itself has encoured an issue (e.g.: timeouts, invalid conditions...)
- **Communication error:** communication between the controller and the physical drive has occurred in an issue (parameter access timeout, unable to writing/reading parameters...). Only possible with real axes.

The diagnostic, together with the aforementioned causes, is collected in the *q\_astErrorDiagnostic* (see [AM\\_ErrorDiagnosticType](#)). *q\_xFault* always signals if an entry in the diagnostic is still pernding.

In order to restore the error state, a rising edge given to *i\_xReset* is probably necessary. Since the *ErrorStop* state leads to an emergency ramp, the axis will restore in *Disabled* state.

If reset procedure is completed successfully, then *q\_astErrorDiagnostic* is completely cleared.





#### NOTE

- The behavior of the axis in the case of an *ErrorStop* condition is the same as triggering *i\_xQuickStop* (see chapter [5.2.3.2](#)).
- If an hardware error was detected, the reset procedure may take some time to acknowledge the drive fault. If the drive fault is non-acknowledgeable, the reset procedure will probably end into an error.
- If the error is caused by an user or FB error, then the restoration, thanks to *i\_xReset*, is immediate.
- *i\_xReset* is always evaluated on the rising edge.



AxisMgr, before to put the new function in execution, performs an initial check to establish whether the function could be launched in the actual state and if the input data (i.e.: *i\_stData*) are valid. If it fails, then the reaction depends on *i\_stData.stInit.eNewCommandFailedBehaviorEnum*.

- **ErrorStop (default):** the new command, that resulted in an error, puts the AxisMgr in *ErrorStop* state and any ongoing movement will be stopped accordingly.
- **Refuse:** the failure on the request of a new command will not lead to a stop of the ongoing function. AxisMgr will only report the error in the *q\_stErrorDiagnostic* and *q\_xFault* will be set to *TRUE*. The ongoing function won't be aborted as if the new failed command was never been requested.

*i\_stData.stInit.eNewCommandFailedBehaviorEnum* is evaluated and stored during *Init* state. Changing this field when AxisMgr was already initialized won't have any effect.



## 5.2.4 Fieldbus management

AxisMgr are designed to manage a specific fieldbus. The fieldbus used usually appears in the name of the FB ([AxisMgr SM](#) only works in combinations with either CANOpen or EtherCAT). Communication with the drive is established thanks to **Cyclic** and **Acyclic** data (e.g.: in the case of EtherCAT and CANOpen, PDO and SDO):

- **Cyclic data:** those information are sent/received to/from the drive in a cyclic manner, and updated at each bus cycle task. Depending on the used AxisMgr, cyclic data represent an integral part of the input/output interface of the FB and should be directly linked to the I/O mapping of the device.



### **WARNING**

#### **CYCLIC DATA**

- Since the I/O information to/from the physical remote device are updated at each bus cycle task, AxisMgr **MUST** be called in the same task. Please make sure to satisfy this rule in order to avoid unwanted behavior of the equipments.
- Some functions requires specific cyclic data to be exchanged with the drive. If those information cannot be transferred/received (e.g.: they has not been mapped), the functions may not work as expected.
- Cyclic data can be eventually assigned to the FB in/out interface by means of assignments in the user program. The user program should not perform any other operations (scaling, casting) that can alterate the datas to/from the physical device.

- **Acyclic data:** AxisMgr may also need to access parameters acyclically. This could happen in some certain conditions (e.g.: reading the fault code when an hardware error was detected), or before the execution of a certain function (e.g.: writing all the homing parameters before to start an home run). Access to acyclic parameters is normally managed internally.



### **WARNING**

#### **ACYCLIC DATA**

- Acyclic data are normally managed one by one by AxisMgr. This is because the drive can only manage one parameter access at time. Considering this, the user should **NOT** perform any acyclic request in the user program outside the scope of the AxisMgr. If more than one request at the same time occur, it may lead to communication faults.  
The user should use the fieldbus management interface provided by the AxisMgr directly (i.e.: `i_stGenericAccess`, `ADD_PARAM()`, see [ST ParamType Common](#)), allowing user-defined parameters to be accessed with the right sequence.



### 5.2.4.1 CANOpen/EtherCAT AxisMgr

AxisMgr of type AxisMgr <inverter> CAN ETC and AxisMgr SM can either work in combination with CANOpen or EtherCAT fieldbuses. The examples below can be applied for both the fieldbuses.

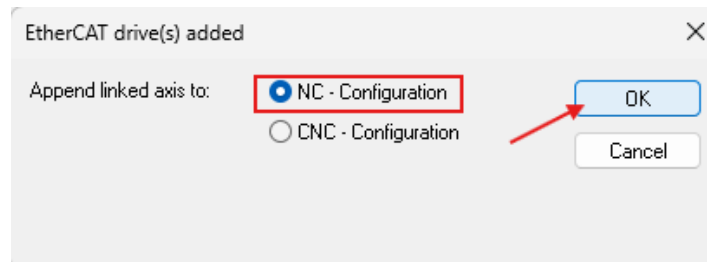
#### NOTE

##### CANOpen Heartbeat

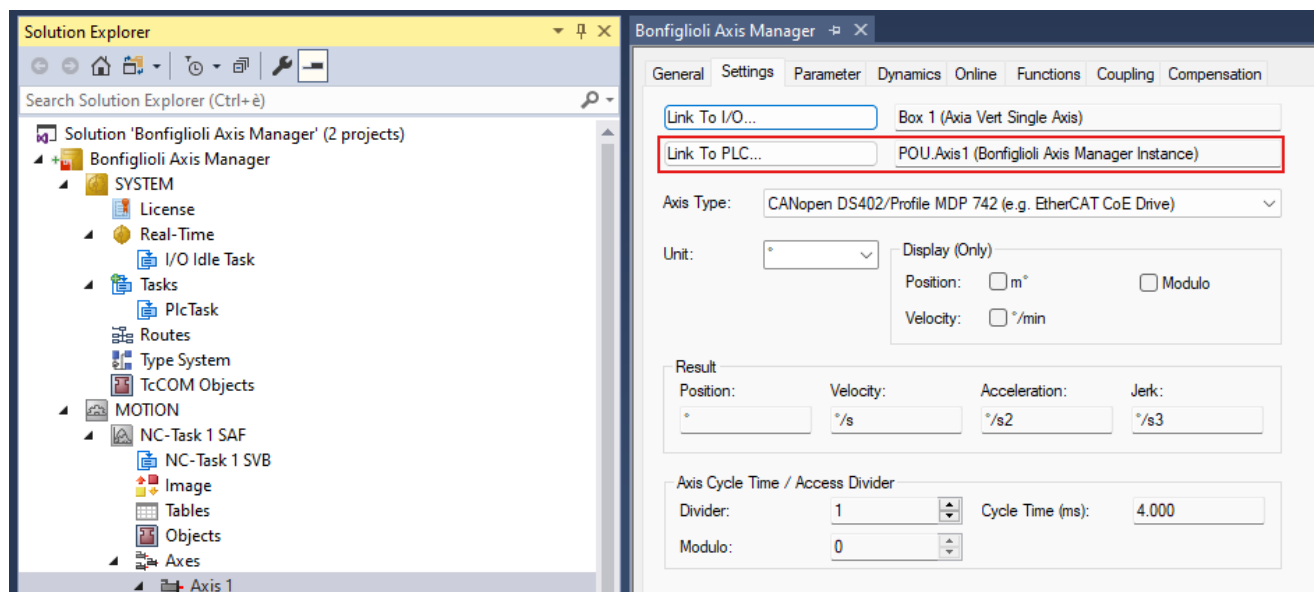
In the case CANOpen is used, always set the Heartbeat telegram to be sent from the drive to the PLC, so Axis Manager can evaluate the slave actual state.

#### AxisMgr\_SM

Mapping of cyclic data is, by default, made automatically by TwinCAT. When adding the device (either via the Scan of the network or manually), the user will be asked to eventually link the device mapping to the NC axis. Check the *NC-Configuration*, then *OK*. By doing this, the required axis data are linked to the PDO In/Out interface. Please make sure to provide the required information in the drive PDO mapping (i.e.: Control and Status Word, Mode of Operation and Mode of Operation Display, Target and Actual position).



An instance of Tc2\_MC2.AXIS\_REF must be declared in the user program by the user. Then used to be linked to the NC axis.





## WARNING

### AxisMgr\_SM OPERATION

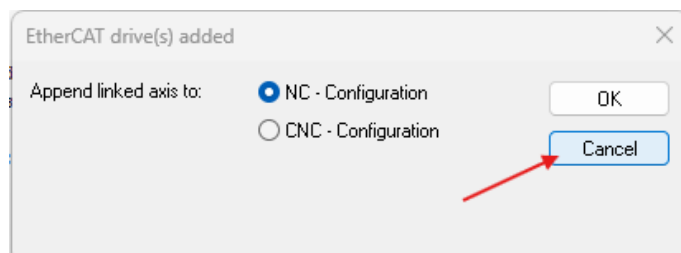
- ➔ Distributed Clock (EtherCAT) or SYNC telegram (CANOpen) should be activated for a proper synchronization. Please always ensure that the reference time assigned to those functions is equal to the bus cycle time.
- ➔ Please always ensure that the automatic mapping performed by TwinCAT matches the In/Out addresses of the I/O mapping of the physical device. If this condition is not satisfied, the device may not work as expected.

### AxisMgr "Light"

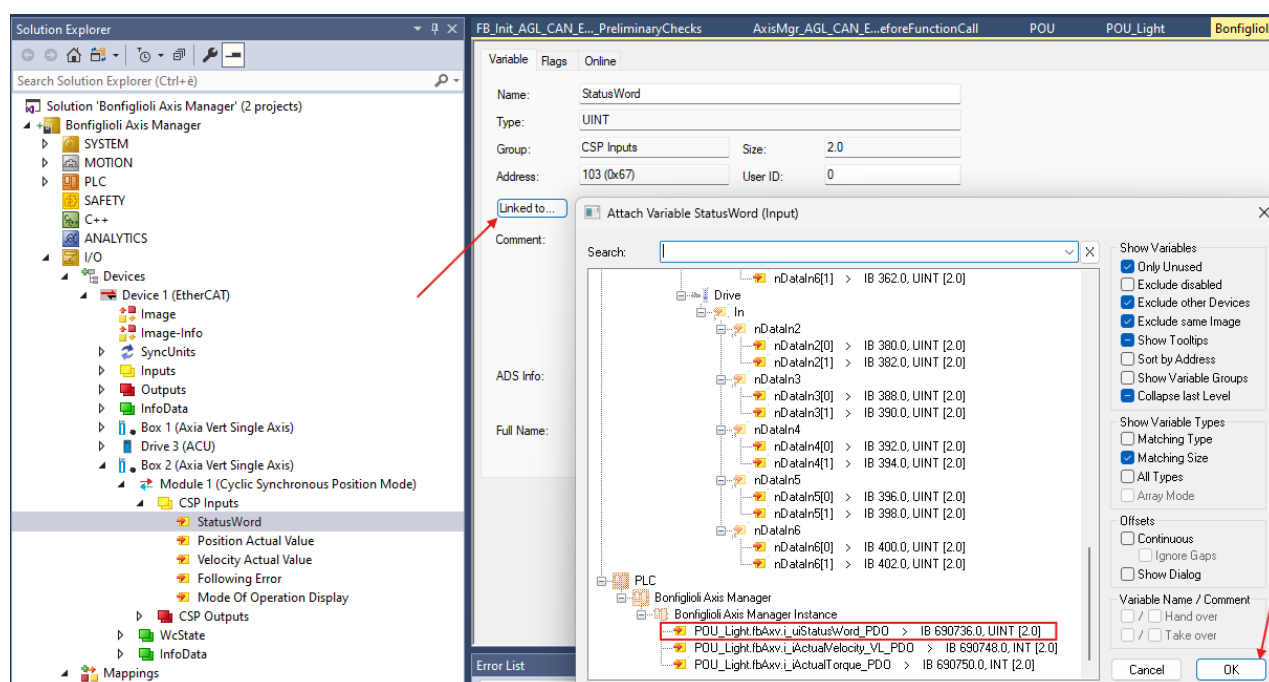
AxisMgr "Light" requires the mapping of cyclic data carried out manually by the user. It can be done either in the PLC program by means of assignments, or linking the In/out variables directly to the I/O mapping of the device (image below).

It is only necessary to link the cyclic data relevant for the functions that the AxisMgr is supposed to use. AxisMgr will NOT report any error if any mandatory link is missing.

In this case, when adding the device, click to *Cancel* in the dialog box that will appear, so no NC axis will be created linked to this device.



Once an instance of AxisMgr "Light" has been declared in the user program, the PDO mapping should be linked (one by one) manually by the user. For each of the required object, under the device tree, link the FB information by using the *Linked to...* button.





### 5.3 Overview of AxisMgr Functions

As mentioned in chapter [5.2](#), AxisMgr manages functions responsible of axis control, movement and diagnostic.

Depending on the type of AxisMgr, different functions are implemented (based on the operating mode, the fieldbus and the inverter model).

Functions can be triggered via the assigned command selections (i.e.: *i\_iCmd*) and *i\_xLaunchCmd* or via a specific Boolean input. The control sequence and the global result is established by the AxisMgr.

Normally functions need valid input data that can be usually found in the *i\_stData* structured input. Refer to the specific structure chapter for further information.

A pool of **"base" Functions** are probably always present regardless the type of AxisMgr or fieldbus used, since they represent the basic functionality for the axis management:

- **Init** (see chapter [5.2.3.1](#))
- **Power** (see chapter [5.2.3.2](#))
- **Reset** (see chapter [5.2.3.4](#))
- **QuickStop** (see chapter [5.2.3.2](#))
- **Hardware error and warning detection.**

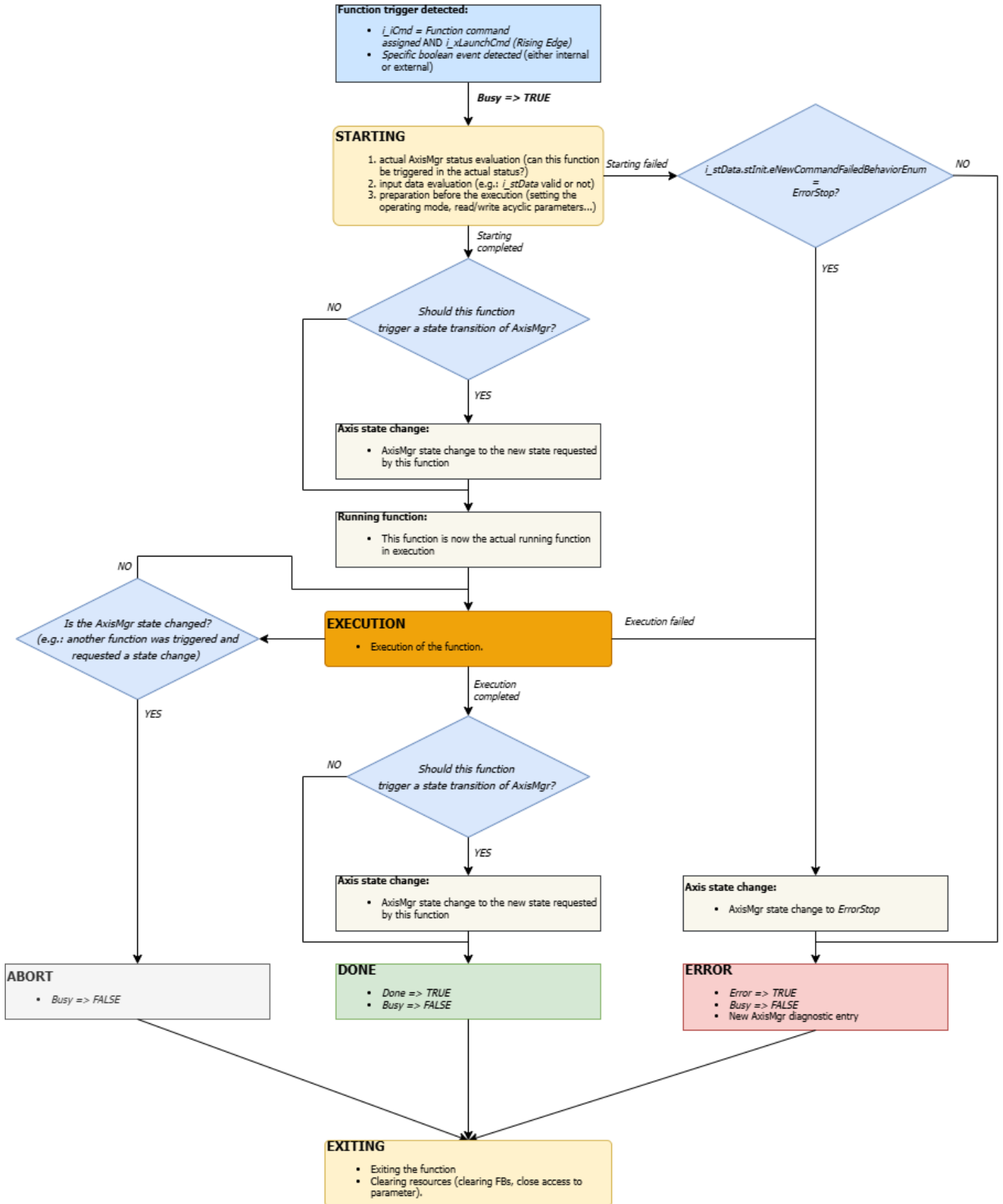
From the request until the end of the function, the general flow chart is shown in the following page. This is only a general overview, it should not be considered as a strict rule for all the functions.

#### NOTE

In addition to the flow chart, functions can implement a "background" routine that is always running, regardless the actual status of the function.

Examples:

- Positionings function may manage the *q\_xInPos* signal cyclically in background
- TouchProbe function does only work in background (when enabled), it is not affected by state changes and/or external aborts.





## 5.3.1 Input/Output data

### 5.3.1.1 *i\_stData* structure

Both *AxisMgr SM* and *AxisMgr "Light"* provide the *i\_stData* structure, which comprehends all the relevant datas for functions to be executed.

The user should only provide the data for the functions that are supposed to be executed by the *AxisMgr*.

*AxisMgr* validates the data values before to execute the function, otherwise an error is probably reported. Moreover, some input data are constantly evaluated and, depending on the function, changes may have immediately effect.

The *i\_stData* structure, as well as FB's output data, contains information (in addition to all the others) that are related to position or speed. Those figures are expressed in **units (u, u/s, u/s<sup>2</sup>)** and are probably affected by conversion factors before to be sent/received by the drive.

### 5.3.1.2 Values conversion factor, user units definition

When working with real axis, information that are sent/receive to/from the drive (either cyclically or acyclically) are probably converted. Moreover, the drive itself features its internal conversion factor, that is applied for the final calculation of the motor turns. The combination of both factors (controller side and drive side) results in the final definition of the **user unit (u)**.

Depending on the *AxisMgr*, conversion is made differently:

- **AxisMgr\_SM**: Informations to/from the real drive are scaled according to what provided in the *NC axis* -> *Enc* -> *Parameters* -> *Scaling factor Numerator/Scaling factor Denominator*. Input and output data (expressed in user units) refers to what defined in these entries. The controller calculates how many *increments* corresponds to the user unit in the application. The number inserted should take into account what it is defined as *increments* in the drive (i.e.: *obj. 0x6092 Feed Constant*). When present, gear box ratio can be also defined in this box. In order to avoid overlapping of factors, keep drive's *obj. 0x6091 Gear ratio* untouched (default = 1).

Parameter	Offline Value	Online Value	T...	Unit
Encoder Evaluations:				
Invert Encoder Counting Direction	FALSE			B
Scaling Factor Numerator	360.0			F * /INC
Scaling Factor Denominator (default: 1.0)	65536.0			F *
Position Bias	0.0			F *
Modulo Factor (e.g. 360.0°)	360.0			F *
Tolerance Window for Module Start	0.0			F *
Encoder Mask (maximum encoder value)	0xFFFFFFFF			D
Encoder Sub Mask (absolute range maximum value)	0x000FFFFFFF			D
Reference System	'INCREMENTAL'			E

- **AxisMgr "Light"**: Informations to/from the real drive are scaled according to what provided to *i\_rFactor\_pos* and *i\_rFactor\_vel*.

- *i\_rFactor\_pos*: it is expressed in *increments/user unit* and defines the scaling factor applied for the **pos. units** informations to/from the drive. It is used for those functions that works in combination with the position controlled mode of operation (e.g.: Profile Position, Profile Velocity, Homing, ecc...). The final calculation of the motor turns is also affected by the drive's *obj. 0x6092 Feed Constant* and *obj. 0x6091 Gear ratio* that are applied afterwards the drive increments.  
*i\_rFactor\_pos* defines how many increments are to be considered for 1 u displacement. The presence of a gear box should be also considered in order to establish the value to set in *i\_rFactor\_pos*. It is always considered as ABSOLUTE.



The following relations result:

- Starting from motor turns, a possible calculation of Position in **u** is as follows:

$$\left( \text{Act. drive increments (e.g.: } i\_diActualPosition\_PDO) [incr] = \left( \text{Act. encoder turns} \times \frac{\text{obj. 0x6091.2 Driving Shaft Revolutions}}{\text{obj. 0x6091.1 Motor Shaft Revolutions}} \times \frac{\text{obj. 0x6092.1 Feed}}{\text{obj. 0x6092.2 (Driving) Shaft Revolutions}} \right) - \text{Home Offset} [incr] \right)$$

$$\text{Act. Position (e.g.: } q\_rActualPosition\_u) [u] = \frac{\text{Act. drive increments} [incr]}{i\_rFactor\_pos [incr/u]}$$

- Which also results:

$$\begin{aligned} \text{Target drive increments (e.g.: } q\_diTargetPosition\_PDO) [incr] &= \\ \text{Target value (e.g.: } i\_stData.stPos.rTargetPosition) [u] \times i\_rFactor\_pos [incr/u] & \end{aligned}$$

Target motor turns =

$$\begin{aligned} \text{Target drive increments} [incr] \times \frac{\text{obj. 0x6092.2 (Driving) Shaft Revolutions}}{\text{obj. 0x6092.1 Feed}} \\ \times \frac{\text{obj. 0x6091.1 Motor Shaft Revolutions}}{\text{obj. 0x6091.2 Driving Shaft Revolutions}} \end{aligned}$$

- $i\_rFactor\_vel$ : it is expressed in *VL Units/user unit* and defines the scaling factor applied for the **vel. units** informations to/from the drive. It is used for those functions that works specifically with the speed controller (e.g.: Velocity Mode). The final calculation of the motor RPM can also be affected by the drive's *obj. 0x604C VL Dimension* that is applied afterwards the drive's VL Units.  
 $i\_rFactor\_vel$  defines how may VL Units are to be considered for 1 u speed. It is always considered as ABSOLUTE.



- The user can eventually decide to leave *obj. 0x6091* and *obj. 0x6092* to their default values, managing the  $i\_rFactor\_pos$  only. Instead it is possible to set  $i\_rFactor\_pos$  to 1, which means that the only active frame of reference is on the drive side.
- The user should consider that the information to/from the drive (in *increments*), are integers value. Setting of *0x6092* to a low value results in big loss of information when converted in a floating number. Depending on the application and the axis travel range, *obj. 0x6092* should be set as big as possible.
- $i\_rFactor\_pos$  can be changed during the execution of the AxisMgr. Depending on the running function, the target values are immediately recalculated and updated the information to/from the drive. This can be useful when the linear travel ratio of the application changes during the movement (e.g.: dynamic diameter recalculation for winding applications).
- $i\_rFactor\_vel$  can be changed during the execution of the AxisMgr. Depending on the running function, the target values are immediately recalculated and updated the information to/from the drive.



### 5.3.2 AxisMgr\_SM supported functions

The following functions are supported by [AxisMgr\\_SM](#) in addition to the basic ones (see chapter [5.3](#)):

Type of AxisMgr	Function	<i>i_iCmd</i> / Bool Input	Description
<a href="#">AxisMgr_SM</a>	Stop ( <a href="#">5.3.2.1</a> )	20	It stops any ongoing movement putting the axis in <i>Stopping</i> state. <i>i_stData.stStop</i> input data is relevant. A successful Stop ends in <i>Standstill</i> state.
	Halt SuperImposed ( <a href="#">5.3.2.2</a> )	21	It stops the SuperImposed movement ONLY, leaving the underlying movement running. <i>i_stData.stStop</i> input data is relevant
	SetPos ( <a href="#">5.3.2.3</a> )	30	It references the axis "on-the-fly" at the set position. <i>i_stData.stSetPos</i> input data is relevant.
	Velocity ( <a href="#">5.3.2.4</a> )	50	The axis moves endless with the specified dynamic settings. It puts the axis in <i>Velocity</i> state. <i>i_stData.stVelo</i> input data is relevant.
	Positioning ( <a href="#">5.3.2.5</a> )	60	The axis moves towards the specified target position (either absolute or relative), with the specified dynamic settings. It puts the axis in <i>Position</i> state. <i>i_stData.stPos</i> input data is relevant.
	Gearing ( <a href="#">5.3.2.6</a> )	70	The axis is engaged in velocity with a master axis and it follows his movements with the defined ratio. Engagement will be performed with the specified dynamic settings, <i>i_stData.stGearing</i> input data is relevant. It puts the axis in <i>Synchronized</i> state.
	Camming ( <a href="#">5.3.2.7</a> )	80	The axis is engaged in position with a master axis and it follows the master/slave position relationship defined by the given Cam Disk. Engagement will be either performed with the specified dynamic settings or depending on the defined master/slave sync positions. <i>i_stData.stCamming</i> input data is relevant. It puts the axis in <i>Synchronized</i> state.
	SuperImposed ( <a href="#">5.3.2.1</a> )	90	It performs a super-imposed movement, without aborting the ongoing one. <i>i_stData.stSuperImposed</i> input data is relevant. It doesn't change the actual AxisMgr state.
	Jog ( <a href="#">5.3.2.2</a> )	<i>i_xJogModeActivate</i>	Axis is controlled forward and backward according to <i>i_xJogForward</i> and <i>i_xJogBackward</i> . Specifically designed for manual operations of the axis. <i>i_stData.stJog</i> input data is relevant.



	TouchProbe ( <a href="#">5.3.4.1</a> )	See description	As long as it is enabled (i.e.: <i>i_stData.stTouchProbe.stTPx.xTouchProbeEnable = TRUE</i> ), AxisMgr reads out the related TouchProbe channel information.
	Set Controller Mode ( <a href="#">5.3.2.3</a> )	100	It changes the Mode of Operation with which the axis will work. By default, AxisMgr_SM works in CSP mode. By means of this function, the mode of operation can be switched to CSV or CST.
	Set Torque ( <a href="#">5.3.2.4</a> )	110	When in CST Mode (Set Controller Mode function shall be triggered beforehand), AxisMgr sends the set torque cyclically to the drive to be used as the reference torque. It puts the axis in <i>Torque</i> state.

#### NOTE

- Functions of [AxisMgr\\_SM](#) that uses SoftMotion FBs are set to work in BufferMode = Aborting.



#### CAUTION

##### MASTER-FOLLOWER operation

Instances of AxisMgr\_SM that are supposed to work in *Synchronize* motion, should be always called (in the program task) **after** the AxisMgr\_SM instance that represent the master axis. The non-compliance of this rule may lead to unwanted behavior due to positions lags.



### 5.3.2.1 Stop function (AxisMgr\_SM)

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
20	v. 3.1.2.0	<i>Position</i> <i>Stopping</i> <i>Synchronized</i> <i>Velocity</i> <i>Standstill</i>	<i>Stopping</i>	<i>Standstill</i>	<a href="#">AxisMgr_SM</a>	<a href="#">AM_StopDataType</a>

The Stop function stops any ongoing movement putting the axis in *Stopping* state. *i\_stData.stStop* input data is relevant. A successful Stop ends in *Standstill* state.

As soon as the functions has been triggered by means of *i\_iCmd* = 20 and a rising edge given to *i\_xLaunchCmd*, [AxisMgr\\_SM](#) reaches the *Stopping* state. The axis will ramp down with the defined *i\_stData.stStop.lrdDeceleration* expressed in  $u/s^2$ . When the axis reaches speed 0, then the AxisMgr will get *Standstill* state.

In the case of invalid input data (e.g.: *i\_stData.stStop.lrdDeceleration*  $\leq 0$ ) the function will be triggered anyway and the deceleration used in the last move command will be used.

#### NOTE

- Input data are constantly evaluated. Any change during the execution of the function will immediately take effect.
- When using jerk limited trajectories (i.e.: *Velocity ramp type*  $\neq$  *Trapezoid*), the maximum Jerk of the movement is always equal to *i\_stData.stStop.lrdDeceleration*  $\times 10$ .



#### WARNING

##### Interruption of Set Torque function of real axis

- When in *Torque* state, stop function shall not be launched. A power-off command should be executed instead. Otherwise, an error will probably occur.

### 5.3.2.2 Halt SuperImposed function (AxisMgr\_SM)

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
21	v. 3.1.2.0	<i>Standstill</i> <i>Position</i> <i>Velocity</i> <i>Synchronized</i> <i>Jog</i>	No change	No change	<a href="#">AxisMgr_SM</a>	

This function stops the SuperImposed movement without aborting the actual underlying movement. The AxisMgr state won't be changed.

As soon as the functions has been triggered by means of *i\_iCmd* = 21 and a rising edge given to *i\_xLaunchCmd*, the function begins its execution. The function will terminate the actual SuperImposed movement.



### 5.3.2.3 SetPos function (AxisMgr\_SM)

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
30	v. 3.1.2.0	<i>Standstill</i> <i>Velocity</i> <i>Synchronized Position</i> <i>Disabled</i> <i>Jog</i> <i>Stopping</i>	No change	No change	<a href="#">AxisMgr_SM</a>	<a href="#">AM_SetPosDataType</a> <a href="#">AM_PosTypeEnum</a>

SetPos function comes useful when the axis has to be referenced "on-the-fly", without interrupting the actual ongoing movement. *i\_stData.stSetPos* input data is relevant.

As soon as the functions has been triggered by means of *i\_iCmd* = 30 and a rising edge given to *i\_xLaunchCmd*, the function begins its execution. Depending on the selection made in *i\_stData.stSetPos.eSetPosMode*, the referencing can be either Absolute or Relative.

- **Absolute:** at the end of the function, the new axis position will become equal to the value of *i\_stData.stSetPos.lrPosition*.
- **Relative:** at the end of the function, the new axis position will be equal to *actual axis Position* + *i\_stData.stSetPos.lrPosition*.

SetPos functions does not lead to any movement and does not affect the actual one, it only shifts the position with a logical offset.



In the case SetPos is called in the middle of a *Position*, the axis will reach the original target position, as if SetPos was never called. The result is that the Position command will end with an offset between target and actual position, equal to the amount of SetPos reference position.

#### NOTE

- *q\_xHomed* will reset as soon as the function gets busy. It will be set when the function is successfully finished.
- The actual position of the axis will be updated at the [AxisMgr\\_SM](#) side (i.e.: *q\_lrActualPosition*) but the position read from the drive won't be affected.



### 5.3.2.4 Velocity function (AxisMgr\_SM)

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
50	v. 3.1.2.0	<i>Position</i> <i>Standstill</i> <i>Synchronized</i> <i>Velocity</i> <i>Stopping</i>	<i>Velocity</i>		<a href="#">AxisMgr_SM</a>	<a href="#">AM_VeloDataType_SM</a>

The axis moves endless with the specified dynamic settings. It puts the axis in *Velocity* state. *i\_stData.stVelo* input data is relevant.

A negative value set in *i\_stData.stVelo.lTargetVelocity* results in a negative direction.

As soon as the functions has been triggered by means of *i\_iCmd* = 50 and a rising edge given to *i\_xLaunchCmd*, the function begins its execution.

#### NOTE

- ➔ Input data are constantly evaluated. Any change during the execution of the function will immediately take effect.
- ➔ When using jerk limited trajectories (i.e.: *Velocity ramp type* ≠ *Trapezoid*), the maximum Jerk of the movement is always equal to *i\_stData.stVelo.lDeceleration* x 10.



### 5.3.2.5 Positioning function (AxisMgr\_SM)

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
60	v. 3.1.2.0	<i>Position</i> <i>Stopping</i> <i>Velocity</i> <i>Position</i> <i>Synchronized</i>	<i>Position</i>	<i>Standstill</i>	<a href="#">AxisMgr_SM</a>	<a href="#">AM_PositioningDataType</a> <a href="#">AM_PosTypeEnum</a> <a href="#">AM_EndOfPositionBehaviorEnum</a>

The axis moves towards the specified target position (either absolute or relative), with the specified dynamic settings. It puts the axis in *Position* state. *i\_stData.stPos* input data is relevant.

As soon as the functions has been triggered by means of *i\_iCmd* = 60 and a rising edge given to *i\_xLaunchCmd*, the function begins its execution.

Depending on the positioning type, the function behaves accordingly:

- *i\_stData.stPos.ePositioningType* = *AM\_PosTypeEnum.Relative*:

the target position (*i\_stData.stPos.lrTargetPosition*) represent the distance that the axis will cover. The sign define the direction of the movement.

In this situation, if during the execution, target position is modified, a new relative movement will be executed immediately. The new distance is calculated from the current position of the axis.

In the same way, a new rising edge of *i\_xLaunchCmd* results in a new relative command, starting from the current position of the axis.

- *i\_stData.stPos.ePositioningType* = *AM\_PosTypeEnum.Absolute*:

the target position (*i\_stData.stPos.lrTargetPosition*) represent the absolute quota that the axis must reach.

Changing of *i\_stData.stPos.lrTargetPosition*, *i\_stData.stPos.lrTargetVelocity*, *i\_stData.stPos.lrDeceleration* and/or *i\_stData.stPos.lrAcceleration* will be taken into account and will immediately take effect.

A new rising edge of *i\_xLaunchCmd* results in a new command.

#### NOTE

- ➔ When using jerk limited trajectories (i.e.: *Velocity ramp type* ≠ *Trapezoid*), the maximum Jerk of the movement is always equal to *i\_stData.stPos.lrDeceleration* x 10.

#### **q\_xInPos**

*q\_xInPos* signals wheter the axis is found in the defined *i\_stData.stPos.rPositionWindow* for the time defined in *i\_stData.stPos.uiPositionWindowTime\_ms*. The Position window is considered around the target position that the axis should reach (*Target Position* +/- (*Position Window* / 2)).

*q\_xInPos* can eventually rise before *q\_xCmdDone* and can be evaluated accordingly. Moreover, *q\_xInPos*, will stay TRUE as long as the position remains inside the window and until the axis change its state (i.e.: the axis leaves *Standstill* state).



#### NOTE

If a SuperImposed movement (see chapter 5.3.2.1) was launched during a Positioning, then  $q\_xInPos$  may not be evaluated to establish the end of the Positioning, since the axis will move towards a different target position. In this case the transition from *Position* to *Standstill* should be monitored.

#### enEndOfPositionBehavior

$i\_stData.stPos.enEndOfPositionBehavior$  defines the behavior of [AxisMgr\\_SM](#) when the actual positioning movement reaches its end:

- **GoInStandstill (default):** the axis ends the movement and reaches the *Standstill* state.  $q\_xCmdDone$  can be evaluated to establish the end of the positioning.
- **RemainInPosition:** the axis remains in *Position* state. In this way the user can update the target parameters (e.g.:  $i\_stData.stPos.lrTargetPosition$ ) to perform new movements, avoiding to trigger  $i\_xLaunchCmd$  again.  
In this case  $q\_xCmdDone$  is not relevant and will not be set at each position done,  $q\_xInPos$  should be used instead.

#### 5.3.2.6 Gearing function (AxisMgr\_SM)

$i\_iCmd$	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
70	v. 3.1.2.0	<i>Standstill</i> <i>Position</i> <i>Velocity</i> <i>Synchronized</i> <i>Stopping</i>	<i>Synchronized</i>		<a href="#">AxisMgr_SM</a>	<a href="#">AM_GearDataType</a>

The axis is engaged in velocity with a master axis ( $i\_pstMasterRef$ ) and it follows his movements with the defined ratio. Engagement will be performed with the specified dynamic settings,  $i\_stData.stGearing$  input data is relevant. It puts the axis in *Synchronized* state.

As soon as the functions has been triggered by means of  $i\_iCmd = 70$  and a rising edge given to  $i\_xLaunchCmd$ , the function begins its execution.

The axis will synchronize to the master actual velocity with the defined dynamic setting.  $q\_xInSync$  signals whether the synchronization phase has finished and that the axis speed is engaged with the master.

#### NOTE

- Any change during the execution of the function won't have effect.
- When using jerk limited trajectories (i.e.: *Velocity ramp type*  $\neq$  *Trapezoid*), the maximum Jerk of the movement is always equal to  $i\_stData.stGear.lrDeceleration \times 10$ .



### 5.3.2.7 Camming function (AxisMgr\_SM)

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
80	v. 3.1.2.0	<i>Standstill</i> <i>Position</i> <i>Velocity</i> <i>Synchronized</i> <i>Stopping</i>	<i>Synchronized</i>		<a href="#">AxisMgr_SM</a>	<a href="#">AM_CamDataType</a> <a href="#">MC_CAM_REF</a> (see <i>Tc2_MC2_Camming</i> documentation)

The axis is engaged in position with a master axis and it follows the master/slave position relationship defined by the given Cam Disk. Engagement will be either performed with the specified dynamic settings. *i\_stData.stCamming* input data is relevant. It puts the axis in *Synchronized* state.

As soon as the functions has been triggered by means of *i\_iCmd* = 80 and a rising edge given to *i\_xLaunchCmd*, the function begins its execution.

The axis will synchronize to the cam profile depending on the master position with the defined dynamic setting. *q\_xInSync* signals whether the synchronization phase has finished and that the axis position is engaged with the cam profile.

Changing of the Cam Disk reference (*i\_stData.stCamming.pCamRef*) will result in another synchronization phase to the new cam profile, *q\_xInSync* will be set to FALSE and set again once finished.

#### Cam Disk definition

The Cam Disk, which is responsible of the position relationship between master and slave, is defined via the MC\_CAM\_REF object. Basically, the user should manually create an instance of MC\_CAM\_REF and provide its pointer to the *i\_stData.stCamming.pCamRef*.

#### Relative/Absolute Cam

Depending on the selection made in *i\_stData.stCamming.xMasterAbsolute* and *i\_stData.stCamming.xSlaveAbsolute* the following will result:

<i>xMasterAbsolute</i>	<i>xSlaveAbsolute</i>	Result
TRUE	TRUE	The actual master position represents the corresponding point in the cam profile. Slave will synchronize to the related absolute position in the cam profile <b>(default)</b> .
TRUE	FALSE	The actual master position represents the corresponding point in the cam profile. The slave will synchronize at the related slave position the in cam profile, relative to the actual position of the slave.
FALSE	TRUE	The actual master position represents the initial point of the cam profile. Slave will synchronize to the initial slave position of the cam profile (absolute).
FALSE	FALSE	The actual master and the slave position are the initial point of the cam. Slave will synchronize at the current position.



### Periodic/non-periodic Cam

Depending on the setting in *i\_stData.stCammin.xPeriodic* the cam can be either periodic, endless (TRUE) or one-shot (FALSE).

#### NOTE

- When using jerk limited trajectories (i.e.: *Velocity ramp type* ≠ *Trapezoid*), the maximum Jerk of the movement is always equal to *i\_stData.stCamming.lfRephaseDec* x 10.

### 5.3.2.1 SuperImposed function (AxisMgr\_SM)

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
90	v. 3.1.2.0	<i>Position</i> <i>Velocity</i> <i>Synchronized</i> <i>Jog</i>			<a href="#">AxisMgr_SM</a>	<a href="#">AM_SuperImposedDataType</a>

The axis performs a super-imposed movement over the ongoing one. At the end of the super-imposed, the axis will have recovered the specified distance.

As soon as the functions has been triggered by means of *i\_iCmd* = 90 and a rising edge given to *i\_xLaunchCmd*, the function begins its execution.

SuperImposed will recover the specified distance (i.e.: *i\_stData.stSuperImposed.lfDistance*) using the specified dynamics. *i\_stData.stSuperImposed.lfMaxVelocity*, *i\_stData.stSuperImposed.lfMaxAcceleration* and *i\_stData.stSuperImposed.lfMaxDeceleration* are to be considered in addition to the actual speed and acceleration of the axis.

As per other functions, *q\_xCmdDone* and *q\_xCmdBusy* can be evaluated, and will refer to the SuperImposed function (not to the underlying movement).

#### NOTE

- When using jerk limited trajectories (i.e.: *Velocity ramp type* ≠ *Trapezoid*), the maximum Jerk of the movement is always equal to *i\_stData.stSuperImposed.lfMaxDeceleration* x 10.



### 5.3.2.2 Jog function (AxisMgr\_SM)

Inputs	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
<i>i_xJogModeActivate</i> <i>i_xJogForward</i> <i>i_xJogBackward</i>	v. 3.1.2.0	<i>Standstill</i> <i>Jog</i>	<i>Jog</i>	<i>Standstill</i>	<a href="#">AxisMgr_SM</a>	<a href="#">AM_JogDataType</a>

Jog function is designed for “manual” operations. From *Standstill*, the axis is controlled in *Jog* as long as *i\_xJogModeActivate* is TRUE.

When in *Jog*, *i\_xJogForward* and *i\_xJogBackward* are evaluated. The axis will move with the specified *i\_stData.stJog.lrJogVelocity*, *i\_stData.stJog.lrJogAcceleration* and *i\_stData.stJog.lrJogDeceleration* depending on the following conditions:

<i>i_xJogModeActivate</i>	<i>i_xJogForward</i>	<i>i_xJogBackward</i>	AxisMgr state transition	Behavior description
FALSE	X	X	<i>Standstill</i>	No change
TRUE	FALSE	FALSE	<i>Standstill</i> => <i>Jog</i>	Axis decelerates to 0 speed with the specified deceleration. From <i>Standstill</i> , axis reaches <i>Jog</i> state.
TRUE	TRUE	TRUE	<i>Standstill</i> => <i>Jog</i>	Axis decelerates to 0 speed with the specified deceleration. From <i>Standstill</i> , axis reaches <i>Jog</i> state.
TRUE	TRUE	FALSE	<i>Standstill</i> => <i>Jog</i>	Axis moves endless in positive direction with the specified target values.
TRUE	FALSE	TRUE	<i>Standstill</i> => <i>Jog</i>	Axis moves endless in negative direction with the specified target values.
FALSE	X	X	<i>Jog</i> => <i>Standstill</i>	Axis decelerates to 0 speed with the specified deceleration. From <i>Jog</i> , axis became <i>Standstill</i> once the axis has reached 0 speed.



*i\_stData.stJog.lrJogVelocity* can eventually be set with negative values. In this case the action of *i\_xJogForward* and *i\_xJogBackward* will be reverted. Therefore, it is possible to use one direction input only, using the sign of velocity for the direction control.

#### NOTE

- ➔ Input data are constantly evaluated. Any change during the execution of the function will immediately take effect.
- ➔ *q\_xCmdBusy* and *q\_xCmdDone* are not used.
- ➔ When using jerk limited trajectories (i.e.: *Velocity ramp type* ≠ *Trapezoid*), the maximum Jerk of the movement is always equal to *i\_stData.stJog.lrJogDeceleration* x 10.



### 5.3.2.3 Set Controller Mode function (AxisMgr\_SM)

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
100	v. 3.1.2.0	<i>Disabled</i>			<a href="#">AxisMgr_SM</a>	AM_ControllerMode (4.5)

Thanks to this function, the NC axis can switch between CSP (initial), CSV and CST mode.

As soon as the functions has been triggered by means of *i\_iCmd* = 100 and a rising edge given to *i\_xLaunchCmd*, the function switches mode of operation according to what it is selected in *i\_stData.enControllerMode*. The function can be triggered in *Disabled* state only and it will affect the movement that will be performed afterwards. The actual mode is displayed by *q\_eModeOfOperation*.

Depending on the selected mode, the following considerations result:

- **CSP** (*i\_stData.enControllerMode* = *SMC\_position*): the drive follows the position trajectory sent by the controller. Object *0x607A Target Position* is mandatory and must be mapped.
- **CSV** (*i\_stData.enControllerMode* = *SMC\_velocity*): the drive follows the velocity trajectory sent by the controller. Object *0x60FF Target Velocity* is mandatory and must be mapped. CODESYS SoftMotion driver interface calculates the velocity trajectory together with the position when using the standard MC blocks. This means that the function that are normally used with CSP can eventually work in CSV (such as Positioning), provided that the position accuracy is not guaranteed.
- **CST** (*i\_stData.enControllerMode* = *SMC\_torque*): the drive follows a torque trajectory sent by the controller. Object *0x6071 Target Torque* is mandatory and must be mapped. In torque control, the speed of the motor is a consequence of the the torque that the motor is generating, referring to the target torque. In this mode, the user is supposed to provide a continuous torque value over time with the help of the Set Torque (5.3.2.4) function.

#### NOTE

- Not all the target modes are supported by all the drives (e.g.: CST can only work with AXIA drive series, otherwise an error is reported)



### 5.3.2.4 Set Torque function (AxisMgr\_SM)

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
110	v. 3.1.2.0	<i>Standstill Torque</i>	<i>Torque</i>		<a href="#">AxisMgr_SM</a>	<a href="#">AM_SetTorqueDataType</a>

When in CST mode (see: [Set Controller Mode function \(AxisMgr\\_SM\)](#)), this function can be used to update the torque reference sent to the drive cyclically.

As soon as the function has been triggered by means of *i\_iCmd* = 110 and a rising edge given to *i\_xLaunchCmd*, the function begins its execution. Axis Manager state is changed to *Torque*.

The value set in *i\_stData.stSetTorque.lrTorque\_Nm* [in Nm] is delivered to the drive without applying any ramp between two task cycles. The user should provide a continuous torque value over time to avoid unwanted "jumps" in the control. A power off command should be executed in order to exit the function and stop the motor.

#### NOTE

- Input data are constantly evaluated. Any change during the execution of the function will immediately take effect.



### 5.3.3 AxisMgr “Light” supported functions

The following functions are supported by AxisMgr “Light” in addition to the basic ones (see chapter 5.3).

✓ = function supported, blank = function not supported

Function	<i>i_iCmd / Bool Input</i>	Description	Type of AxisMgr				
			CANOpen / EtherCAT				
			AxisMgr ACUx10 CAN ETC	AxisMgr ANG CAN ETC	AxisMgr AGL CAN ETC	AxisMgr AXV CAN ETC	AxisMgr AXM CAN ETC
Stop (5.3.3.1)	20	It stops any ongoing movement putting the axis in <i>Stopping</i> state. <i>i_stData.stStop</i> input data is relevant. End state depends on the actual Mode Of Operation.	✓	✓	✓	✓	✓
Homing (5.3.3.2)	40	It performs the homing procedure provided by the drive (CiA402 Mode Of Operation 6) putting the axis in <i>Homing</i> state. <i>i_stData.stHoming</i> input data is relevant. A successful Homing ends in <i>Standstill</i> state.	✓	✓		✓	✓
Profile Velocity (5.3.3.3)	50	The axis moves endless with the specified dynamic settings. It puts the axis in <i>Velocity</i> state. <i>i_stData.stProfileVelocity</i> input data is relevant. Mode Of Operation 3 is used.	✓	✓		✓	✓
Velocity Mode (5.3.3.4)	51	The axis moves endless with the specified dynamic settings. It puts the axis in <i>Velocity</i> state. <i>i_stData.stVelocityMode</i> input data is relevant. Mode Of Operation 2 is used.	✓	✓	✓	✓	✓
Profile Position (5.3.3.5)	60	The axis moves towards the specified target position (either absolute or relative), with the specified dynamic settings. It puts the axis in <i>Position</i> state. <i>i_stData.stProfilePos</i> input data is relevant. Mode Of Operation 1 is used	✓	✓		✓	✓
Profile Torque (5.3.3.6)	110	The axis speeds up until it reaches the specified target torque, with the specified torque slope. It puts the axis in <i>Torque</i> state. <i>i_stData.stTorque</i> input data is relevant. Mode Of Operation 4 is used				✓	✓
Touch Probe (5.3.4.1)	See description	As long as it is enabled (i.e.: <i>i_stData.stTouchProbe.stTPx.xTouchProbeEnable</i> = <i>TRUE</i> ), AxisMgr reads out the related TouchProbe channel information.	✓	✓		✓	✓



### 5.3.3.1 Stop function (AxisMgr "Light")

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
20	v. 3.1.2.0	<i>Position</i> <i>Homing</i> <i>Stopping</i> <i>Synchronized</i> <i>Velocity</i> <i>Standstill</i>	<i>Stopping</i>	<i>Standstill</i> <i>Disabled</i>	All AxisMgr "Light"	<a href="#">AM_StopDataType</a> <b>Errore. L'origine riferimento non è stata trovata.</b>

The Stop function stops any ongoing movement putting the axis in *Stopping* state. *i\_stData.stStop* input data is relevant.

Depending on the actual Mode of Operation, Stop function behaves accordingly:

- **2 - Velocity mode** (i.e.: Velocity Mode function was running): As soon as Stop function was triggered by means of *i\_iCmd* = 20 and a rising edge given to *i\_xLaunchCmd*, the function writes the target speed to 0, then Axis Mgr reaches *Stopping* state. The axis decelerates with the specified *i\_stData.stStop.lrdDeceleration* [*vel. units/s*] to 0 speed. Stop function ends its execution once the drive disables the motor power, so Axis Mgr reaches *Disabled* state.

#### NOTE

- The end of Stop function when Velocity Mode was running is affected by the setting made in *i\_stData.stVelocityMode.rVelocityWindow* and in *i\_stData.stVelocityMode.uiVelocityWindowTime\_ms*. These values determine when the axis has to be considered at 0 speed. Axis Mgr disable the operation of the drive when the actual speed lies in the *Velocity Window* over the *Velocity Window Time*. Value set in *obj. 0x605C Disable Operation Option Code* is also relevant.
- At the end of the Stop function (after a running Velocity Mode), the drive may keep the motor powered and standstill until the *Holding time* has been expired (drive's internal parameter, refer to the drive documentation).

- **4 - Profile Torque mode** (i.e.: Profile Torque function was running): same behavior as per Velocity mode.

- **3 - Profile Velocity mode** (i.e.: Profile Velocity function was running): As soon as Stop function was triggered by means of *i\_iCmd* = 20 and a rising edge given to *i\_xLaunchCmd*, the function writes the target speed to 0, then AxisMgr reaches *Stopping* state. The axis decelerates with the specified *i\_stData.stStop.lrdDeceleration* (which is expressed in *pos. units/s<sup>2</sup>*) to 0 speed. Stop function ends its execution once the drive reaches 0 speed, so Axis Mgr reaches *Standstill* state.

#### NOTE

- The end of Stop function when Profile Velocity was running is affected by the setting made in *i\_stData.stProfileVelocity.rVelocityWindow* and in *i\_stData.stProfileVelocity.uiVelocityWindowTime\_ms*. These values determine when the axis has to be considered at 0 speed. *Standstill* state, as well as *q\_xCmdDone* signal, occur when the actual speed lies in the *Velocity Window* over the *Velocity Window Time*.

- **1 - Profile Position mode** (i.e.: Profile Position function was running): As soon as Stop function was triggered by means of *i\_iCmd* = 20 and a rising edge given to *i\_xLaunchCmd*, the function begins its execution and Axis Mgr reaches *Stopping* state. AxisMgr halts the drive and the axis decelerates with the specified *i\_stData.stStop.lrdDeceleration* (which is expressed in *pos. units/s<sup>2</sup>*) to



0 speed. Stop function ends its execution once the drive reaches 0 speed, so Axis Mgr reaches *Standstill* state.

#### NOTE

- Behavior of the drive when Stop function has been triggered during Profile Position Mode function may depend on the value set in *obj. 0x605D Halt Option Code* (for Axia series). Value of *i\_stData.stStop.lDeceleration* is only relevant when this object is set to 1.

- **6 - Homing mode** (i.e.: Homing function was running): As soon as Stop function was triggered by means of *i\_iCmd = 20* and a rising edge given to *i\_xLaunchCmd*, the function begins its execution and Axis Mgr reaches *Stopping* state. AxisMgr halts the drive and the axis decelerates to 0 speed. In this case *i\_stData.stStop.lDeceleration* is not relevant, instead the axis will decelerate with the value defined in *i\_stData.stHoming.lAcceleration*. Stop function ends its execution once the drive reaches 0 speed, so Axis Mgr reaches *Standstill* state.

#### NOTE

- The deceleration ramp used by the Stop function, when during an Homing procedure, is the value of *i\_stData.stHoming.lAcceleration* when Homing began its execution. Changing this value afterwards won't have any effect.
- Behavior of the drive when Stop function has been triggered during Homing Mode function may depend on the value set in *obj. 0x605D Halt Option Code* (for Axia series). Value of *i\_stData.stHoming.lAcceleration* is only relevant when this object is set to 1.

In the case of invalid input data (e.g.: *i\_stData.stStop.lDeceleration*  $\leq 0$ ) the function will be triggered anyway and the last deceleration value stored in the drive will be used. The deceleration value specified in the actual running function will be probably used.

#### NOTE

- Input data are constantly evaluated. Any change during the execution of the function will take effect.
- Deceleration value are probably written into the drive via **acyclic requests**. In this case, the start of the execution of Stop function is delayed by the time taken for the data to be written. This must be considered in time-critical situation; consider to trigger a Quick Stop instead.



### 5.3.3.2 Homing function (AxisMgr "Light")

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
40	v. 3.1.2.0	<i>Standstill</i> <i>Homing</i>	<i>Homing</i>	<i>Standstill</i>	<a href="#">AxisMgr ANG CAN ETC</a> <a href="#">AxisMgr ACUx10 CAN ETC</a> <a href="#">AxisMgr AXV CAN ETC</a> <a href="#">AxisMgr AXM CAN ETC</a>	<b>Errore. L'origine riferimento non è stata trovata.</b> <a href="#">AM_HomingMethodsEnum</a>

Homing function triggers the drive's homing procedure (CiA DS402 Mode Of Operation 6) and puts the AxisMgr in *Homing* state. *i\_stData.stHoming* input data is relevant. A successful Homing ends in *Standstill* state.

As soon as the functions has been triggered by means of *i\_iCmd* = 40 and a rising edge given to *i\_xLaunchCmd*, AxisMgr requests the Homing mode of operation.

Once the drive reaches Homing mode, before to start the movement, AxisMgr will write all the necessary Homing objects to the drive. Depending on the selected *i\_stData.stHoming.eMode*, a proper parametrization at the drive side is probably needed (i.e.: Home sensor signal, Pos/Neg HW limit switches...) and the home signal is evaluated accordingly. As soon as the home signal has been detected by the drive, the position set in *i\_stData.stHoming.lrFinalPosition* will take effect and the axis will stop the movement.

#### NOTE

- *q\_xHomed* will reset as soon as the function gets busy. It will be set when the function is successfully finished.
- Unlike other functions, *i\_stData.stHoming* is only evaluated at the beginning of the function. A value change, while the function is already running, won't have any effect.



### 5.3.3.3 Profile Velocity function (AxisMgr "Light")

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
50	v. 3.1.2.0	<i>Standstill</i> <i>Stopping</i> <i>Velocity</i> <i>Position</i> <i>Synchr.</i>	<i>Velocity</i>		<a href="#">AxisMgr ANG CAN ETC</a> <a href="#">AxisMgr ACUx10 CAN ETC</a> <a href="#">AxisMgr AXV CAN ETC</a> <a href="#">AxisMgr AXM CAN ETC</a>	AM_ProfileVelocity DataType_<inverter>

The axis moves endless with the specified dynamic settings. It puts the axis in *Velocity* state. *i\_stData.stProfileVelocity* input data is relevant.

As soon as the functions has been triggered by means of *i\_iCmd* = 50 and a rising edge given to *i\_xLaunchCmd*, AxisMgr writes the set values in the proper drive objects, then puts the drive in Modes Of Operation 3 – Profile Velocity Mode. Once the drive reaches Profile Velocity Mode, the function begins its execution.

A negative value set in *i\_stData.stProfileVelocity.rTargetVelocity* (expressed in *pos. units/s*) results in a negative direction.

#### NOTE

- ➔ Input data are constantly evaluated. Any change during the execution of the function will take effect.

#### **q\_xInVel**

*q\_xInVel* signals wheter the axis speed lays in the defined *i\_stData.stProfileVelocity.rVelocityWindow* for the time defined in *i\_stData.stProfileVelocity.uiVelocityWindowTime\_ms*. The Velocity window is considered around the target velocity that the axis should reach (*Target Velocity +/- (Velocity Window / 2)*).



### 5.3.3.4 Velocity Mode function (AxisMgr "Light")

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
51	v. 3.1.2.0	<i>Disabled Standstill Stopping Velocity</i>	<i>Velocity</i>		All AxisMgr "Light"	AM_VelocityMode DataType_<inverter>

The axis moves endless with the specified dynamic settings. It puts the axis in *Velocity* state. *i\_stData.stVelocityMode* input data is relevant.

As soon as the functions has been triggered by means of *i\_iCmd* = 51 and a rising edge given to *i\_xLaunchCmd*, AxisMgr writes the set values in the proper drive objects, then puts the drive in Modes Of Operation 2 – Velocity Mode. Once the drive reaches Velocity Mode, the function begins its execution.

The function is designed to be started from *Disabled* state. The user does not have to set *i\_xPower* in order to enable the power stage, but the function itself will power the axis during the starting phase. Nevertheless, *i\_xPower* can be used in order to disable the axis power by means of a falling edge.

A negative value set in *i\_stData.stVelocityMode.rTargetVelocity* (expressed in *vel. units*) results in a negative direction.

#### NOTE

- ➔ Input data are constantly evaluated. Any change during the execution of the function will take effect.

#### q\_xInVel

*q\_xInVel* signals wheter the axis speed lays in the defined *i\_stData.stVelocityMode.rVelocityWindow* for the time defined in *i\_stData.stVelocityMode.uiVelocityWindowTime\_ms*. The Velocity window is considered around the target velocity that the axis should reach (*Target Velocity +/- (Velocity Window / 2)*).



### 5.3.3.5 Profile Position function (AxisMgr "Light")

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
60	v. 3.1.2.0	<i>Standstill</i> <i>Stopping</i> <i>Velocity</i> <i>Position</i> <i>Synchr.</i>	<i>Position</i>	<i>Standstill</i>	<a href="#">AxisMgr ANG CAN ETC</a> <a href="#">AxisMgr ACUx10 CAN ETC</a> <a href="#">AxisMgr AXV CAN ETC</a> <a href="#">AxisMgr AXM CAN ETC</a>	AM_ProfilePosition DataType_<inverter>  <a href="#">AM_PosTypeEnum</a> <a href="#">AM_ChangeProfileBehaviorEnum</a> <a href="#">AM_EndOfPositionBehaviorEnum</a>

The axis moves towards the specified target position (either absolute or relative), with the specified dynamic settings. It puts the axis in *Position* state. *i\_stData.stProfilePos* input data is relevant.

As soon as the functions has been triggered by means of *i\_iCmd* = 60 and a rising edge given to *i\_xLaunchCmd*, AxisMgr writes the set values in the proper drive objects, then puts the drive in Modes Of Operation 1 – Profile Position Mode. Once the drive reaches Profile Position Mode, the function begins its execution.

Depending on the positioning type (Absolute or Relative) the *i\_stData.stProfilePos.rTargetPosition* [u] defines the absolute quota to reach (referred to the last home position) or the distance to cover.

#### NOTE

- Input data are constantly evaluated. Any change during the execution of the function will take effect.

#### eChangeProfileBehavior

The selection made in *i\_stData.stProfilePos.eChangeProfileBehavior* affects the way that the target values are considered when updated:

- *i\_stData.stProfilePos.eChangeProfileBehavior* = *Immediately*: as soon as target position, velocity, acceleration and/or deceleration are updated during an ongoing positioning, the function writes the information into the drive's object. When all the information are written successfully, a new positioning command is triggered and the new target values will be applied.

#### NOTE

- Acceleration and Deceleration values are probably sent via **acyclic requests**, it may take some time for the data to be updated and for the new positioning to be requested.
  - When using relative movement, change of target values won't have any effect. A new rising edge of *i\_xLaunchCmd* should be used to trigger a new positioning during the ongoing one.
- 
- *i\_stData.stProfilePos.eChangeProfileBehavior* = *Immediate\_OnSetPoint*: as soon as target position, velocity, acceleration and/or deceleration are updated during an ongoing positioning, the function writes the information into the drive's object. The new target values will be applied only once the previous movement reaches the target. The axis **won't stop** on the target but it will maintain the same operating speed, then the new profile position will be applied immediately "on-the-fly".



#### NOTE

- Acceleration and Deceleration values are probably sent via **acyclic requests**, it may take some time for the data to be updated and to request a change profile.
- When using relative movement, change of target values won't have any effect. A new rising edge of *i\_xLaunchCmd* should be used to trigger a new positioning during the ongoing one.
- If more than one values should be applied for the next positioning movement, they must be updated all together before the execution of AxisMgr. Only one profile update is possible during the ongoing position.

→ *i\_stData.stProfilePos.eChangeProfileBehavior = Stop\_OnSetPoint*: as soon as target position, velocity, acceleration and/or deceleration are updated during an ongoing positioning, the function writes the information into the drive's object. The new target values will be applied only once the previous movement reaches the target. The axis **will stop** on the target, then the new profile position will immediately start from 0 speed.

#### NOTE

- Acceleration and Deceleration values are probably sent via **acyclic requests**, it may take some time for the data to be updated and to request a change profile.
- When using relative movement, change of target values won't have any effect. A new rising edge of *i\_xLaunchCmd* should be used to trigger a new positioning during the ongoing one.
- If more than one values should be applied for the next positioning movement, they must be updated all together before the execution of AxisMgr. Only one profile update is possible during the ongoing position.

### q\_xInPos and q\_xCmdDone

The end of the positioning movement is displayed in *q\_xCmdDone* and *q\_xInPos*. Both *q\_xCmdDone* and *q\_xInPos* are affected by the values set in *i\_stData.stProfilePos.rPositionWindow [u]* and *i\_stData.stProfilePos.uiPositionWindowTime\_ms [ms]*. If a value bigger than 0 is set in *i\_stData.stProfilePos.rPositionWindow*, then *i\_stData.stProfilePos.rPositionWindow* and *i\_stData.stProfilePos.uiPositionWindowTime\_ms* will be transferred to drive's *obj. 0x6067 Position window* and *obj. 0x6068 Position window time* which will determine when the axis has reached the target position. If *i\_stData.stProfilePos.rPositionWindow* is not set, the value stored in *obj. 0x6067* and *obj. 0x6068* will have effect.

Unlike *q\_xCmdDone*, *q\_xInPos* will monitor the axis position in the target window as long as the drive remains in Profile Position Mode. If it leaves this mode of operation (due to a different function triggered) or a new Position command has been started, then *q\_xInPos* is reset.



If *i\_stData.stProfilePos.eChangeProfileBehavior = Stop\_OnSetPoint* or *i\_stData.stProfilePos.eChangeProfileBehavior = Immediate\_OnSetPoint*, a new positioning profile may be requested and it will be processed at the end of the ongoing one. In this case, *q\_xInPos* will signal (for just one task cycle) the end of the ongoing positioning and the start of the new one. By monitoring this trigger, another new positioning profile may be requested by updating input datas.



### enEndOfPositionBehavior

*i\_stData.stPos.enEndOfPositionBehavior* defines the behavior of AxisMgr when the actual positioning movement reaches its end:

- ➔ **GoInStandstill (default):** the axis ends the movement and reaches the *Standstill* state. *q\_xCmdDone* can be evaluated to establish the end of the positioning.
- ➔ **RemainInPosition:** the axis remains in *Position* state. In this way the user can update the target parameters (e.g.: *i\_stData.stProfilePos.rTargetPosition*) to perform new movements, avoiding to trigger *i\_xLaunchCmd* again.  
In this case *q\_xCmdDone* is not relevant and will not be set at each position done, *q\_xInPos* should be used instead.

### 5.3.3.6 Profile Torque function (AxisMgr "Light")

<i>i_iCmd</i>	Available as of	Supported state	Beginning state	Ending state	Supported by	Relevant Data Types
110	v. 3.1.2.0	<i>Standstill Torque</i>	<i>Torque</i>		<a href="#">AxisMgr AXV CAN ETC</a> <a href="#">AxisMgr AXM CAN ETC</a>	<a href="#">AM ProfileTorqueData Type Axia</a>

The drive energizes the motor in order to reach the target torque, with the specified torque slope. It puts the axis in *Torque* state. *i\_stData.stProfileTorque* input data is relevant.

As soon as the functions has been triggered by means of *i\_iCmd* = 110 and a rising edge given to *i\_xLaunchCmd*, AxisMgr writes the set values in the proper drive objects, then puts the drive in Modes Of Operation 4 – Profile Torque Mode. Once the drive reaches Profile Torque Mode, the function begins its execution.

A power off command should be executed in order to exit the function and stop the motor.

#### NOTE

- ➔ Input data are constantly evaluated. Any change during the execution of the function will take effect.



## 5.3.4 Common functions

### 5.3.4.1 TouchProbe function (AxisMgr SM and "Light")

Inputs	Available as of	Supported by	Relevant Data Types
<i>i_stData.stTouchProbe.stTPx.xTouchProbeEnable</i>	v. 3.1.2.0	<a href="#">AxisMgr SM</a> <a href="#">AxisMgr ANG CAN ETC</a> <a href="#">AxisMgr ACUx10 CAN ETC</a> <a href="#">AxisMgr AXV CAN ETC</a> <a href="#">AxisMgr AXM CAN ETC</a>	<a href="#">AM_TouchProbeDataType</a> <a href="#">AM_TPChannelType</a> <a href="#">AM_TouchProbeModeEnum</a> <a href="#">AM_TouchProbeStatusType</a>

TouchProbe function sets the special touch probe channels provided by the drive and gets the information from them. When enabled, the drive's TP channel detects the edges (both positive and negative) of the related physical digital input and stores the actual axis position in the lowest time, resulting in the most precise position capture. This is mostly needed when PLC cycle time is not fast enough to establish the exact axis position at a precise event.

Refer to the used drive documentation for further information about the touch probe function.

- ➔ **Real axis:** drive's TP channel will be used (*i\_stData.stTouchProbe.stTPx.xTP\_input* is not relevant). As soon as *i\_stData.stTouchProbe.stTPx.xTouchProbeEnable* is set to TRUE, AxisMgr will enable the monitoring of the related TP channel on the drive side.
- As long as the TPx is enabled, AxisMgr will continuously ask for the TP channel status to the drive (via acyclic data requests), at each *i\_stData.stTouchProbe.tRefreshTouchProbeStatusTimer* clock. Once the TP channel status signals an edge detection (either positive or negative), TouchProbe function will get the information about the axis position and the current edge counter (related to the last TP event) from the drive, then returns the information in *q\_stTouchProbeStatus.stTPx*. While doing this, depending on the detected edge, *q\_stTouchProbeStatus.stTPx.xPosEdgeValueValid* or *q\_stTouchProbeStatus.stTPx.xNegEdgeValueValid* are reset to FALSE and set to TRUE again once the information has been successfully get from the drive. Therefore, the user program can monitor the positive edge of these two signals in order to know whether a TP event was detected and to know if information are ready to be evaluated.

#### NOTE

##### TouchProbe function limitations, Real axis

- ➔ Enabling of TP channel, as well as mode (either continuous or single shot), is carried out via acyclic data (see chapter [5.2.4](#)) as soon as *i\_stData.stTouchProbe.stTPx.xTouchProbeEnable* is set to TRUE. If one (or more) TP event occurs during the TP enabling, it may not be detected. Be sure to enable the function far in advance.
- ➔ Requests of TP channel status and information are both made via acyclic requests (see chapter [5.2.4](#)). For this reason *i\_stData.stTouchProbe.tRefreshTouchProbeStatusTimer* is provided in order to delay the TP status requests. The lower this value is, the more often TP informations are updated.  
Nevertheless, the total time span from the physical TP event to the information updated is not always predictable, since it is also affected by the PLC cycle tasks, by the drive responding time and by the amount of acyclic data that is being requested at the same time.  
This means that, if more TP events occur during this time, the information provided may not be related to a known TP event.  
TouchProbe function is not intended for fast and cyclic events, but rather for very precise position evaluation of time-distant position captures.
- ➔ Since TP channel inputs of the drive are usually not affected by any filter time, electromechanical sensor should not be used for this purpose. Additional signal edges (due to bouncing effects) may be detected by the drive, which overwrites Pos and Neg edge positions.
- ➔ TouchProbe function can only work with Position control configurations of the drive.



- Position informations are expressed in user-units. It means that the drive's increment conversion is already carried out by the function itself, as well as modulo calculation (for modulo axis).
- *q\_stTouchProbeStatus.stTPx.uiPosEdgeCounter* and *q\_stTouchProbeStatus.stTPx.uiNegEdgeCounter* are stored in the non-volatile memory of the drive. A reset/reboot of the PLC won't restore those values. Instead, a drive reset may be necessary.

- **Virtual/simulation axis (only for AxisMgr SM):** Touch probe function is simulated and replicates the behavior that a physical drive would have.
- The function monitors *i\_stData.stTouchProbe.stTPx.xTP\_input* (at each PLC task cycle) as long as *i\_stData.stTouchProbe.stTPx.xTouchProbeEnable* is set to TRUE.
- Once detected, depending on the edge, *q\_stTouchProbeStatus.xPosEdgeValueValid* or *q\_stTouchProbeStatus.xNegEdgeValueValid* are reset to FALSE and set to TRUE the next task cycle, and returns the information in *q\_stTouchProbeStatus*. Therefore, the user program can monitor the positive edge of these two signals in order to know whether a TP event was detected and to evaluate the related information (the same logic can be used as per Real Axis TP).

#### NOTE

##### **TouchProbe function limitations, Virtual/simulation axis**

- Unlike Real Axis TP, monitor of *xTP\_input* is made by the PLC each task cycle. Position information may be affected by an error which depends on task cycle time and axis speed. Position capture is not precise as per Real Axis.

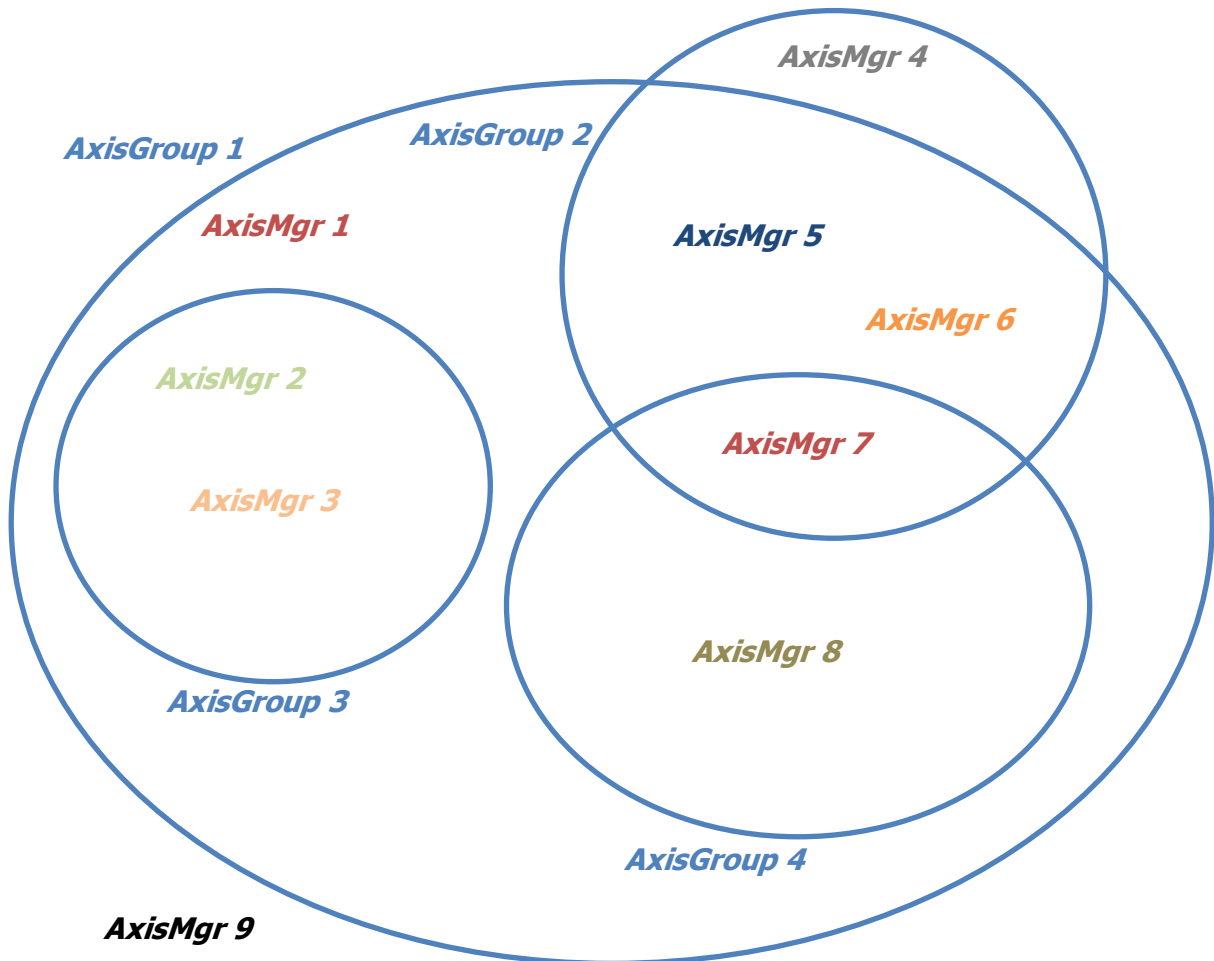
TouchProbe function is not affected by the AxisMgr current state and it will always work in background. Nevertheless, *Init* must be performed (*q\_xEnabled* => *TRUE*) before to be used.



## 6 Overview of Axis Manager Groups

As mentioned in chapter 5, the Axis Managers Groups are designed to control several Axis Managers and to collect diagnostic from them all in one single point. It is possible to create several logical group of axes based on the application or, in some occasions, to have just one group that comprehends all the axes defined in the software.

The picture below shows a possible configuration that can be created to group Axis Managers.



Based on the picture, the following considerations result:

- AxisMgr 1 is only controlled by AxisGroup 1;
- AxisMgr 2 and 3 are both controlled by AxisGroup 1 and 3;
- AxisMgr 4 is only controlled by AxisGroup 2;
- AxisMgr 5 and 6 are both controlled by AxisGroup 1 and 2;
- AxisMgr 7 is controlled by AxisGroup 1, 2 and 4;
- AxisMgr 8 is only controlled by AxisGroup 1 and 4;
- AxisMgr 9 is not controlled by any group.



The output signals of the group (e.g.: *q\_xPowered*, *q\_xHomed*, ...) follow this logic:

- ➔ Those output which are "not-faulty" are set to TRUE only if all the axes added to the group is in that condition. For example, *q\_xPowered* is set only when all the axes of the group are powered.
- ➔ Those output which represents a "faulty" condition (*q\_xFault*, *q\_xWarning*, ...) are set to TRUE when at least one of the axes of the group raise that condition.
- ➔ The axes that do not support some condition (i.e. *q\_xHomed* is not present in *AxisMgr\_AGL\_<fieldbus>*) are bypassed by the group. Means that if in the same group there are both [AxisMgr\\_SM](#) and *AxisMgr\_AGL\_<fieldbus>*, the *q\_xHomed* output is set when all the [AxisMgr\\_SM](#) are homed, without considering *AxisMgr\_AGL\_<fieldbus>*
- ➔ If one or more axes are disabled in the device tree of the project, then they are bypassed by the group, even if they are added. This is made in order to keep the same logic in the software even if one axis has been disabled for some reason (e.g. is physically missing).
- ➔ The diagnostic collected attains only for those axes that has been added to the group and the fault message is filled with the name of the instance of the AxisMgr which reported the fault.

Special application-oriented AxisMgr Groups has been developed. These Axis Groups follow the same logic described before, but there is a limitation about the possibility to add as many axes as the user wants. For example in *AxisMgr\_Group\_CN\_xx*, axes are limited depending on the type of kinematic, also the role of the axis is defined.

Thanks to Groups, special command can be triggered. The same command is not available using the axis as a single. It is possible that the group will bring the axis to dedicated state during the group command. Additional signal, which is related to the application, has been added.



## 7 Program Organization Units (POU)

This part contains the following chapters:

Chapter	Chapter Name	Page
<a href="#">7.1</a>	<a href="#">Function Blocks</a>	<a href="#">60</a>
<a href="#">7.2</a>	<a href="#">Functions</a>	<a href="#">88</a>



## 7.1 Function Blocks

This chapter contains the following topics:

Chapter	Chapter Name	Page
<a href="#">7.1.1</a>	<a href="#">AxisMgr_SM</a>	<a href="#">61</a>
<a href="#">7.1.2</a>	<a href="#">AxisMgr_ANG_CAN_ETC</a>	<a href="#">64</a>
<a href="#">7.1.3</a>	<a href="#">AxisMgr_AGL_CAN_ETC</a>	<a href="#">68</a>
<a href="#">7.1.4</a>	<a href="#">AxisMgr_ACUx10_CAN_ETC</a>	<a href="#">71</a>
<a href="#">7.1.5</a>	<a href="#">AxisMgr_AXV_CAN_ETC</a>	<a href="#">75</a>
<a href="#">7.1.6</a>	<a href="#">AxisMgr_AXM_CAN_ETC</a>	<a href="#">79</a>
<a href="#">7.1.7</a>	<a href="#">AxisMgr_Group</a>	<a href="#">83</a>
<a href="#">7.1.8</a>	<a href="#">FB_GenericAcces_CAN_ETC</a>	<a href="#">86</a>

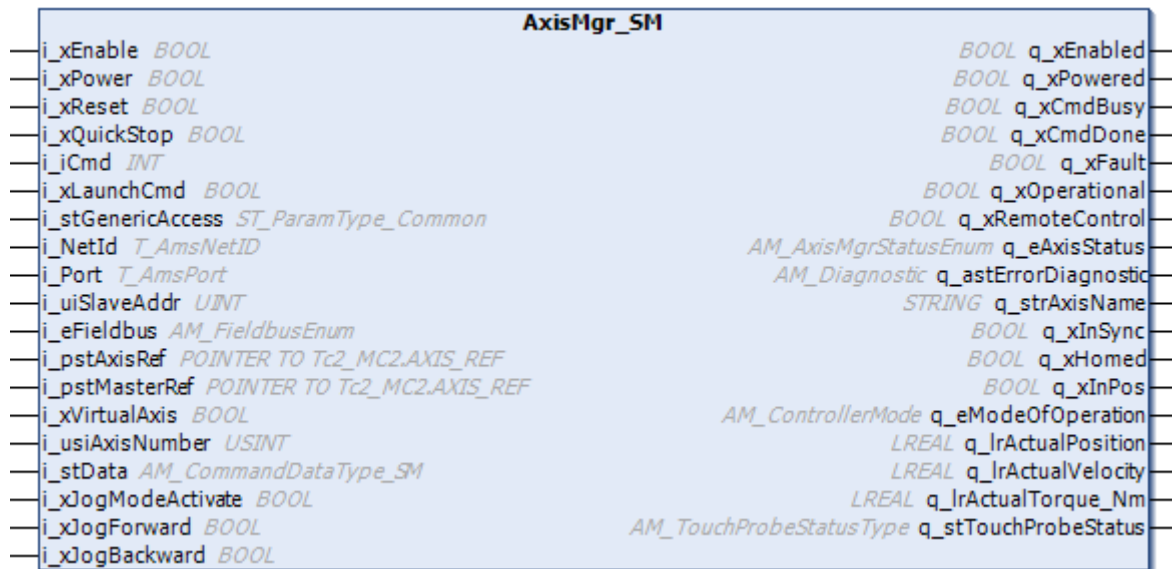


## 7.1.1 AxisMgr\_SM

### Overview

Type:	Function Block
Available as of:	v. 3.1.2.0

### FUNCTION\_BLOCK AxisMgr\_SM EXTENDS AxisMgr\_CAN\_ETC



### Description

This FB is developed to work mainly with CiA402 Cyclic Synchronous Position Mode 8. This manager must be used in combination with TwinCAT NC.

**AxisMgr\_SM** uses PLCOpen Motion blocks in its inside, giving the right sequence of command to them and collecting the diagnostic in one single point. Please refer to PLCOpen and Tc2\_MC2 documentations for further informations.

It is suitable for real and virtual axes. In both cases, the target controller should have real-time capabilities. In case of real axis, motion control trajectories are computed by the motion controller, then sent point-to-point to the drive at each cycle time. The controller that runs this kind of FB needs a real-time system and a deterministic fieldbus as well. Those devices which do not satisfy these requirements may fail. **AxisMgr\_SM** is therefore essential when it comes to synchronize multiple axes together (e.g.: Gearing, Camming...), in order to perform more complex movements.

**AxisMgr\_SM** only works in combination with Bonfiglioli's drive. The drive recognition, as well as the axis number (i.e.: Axis Move Double Axis), is automatically carried out during *Init*.

**AxisMgr\_SM** is based on the base architecture described in chapter 5. The base interface (chapter 5.2.2) won't be described in this chapter.

The supported function are listed in chapter 5.3.2 and described in the following. This chapter won't give further information.

Fieldbus management (cyclic and acyclic datas), as well as specific precautions, has been described in chapter 5.2.4.1.



## NOTE

- Functions of [AxisMgr\\_SM](#) that uses SoftMotion FBs are set to work in BufferMode = Aborting.



## CAUTION

### MASTER-FOLLOWER operation

Instances of AxisMgr\_SM that are supposed to work in *Synchronize* motion, should be always called (in the program task) **after** the AxisMgr\_SM instance that represent the master axis. The non-compliance of this rule may lead to unwanted behavior due to positions lags.

## Interface

In addition to the base interface (chapter [5.2.2](#)), the [AxisMgr\\_SM](#)'s specific In/Out are listed below.

Input	Data Type	Initial	Description
i_NetId	T_AmsNetId		Identification of the EtherCAT master network (only relevant when EtherCAT is used).
i_Port	T_AmsPort		Identification of the CANOpen port (only relevant when CANOpen is used).
i_uiSlaveAddr	UINT		Physical slave address (EtherCAT) or the device Node ID (CANOpen).
i_eFieldBus	AM_FieldbusEnum		Fieldbus selection (either CANOpen or EtherCAT).
i_pstAxisRef	AXIS_REF		Reference to the SoftMotion driver interface instance that has to be controlled by the FB. This input is mandatory and cannot be changed after <i>Init</i> procedure has been completed.
i_pstMasterRef	AXIS_REF		Reference to the master axis. Mandatory in <i>Synchronized</i> motion. It can eventually change during the execution.
i_xVirtualAxis	BOOL		TRUE: the axis is virtual and it is not linked to any physical device.
i_usiAxisNumber	USINT	1	In the case of an Axia Move Double Axis, it defines the axis number (either 1 or 2) that the AxisMgr should control.
i_stData	<a href="#">AM_CommandDataType_SM</a>		Structure of data relevant for the supported functions. Compile the data structures to configure the function parameters. Each structure is only relevant when using the related function.
i_xJogModeActivate	BOOL		Activation of <i>Jog</i> function. <i>Jog</i> mode can be activated only when in <i>Standstill</i> and remains active as long as this input remains TRUE.
i_xJogForward	BOOL		When in <i>Jog</i> function, move the axis in positive direction as long as is set to TRUE and <i>i_xJogBackward</i> is FALSE.
i_xJogBackward	BOOL		When in <i>Jog</i> function, move the axis in negative direction as long as is set to TRUE and <i>i_xJogForward</i> is FALSE.



Output	Data Type	Description
q_xInSync	BOOL	TRUE: the axis is engaged in a synchronized motion (e.g.: Gear, Cam...)
q_xHomed	BOOL	TRUE: The axis has been successfully homed. It will be reset at each new home request and set again when done.
q_xInPos	BOOL	TRUE: the axis has reached the target position after a <i>Positioning</i> command. This output is affected by the parameters <i>rPositionWindow</i> and <i>uiPositionWindowTime_ms</i> in <i>i_stData.stPos</i> . This output is reset as soon as a new <i>Positioning</i> command is triggered or if the axis leaves the <i>Standstill</i> state.
q_eModeOfOperation	AM_ControllerMode ( <a href="#">4.5</a> )	It returns the actual operating mode of the drive, which can be changed by the Set Controller mode function.
q_lrActualPosition	LREAL	It returns the actual position of the axis [u]
q_lrActualVelocity	LREAL	It returns the actual velocity of the axis [u/s]
q_lrActualTorque_Nm	LREAL	It returns the actual torque generated by the axis [Nm]. <i>Obj. 0x6077</i> must be mapped.
q_stTouchProbeStatus	<a href="#">AM_TouchProbeStatusType</a>	It contains the TouchProbe informations, when enabled (see <i>i_stData.stTouchProbe</i> )

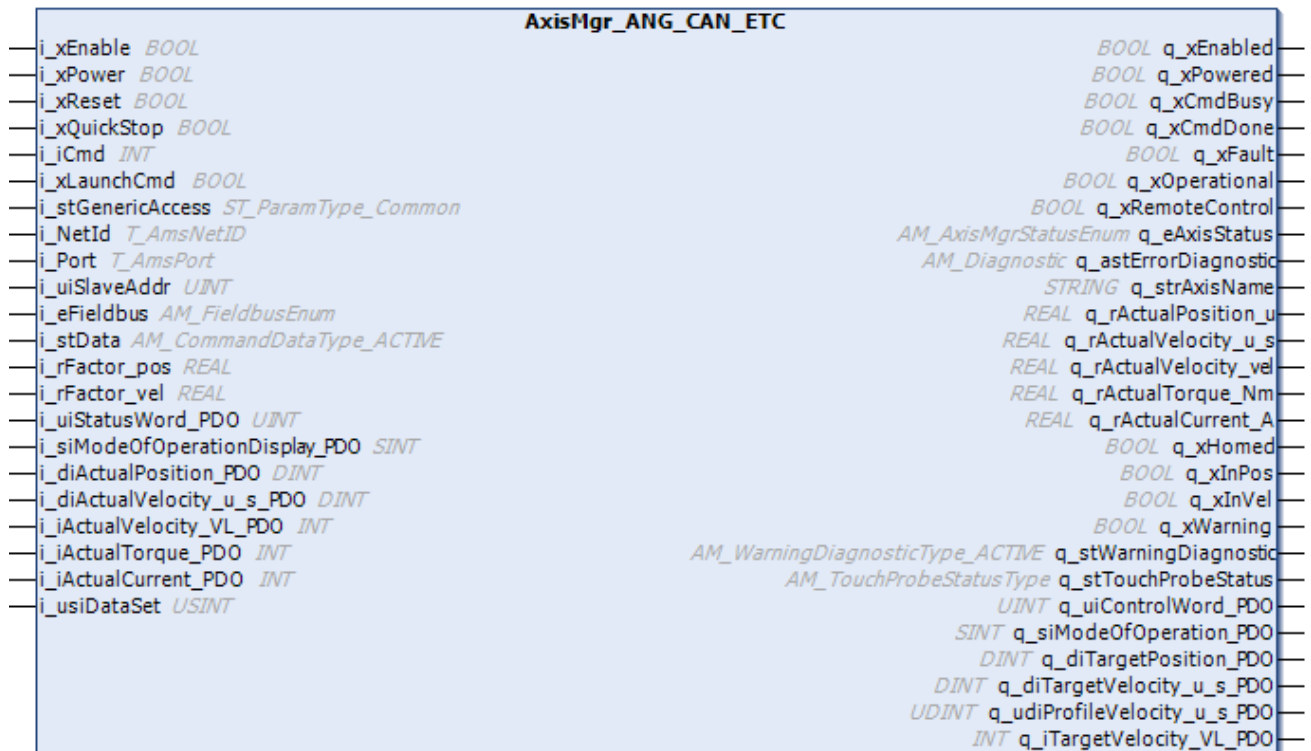


## 7.1.2 AxisMgr\_ANG\_CAN\_ETC

### Overview

Type:	Function Block
Available as of:	v. 3.1.2.0

### FUNCTION\_BLOCK AxisMgr\_ANG\_CAN\_ETC EXTENDS AxisMgr\_CAN\_ETC



### Description

This FB allows the user to easily manage ANG drive series either in CANOpen or in EtherCAT. The motion profiles defined by the CiA402 specifications has been implemented, in combination with the manufacturer specific features. Unlike [AxisMgr\\_SM](#), the controller does not calculate and send the trajectory to the drive, but the drive's motion profile generator is used. This also means that the controller is not intended to have demanding real-time capabilities, since the control of the drive may not be deterministic.

[AxisMgr\\_ANG\\_CAN\\_ETC](#) only works in combination with ANG drive. The FB is designed to work in State Machine control only (*P.412 = 1 - State Machine*). Depending on the actual drive configuration (*P.30 Configuration*), some functions may not be supported.

[AxisMgr\\_ANG\\_CAN\\_ETC](#) is based on the base architecture described in chapter 5. The base interface (chapter 5.2.2) won't be described in this chapter.

The supported function are listed in chapter 5.3.3 and described in the following. This chapter won't give further information.

Fieldbus management (cyclic and acyclic datas), as well as specific precautions, has been described in chapter 5.2.4.1.



## Interface

In addition to the base interface (chapter [5.2.2](#)), the AxisMgr ANG CAN ETC's specific In/Out are listed below.

Input	Data Type	Initial	Description
i_NetId	T_AmsNetId		Identification of the EtherCAT master network (only relevant when EtherCAT is used).
i_Port	T_AmsPort		Identification of the CANOpen port (only relevant when CANOpen is used).
i_uiSlaveAddr	UINT		Physical slave address (EtherCAT) or the device Node ID (CANOpen).
i_eFieldBus	AM_FieldbusEnum		Fieldbus selection (either CANOpen or EtherCAT).
i_stData	<a href="#">AM_CommandDataType</a> <a href="#">ACTIVE</a>		Structure of data relevant for the supported functions. Compile the data structures to configure the function parameters. Each structure is only relevant when using the related function.
i_rFactor_pos	REAL	65536 / 360.0	Conversion factor [increments/units]. It defines the amount of the drive's increment that represent 1u displacement in the user program. This value is relevant for the position-controlled functions (Profile Position, Profile Velocity, Homing, ecc...). See chapter <a href="#">5.3.1.2</a> .
i_rFactor_vel	REAL	1	Conversion factor [VL Units/vel. units]. It defines the amount of drive's VL units that represent 1u speed in the user program. This value is relevant for the speed-controlled functions (Velocity Mode). See chapter <a href="#">5.3.1.2</a> .
i_uiStatusWord_PDO	UINT		Cyclic information for drive's status word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6041</i> ). This object is always mandatory.
i_siModeOfOperationDisplay_PDO	SINT		Cyclic information for drive's mode of operation display. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6061</i> ). This object is always mandatory.
i_diActualPosition_PDO	DINT		Cyclic information for drive's actual position. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6064</i> ).



i_diActualVelocity_u_s_PDO	DINT		Cyclic information for drive's actual velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x606C</i> ).
i_iActualVelocity_VL_PDO	INT		Cyclic information for drive's actual velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6044</i> ).
i_iActualTorque_PDO	INT		Cyclic information for drive's actual torque [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6077</i> ).
i_iActualCurrent_PDO	INT		Cyclic information for drive's actual current [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6078</i> ).
i_usiDataSet	USINT		Not implemented yet

Output	Data Type	Description
q_rActualPosition_u	REAL	Actual position of the axis [u], referred to the last home referencing. The value is updated only if <i>i_diActualPosition_PDO</i> has been provided, and it is scaled by <i>i_rFactorPos</i> .
q_rActualVelocity_u_s	REAL	Actual velocity of the axis [u/s]. The value is updated only if <i>i_diActualVelocity_u_s_PDO</i> has been provided, and it is scaled by <i>i_rFactorPos</i> .
q_rActualVelocity_vel	REAL	Actual velocity of the axis [vel. units]. The value is updated only if <i>i_iActualVelocity_VL_PDO</i> has been provided, and scaled by <i>i_rFactorVel</i> .
q_rActualTorque_Nm	REAL	Actual torque of the axis [Nm]. The value is updated only if <i>i_iActualTorque_PDO</i> has been provided, and it is scaled by the rated torque of the motor (read out during <i>Init</i> ).
q_rActualCurrent_A	REAL	Actual current of the axis [A]. The value is updated only if <i>i_iActualCurrent_PDO</i> has been provided, and it is scaled by the rated current of the motor (read out during <i>Init</i> ).
q_xHomed	BOOL	TRUE: The axis has been successfully homed. It will be reset at each new home request and set again when done.
q_xInPos	BOOL	TRUE: the axis has reached the target position after a <i>Positioning</i> command. This output is affected by the parameters <i>rPositionWindow</i> and <i>uiPositionWinodwTime_ms</i> in <i>i_stData.stProfilePos</i> . This output is reset as soon as a new <i>Positioning</i> command is triggered or if the axis leaves the <i>Standstill</i> state. In case of <i>Stop_OnSetPoint</i> or <i>Immediate_OnSetPoint</i> positionings, <i>q_xInPos</i> signals the end of the ongoing profile and the start of the new profile requested by rising for one task cycle.



q_xInVel	BOOL	TRUE: the axis has reached the target speed during a <i>Velocity</i> command. This output is affected by the parameters <i>rVelocityWindow</i> and <i>uiVelocityWindowTime_ms</i> in <i>i_stData.stProfileVelocity</i> and <i>i_stData.stVelocityMode</i> . This output is reset as soon as the target speed changes or if the axis leaves the actual operating mode.
q_xWarning	BOOL	TRUE: a drive warning is active
q_stWarningDiagnostic	<a href="#">AM_WarningDiagnosticType_ACTIVE</a>	It contains the warning informations
q_stTouchProbeStatus	<a href="#">AM_TouchProbeStatusType</a>	It contains the TouchProbe informations, when enabled (see <i>i_stData.stTouchProbe</i> )
q_uiControlWord_PDO	UINT	Cyclic information for drive's control word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6040</i> ). This object is always mandatory.
q_siModeOfOperation_PDO	SINT	Cyclic information for drive's mode of operation. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6060</i> ). This object is always mandatory.
q_diTargetPosition_PDO	DINT	Cyclic information for drive's target position. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x607A</i> ). Only relevant for Profile Position function.
q_diTargetVelocity_u_s_PDO	DINT	Cyclic information for drive's target velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x60FF</i> ). Only relevant for Profile Velocity function.
q_udiProfileVelocity_u_s_PDO	UDINT	Cyclic information for drive's target profile velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6081</i> ). Only relevant for Profile Position function.
q_iTargetVelocity_VL_PDO	INT	Cyclic information for drive's target velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6042</i> ). Only relevant for Velocity Mode function.



### 7.1.3 AxisMgr\_AGL\_CAN\_ETC

#### Overview

Type:	Function Block
Available as of:	v. 3.1.2.0

#### FUNCTION\_BLOCK AxisMgr\_AGL\_CAN\_ETC EXTENDS AxisMgr\_CAN\_ETC

AxisMgr_AGL_CAN_ETC	
i_xEnable <i>BOOL</i>	<i>BOOL</i> q_xEnabled
i_xPower <i>BOOL</i>	<i>BOOL</i> q_xPowered
i_xReset <i>BOOL</i>	<i>BOOL</i> q_xCmdBusy
i_xQuickStop <i>BOOL</i>	<i>BOOL</i> q_xCmdDone
i_iCmd <i>INT</i>	<i>BOOL</i> q_xFault
i_xLaunchCmd <i>BOOL</i>	<i>BOOL</i> q_xOperational
i_stGenericAccess <i>ST_ParamType_Common</i>	<i>BOOL</i> q_xRemoteControl
i_NetId <i>T_AmsNetID</i>	<i>AM_AxisMgrStatusEnum</i> q_eAxisStatus
i_Port <i>T_AmsPort</i>	<i>AM_Diagnostic</i> q_astErrorDiagnostic
i_uiSlaveAddr <i>UINT</i>	<i>STRING</i> q_strAxisName
i_eFieldbus <i>AM_FieldbusEnum</i>	<i>REAL</i> q_rActualVelocity
i_stData <i>AM_CommandDataType_AGL</i>	<i>BOOL</i> q_xWarning
i_rFactor_vel <i>REAL</i>	<i>BOOL</i> q_xInVel
i_usiActiveDataSet <i>USINT</i>	<i>AM_WarningDiagnosticType_ACTIVE</i> q_stWarningDiagnostic
i_usiDataSet <i>USINT</i>	<i>UINT</i> q_uiControlWord_PDO
i_uiStatusWord_PDO <i>UINT</i>	<i>SINT</i> q_siModeOfOperation_PDO
i_siModeOfOperationDisplay_PDO <i>SINT</i>	<i>INT</i> q_iTargetVelocity_VL_PDO
i_iActualVelocity_VL_PDO <i>INT</i>	<i>REAL</i> q_rActualTorque_Nm
i_iActualTorque_PDO <i>INT</i>	<i>REAL</i> q_rActualCurrent_A
i_iActualCurrent_PDO <i>INT</i>	

#### Description

This FB allows the user to easily manage AGL inverter series either in CANOpen or in EtherCAT. The Velocity Mode by the CiA402 specifications has been implemented.

AxisMgr\_AGL\_CAN\_ETC only works in combination with AGL drive. The FB is designed to work in State Machine control only (*P.412 = 1 - State Machine*). For a proper usage, *P.475 Reference Frequency Source 1* shall be set to *20 - Fieldbus Reference Value* and *P.492 Reference Frequency Source 2* to *0 - Zero*.

AxisMgr\_AGL\_CAN\_ETC is based on the base architecture described in chapter 5. The base interface (chapter 5.2.2) won't be described in this chapter.

The supported function are listed in chapter 5.3.3 and described in the following. This chapter won't give further information.

Management of the fieldbus (cyclic and acyclic datas), as well as specific precautions, has been described in chapter 5.2.4.1.



## Interface

In addition to the base interface (chapter [5.2.2](#)), the AxisMgr AGL CAN ETC's specific In/Out are listed below.

Input	Data Type	Initial	Description
i_NetId	T_AmsNetId		Identification of the EtherCAT master network (only relevant when EtherCAT is used).
i_Port	T_AmsPort		Identification of the CANOpen port (only relevant when CANOpen is used).
i_uiSlaveAddr	UINT		Physical slave address (EtherCAT) or the device Node ID (CANOpen).
i_eFieldBus	AM_FieldbusEnum		Fieldbus selection (either CANOpen or EtherCAT).
i_stData	<a href="#">AM_CommandDataType_AGL</a>		Structure of data relevant for the supported functions. Compile the data structures to configure the function parameters. Each structure is only relevant when using the related function.
i_rFactor_vel	REAL	1	Conversion factor [VL Units/vel. units]. It defines the amount of drive's VL units that represent 1u speed in the user program. This value is relevant for the speed-controller functions (Velocity Mode). See chapter <a href="#">5.3.1.2</a> .
i_usiActiveDataSet	USINT		Not implemented yet.
i_usiDataSet	USINT		Not implemented yet.
i_uiStatusWord_PDO	UINT		Cyclic information for drive's status word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6041</i> ). This object is always mandatory.
i_siModeOfOperationDisplay_PDO	SINT		Cyclic information for drive's mode of operation display. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6061</i> ). This object is always mandatory.
i_iActualVelocity_VL_PDO	INT		Cyclic information for drive's actual velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6044</i> ).
i_iActualTorque_PDO	INT		Cyclic information for drive's actual torque [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6077</i> ).
i_iActualCurrent_PDO	INT		Cyclic information for drive's actual current [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6078</i> ).



Output	Data Type	Description
q_rActualVelocity	REAL	Actual velocity of the axis [vel. units]. The value is updated only if <i>i_iActualVelocity_VL_PDO</i> has been provided, and it is scaled by <i>i_rFactorVel</i> .
q_xWarning	BOOL	TRUE: a drive warning is active
q_xInVel	BOOL	TRUE: the axis has reached the target speed during a <i>Velocity</i> command. This output is affected by the parameters <i>rVelocityWindow</i> and <i>uiVelocityWindowTime_ms</i> in <i>i_stData.stVelocityMode</i> . This output is reset as soon as the target speed changes or if the axis leaves the actual operating mode.
q_stWarningDiagnostic	<a href="#">AM_WarningDiagnosticType_ACTIVE</a>	It contains the warning informations
q_uiControlWord_PDO	UINT	Cyclic information for drive's control word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6040</i> ). This object is always mandatory.
q_siModeOfOperation_PDO	SINT	Cyclic information for drive's mode of operation. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6060</i> ). This object is always mandatory.
q_iTargetVelocity_VL_PDO	INT	Cyclic information for drive's target velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6042</i> ). Only relevant for Velocity Mode function.
q_rActualTorque_Nm	REAL	Actual torque of the axis [Nm]. The value is updated only if <i>i_iActualTorque_PDO</i> has been provided, and it is scaled by the rated torque of the motor (read out during <i>Init</i> ).
q_rActualCurrent_A	REAL	Actual current of the axis [A]. The value is updated only if <i>i_iActualCurrent_PDO</i> has been provided, and it is scaled by the rated current of the motor (read out during <i>Init</i> ).



## 7.1.4 AxisMgr\_ACUx10\_CAN\_ETC

### Overview

Type:	Function Block
Available as of:	v. 3.1.2.0

### FUNCTION\_BLOCK AxisMgr\_ACUx10\_CAN\_ETC EXTENDS AxisMgr\_CAN\_ETC

AxisMgr_ACUx10_CAN_ETC	
i_xEnable <i>BOOL</i>	<i>BOOL</i> q_xEnabled
i_xPower <i>BOOL</i>	<i>BOOL</i> q_xPowered
i_xReset <i>BOOL</i>	<i>BOOL</i> q_xCmdBusy
i_xQuickStop <i>BOOL</i>	<i>BOOL</i> q_xCmdDone
i_iCmd <i>INT</i>	<i>BOOL</i> q_xFault
i_xLaunchCmd <i>BOOL</i>	<i>BOOL</i> q_xOperational
i_stGenericAccess <i>ST_ParamType_Common</i>	<i>BOOL</i> q_xRemoteControl
i_NetId <i>T_AmsNetID</i>	<i>AM_AxisMgrStatusEnum</i> q_eAxisStatus
i_Port <i>T_AmsPort</i>	<i>AM_Diagnostic</i> q_astErrorDiagnostic
i_uiSlaveAddr <i>UINT</i>	<i>STRING</i> q_strAxisName
i_eFieldbus <i>AM_FieldbusEnum</i>	<i>REAL</i> q_rActualPosition_u
i_stData <i>AM_CommandDataType_ACTIVE</i>	<i>REAL</i> q_rActualVelocity_u_s
i_rFactor_pos <i>REAL</i>	<i>REAL</i> q_rActualVelocity_vel
i_rFactor_vel <i>REAL</i>	<i>REAL</i> q_rActualTorque_Nm
i_uiStatusWord_PDO <i>UINT</i>	<i>REAL</i> q_rActualCurrent_A
i_siModeOfOperationDisplay_PDO <i>SINT</i>	<i>BOOL</i> q_xHomed
i_diActualPosition_PDO <i>DINT</i>	<i>BOOL</i> q_xInPos
i_diActualVelocity_u_s_PDO <i>DINT</i>	<i>BOOL</i> q_xInVel
i_iActualVelocity_VL_PDO <i>INT</i>	<i>BOOL</i> q_xWarning
i_iActualTorque_PDO <i>INT</i>	<i>AM_WarningDiagnosticType_ACTIVE</i> q_stWarningDiagnostic
i_iActualCurrent_PDO <i>INT</i>	<i>AM_TouchProbeStatusType</i> q_stTouchProbeStatus
i_usiDataSet <i>USINT</i>	<i>UINT</i> q_uiControlWord_PDO
	<i>SINT</i> q_siModeOfOperation_PDO
	<i>DINT</i> q_diTargetPosition_PDO
	<i>DINT</i> q_diTargetVelocity_u_s_PDO
	<i>UDINT</i> q_udiProfileVelocity_u_s_PDO
	<i>INT</i> q_iTargetVelocity_VL_PDO

### Description

This FB allows the user to easily manage ACUx10 drive series either in CANOpen or in EtherCAT. The motion profiles defined by the CiA402 specifications has been implemented, in combination with the manufacturer specific features. Unlike [AxisMgr\\_SM](#), the controller does not calculate and send the trajectory to the drive, but the drive's motion profile generator is used. This also means that the controller is not intended to have demanding real-time capabilities, since the control of the drive may not be deterministic.

AxisMgr\_ACUx10\_CAN\_ETC only works in combination with ANG drive. The FB is designed to work in State Machine control only (*P.412 = 1 - State Machine*). Depending on the actual drive configuration (*P.30 Configuration*), some functions may not be supported.

AxisMgr\_ACUx10\_CAN\_ETC is based on the base architecture described in chapter [5](#). The base interface (chapter [5.2.2](#)) won't be described in this chapter.

The supported function are listed in chapter [5.3.3](#) and described in the following. This chapter won't give further information.

Fieldbus management (cyclic and acyclic datas), as well as specific precautions, has been described in chapter [5.2.4.1](#).



## Interface

In addition to the base interface (chapter [5.2.2](#)), the AxisMgr ACUx10 CAN ETC's specific In/Out are listed below.

Input	Data Type	Initial	Description
i_NetId	T_AmsNetId		Identification of the EtherCAT master network (only relevant when EtherCAT is used).
i_Port	T_AmsPort		Identification of the CANOpen port (only relevant when CANOpen is used).
i_uiSlaveAddr	UINT		Physical slave address (EtherCAT) or the device Node ID (CANOpen).
i_eFieldBus	AM_FieldbusEnum		Fieldbus selection (either CANOpen or EtherCAT).
i_stData	<a href="#">AM_CommandDataType ACTIVE</a>		Structure of data relevant for the supported functions. Compile the data structures to configure the function parameters. Each structure is only relevant when using the related function.
i_rFactor_pos	REAL	65536 / 360.0	Conversion factor [increments/units]. It defines the amount of the drive's increment that represent 1u displacement in the user program. This value is relevant for the position-controlled functions (Profile Position, Profile Velocity, Homing, ecc...). See chapter <a href="#">5.3.1.2</a> .
i_rFactor_vel	REAL	1	Conversion factor [VL Units/vel. units]. It defines the amount of drive's VL units that represent 1u speed in the user program. This value is relevant for the speed-controlled functions (Velocity Mode). See chapter <a href="#">5.3.1.2</a> .
i_uiStatusWord_PDO	UINT		Cyclic information for drive's status word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6041</i> ). This object is always mandatory.
i_siModeOfOperationDisplay_PDO	SINT		Cyclic information for drive's mode of operation display. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6061</i> ). This object is always mandatory.
i_diActualPosition_PDO	DINT		Cyclic information for drive's actual position. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6064</i> ).



i_diActualVelocity_u_s_PDO	DINT		Cyclic information for drive's actual velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x606C</i> ).
i_iActualVelocity_VL_PDO	INT		Cyclic information for drive's actual velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6044</i> ).
i_iActualTorque_PDO	INT		Cyclic information for drive's actual torque [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6077</i> ).
i_iActualCurrent_PDO	INT		Cyclic information for drive's actual current [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6078</i> ).
i_usiDataSet	USINT		Not implemented yet

Output	Data Type	Description
q_rActualPosition_u	REAL	Actual position of the axis [u], referred to the last home referencing. The value is updated only if <i>i_diActualPosition_PDO</i> has been provided, and it is scaled by <i>i_rFactorPos</i> .
q_rActualVelocity_u_s	REAL	Actual velocity of the axis [u/s]. The value is updated only if <i>i_diActualVelocity_u_s_PDO</i> has been provided, and it is scaled by <i>i_rFactorPos</i> .
q_rActualVelocity_vel	REAL	Actual velocity of the axis [vel. units]. The value is updated only if <i>i_iActualVelocity_VL_PDO</i> has been provided, and scaled by <i>i_rFactorVel</i> .
q_rActualTorque_Nm	REAL	Actual torque of the axis [Nm]. The value is updated only if <i>i_iActualTorque_PDO</i> has been provided, and it is scaled by the rated torque of the motor (read out during <i>Init</i> ).
q_rActualCurrent_A	REAL	Actual current of the axis [A]. The value is updated only if <i>i_iActualCurrent_PDO</i> has been provided, and it is scaled by the rated current of the motor (read out during <i>Init</i> ).
q_xHomed	BOOL	TRUE: The axis has been successfully homed. It will be reset at each new home request and set again when done.
q_xInPos	BOOL	TRUE: the axis has reached the target position after a <i>Positioning</i> command. This output is affected by the parameters <i>rPositionWindow</i> and <i>uiPositionWinodwTime_ms</i> in <i>i_stData.stProfilePos</i> . This output is reset as soon as a new <i>Positioning</i> command is triggered or if the axis leaves the <i>Standstill</i> state. In case of <i>Stop_OnSetPoint</i> or <i>Immediate_OnSetPoint</i> positionings, <i>q_xInPos</i> signals the end of the ongoing profile and the start of the new profile requested by rising for one task cycle.



q_xInVel	BOOL	TRUE: the axis has reached the target speed during a <i>Velocity</i> command. This output is affected by the parameters <i>rVelocityWindow</i> and <i>uiVelocityWindowTime_ms</i> in <i>i_stData.stProfileVelocity</i> and <i>i_stData.stVelocityMode</i> . This output is reset as soon as the target speed changes or if the axis leaves the actual operating mode.
q_xWarning	BOOL	TRUE: a drive warning is active
q_stWarningDiagnostic	<a href="#">AM_WarningDiagnosticType_ACTIVE</a>	It contains the warning informations
q_stTouchProbeStatus	<a href="#">AM_TouchProbeStatusType</a>	It contains the TouchProbe informations, when enabled (see <i>i_stData.stTouchProbe</i> )
q_uiControlWord_PDO	UINT	Cyclic information for drive's control word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6040</i> ). This object is always mandatory.
q_siModeOfOperation_PDO	SINT	Cyclic information for drive's mode of operation. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6060</i> ). This object is always mandatory.
q_diTargetPosition_PDO	DINT	Cyclic information for drive's target position. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x607A</i> ). Only relevant for Profile Position function.
q_diTargetVelocity_u_s_PDO	DINT	Cyclic information for drive's target velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x60FF</i> ). Only relevant for Profile Velocity function.
q_udiProfileVelocity_u_s_PDO	UDINT	Cyclic information for drive's target profile velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6081</i> ). Only relevant for Profile Position function.
q_iTargetVelocity_VL_PDO	INT	Cyclic information for drive's target velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6042</i> ). Only relevant for Velocity Mode function.



## 7.1.5 AxisMgr\_AXV\_CAN\_ETC

### Overview

Type:	Function Block
Available as of:	v. 3.1.2.0

### FUNCTION\_BLOCK AxisMgr\_AXV\_CAN\_ETC EXTENDS AxisMgr\_CAN\_ETC

AxisMgr_AXV_CAN_ETC	
i_xEnable <i>BOOL</i>	<i>BOOL</i> q_xEnabled
i_xPower <i>BOOL</i>	<i>BOOL</i> q_xPowered
i_xReset <i>BOOL</i>	<i>BOOL</i> q_xCmdBusy
i_xQuickStop <i>BOOL</i>	<i>BOOL</i> q_xCmdDone
i_iCmd <i>INT</i>	<i>BOOL</i> q_xFault
i_xLaunchCmd <i>BOOL</i>	<i>BOOL</i> q_xOperational
i_stGenericAccess <i>ST_ParamType_Common</i>	<i>BOOL</i> q_xRemoteControl
i_NetId <i>T_AmsNetID</i>	<i>AM_AxisMgrStatusEnum</i> q_eAxisStatus
i_Port <i>T_AmsPort</i>	<i>AM_Diagnostic</i> q_astErrorDiagnostic
i_uiSlaveAddr <i>UINT</i>	<i>STRING</i> q_strAxisName
i_eFieldbus <i>AM_FieldbusEnum</i>	<i>REAL</i> q_rActualPosition_u
i_stData <i>AM_CommandDataType_Axia</i>	<i>REAL</i> q_rActualVelocity_u_s
i_rFactor_pos <i>REAL</i>	<i>REAL</i> q_rActualVelocity_vel
i_rFactor_vel <i>REAL</i>	<i>REAL</i> q_rActualTorque_Nm
i_uiStatusWord_PDO <i>UINT</i>	<i>BOOL</i> q_xHomed
i_siModeOfOperationDisplay_PDO <i>SINT</i>	<i>BOOL</i> q_xInPos
i_diActualPosition_PDO <i>DINT</i>	<i>BOOL</i> q_xInVel
i_diActualVelocity_u_s_PDO <i>DINT</i>	<i>BOOL</i> q_xWarning
i_iActualVelocity_VL_PDO <i>INT</i>	<i>AM_WarningDiagnosticType_Axia</i> q_stWarningDiagnostic
i_iActualTorque_PDO <i>INT</i>	<i>AM_TouchProbeStatusType</i> q_stTouchProbeStatus
i_usiDataSet <i>USINT</i>	<i>UINT</i> q_uiControlWord_PDO
	<i>SINT</i> q_siModeOfOperation_PDO
	<i>DINT</i> q_diTargetPosition_PDO
	<i>DINT</i> q_diTargetVelocity_u_s_PDO
	<i>UDINT</i> q_udiProfileVelocity_u_s_PDO
	<i>INT</i> q_iTargetVelocity_VL_PDO
	<i>INT</i> q_iTargetTorque_PDO

### Description

This FB allows the user to easily manage Axia Vert drive series either in CANOpen or in EtherCAT. The motion profiles defined by the CiA402 specifications has been implemented, in combination with the manufacturer specific features. Unlike [AxisMgr\\_SM](#), the controller does not calculate and send the trajectory to the drive, but the drive's motion profile generator is used. This also means that the controller is not intended to have demanding real-time capabilities, since the control of the drive may not be deterministic.

AxisMgr AXV CAN ETC only works in combination with Axia Vert drive. The FB is designed to work in State Machine control only (*obj. 0x2200 = 3 - State Machine*). Depending on the actual drive configuration, some functions may not be supported.

AxisMgr AXV CAN ETC is based on the base architecture described in chapter [5](#). The base interface (chapter [5.2.2](#)) won't be described in this chapter.

The supported function are listed in chapter [5.3.3](#) and described in the following. This chapter won't give further information.

Management of the fieldbus (cyclic and acyclic datas), as well as specific precautions, has been described in chapter [5.2.4.1](#).



## Interface

In addition to the base interface (chapter [5.2.2](#)), the AxisMgr AXV CAN ETC's specific In/Out are listed below.

Input	Data Type	Initial	Description
i_NetId	T_AmsNetId		Identification of the EtherCAT master network (only relevant when EtherCAT is used).
i_Port	T_AmsPort		Identification of the CANOpen port (only relevant when CANOpen is used).
i_uiSlaveAddr	UINT		Physical slave address (EtherCAT) or the device Node ID (CANOpen).
i_eFieldBus	AM_FieldbusEnum		Fieldbus selection (either CANOpen or EtherCAT).
i_stData	<a href="#">AM_CommandDataType_Axia</a>		Structure of data relevant for the supported functions. Compile the data structures to configure the function parameters. Each structure is only relevant when using the related function.
i_rFactor_pos	REAL	65536 / 360.0	Conversion factor [increments/units]. It defines the amount of the drive's increment that represent 1u displacement in the user program. This value is relevant for the position-controlled functions (Profile Position, Profile Velocity, Homing, ecc...). See chapter <a href="#">5.3.1.2</a> .
i_rFactor_vel	REAL	1	Conversion factor [VL Units/vel. units]. It defines the amount of drive's VL units that represent 1u speed in the user program. This value is relevant for the speed-controlled functions (Velocity Mode). See chapter <a href="#">5.3.1.2</a> .
i_uiStatusWord_PDO	UINT		Cyclic information for drive's status word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6041</i> ). This object is always mandatory.
i_siModeOfOperationDisplay_PDO	SINT		Cyclic information for drive's mode of operation display. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6061</i> ). This object is always mandatory.
i_diActualPosition_PDO	DINT		Cyclic information for drive's actual position. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6064</i> ).



i_diActualVelocity_u_s_PDO	DINT		Cyclic information for drive's actual velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x606C</i> ).
i_iActualVelocity_VL_PDO	INT		Cyclic information for drive's actual velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6044</i> ).
i_iActualTorque_PDO	INT		Cyclic information for drive's actual torque [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6077</i> ).
i_usiDataSet	USINT		Not implemented yet

Output	Data Type	Description
q_rActualPosition_u	REAL	Actual position of the axis [u], referred to the last home referencing. The value is updated only if <i>i_diActualPosition_PDO</i> has been provided, and scaled by <i>i_rFactorPos</i> .
q_rActualVelocity_u_s	REAL	Actual velocity of the axis [u/s]. The value is updated only if <i>i_diActualVelocity_u_s_PDO</i> has been provided, and scaled by <i>i_rFactorPos</i> .
q_rActualVelocity_vel	REAL	Actual velocity of the axis [vel. units]. The value is updated only if <i>i_iActualVelocity_VL_PDO</i> has been provided, and scaled by <i>i_rFactorVel</i> .
q_rActualTorque_Nm	REAL	Actual torque of the axis [Nm]. The value is updated only if <i>i_iActualTorque_PDO</i> has been provided, and it is scaled by the rated torque of the motor (read out during <i>Init</i> ).
q_xHomed	BOOL	TRUE: The axis has been successfully homed. It will be reset at each new home request and set again when done.
q_xInPos	BOOL	TRUE: the axis has reached the target position after a <i>Positioning</i> command. This output is affected by the parameters <i>rPositionWindow</i> and <i>uiPositionWindowTime_ms</i> in <i>i_stData.stProfilePos</i> . This output is reset as soon as a new <i>Positioning</i> command is triggered or if the axis leaves the <i>Standstill</i> state.
q_xInVel	BOOL	TRUE: the axis has reached the target speed during a <i>Velocity</i> command. This output is affected by the parameters <i>rVelocityWindow</i> and <i>uiVelocityWindowTime_ms</i> in <i>i_stData.stProfileVelocity</i> and <i>i_stData.stVelocityMode</i> . This output is reset as soon as the target speed changes or if the axis leaves the actual operating mode.
q_xWarning	BOOL	TRUE: a drive warning is active



q_stWarningDiagnostic	<a href="#">AM_WarningDiagnosticType_Axia</a>	It contains the warning informations
q_stTouchProbeStatus	<a href="#">AM_TouchProbeStatusType</a>	It contains the TouchProbe informations, when enabled (see <i>i_stData.stTouchProbe</i> )
q_uiControlWord_PDO	UINT	Cyclic information for drive's control word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6040</i> ). This object is always mandatory.
q_siModeOfOperation_PDO	SINT	Cyclic information for drive's mode of operation. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6060</i> ). This object is always mandatory.
q_diTargetPosition_PDO	DINT	Cyclic information for drive's target position. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x607A</i> ). Only relevant for Profile Position function.
q_diTargetVelocity_u_s_PDO	DINT	Cyclic information for drive's target velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x60FF</i> ). Only relevant for Profile Velocity function.
q_udiProfileVelocity_u_s_PDO	UDINT	Cyclic information for drive's target profile velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6081</i> ). Only relevant for Profile Position function.
q_iTargetVelocity_VL_PDO	INT	Cyclic information for drive's target velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6042</i> ). Only relevant for Velocity Mode function.
q_iTargetTorque_PDO	INT	Cyclic information for drive's target torque [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6071</i> ). Only relevant for Profile Torque function.



## 7.1.6 AxisMgr\_AXM\_CAN\_ETC

### Overview

Type:	Function Block
Available as of:	v. 3.1.2.0

### FUNCTION\_BLOCK AxisMgr\_AXM\_CAN\_ETC EXTENDS AxisMgr\_CAN\_ETC

AxisMgr_AXM_CAN_ETC	
i_xEnable <i>BOOL</i>	<i>BOOL</i> q_xEnabled
i_xPower <i>BOOL</i>	<i>BOOL</i> q_xPowered
i_xReset <i>BOOL</i>	<i>BOOL</i> q_xCmdBusy
i_xQuickStop <i>BOOL</i>	<i>BOOL</i> q_xCmdDone
i_iCmd <i>INT</i>	<i>BOOL</i> q_xFault
i_xLaunchCmd <i>BOOL</i>	<i>BOOL</i> q_xOperational
i_stGenericAccess <i>ST_ParamType_Common</i>	<i>BOOL</i> q_xRemoteControl
i_NetId <i>T_AmsNetID</i>	<i>AM_AxisMgrStatusEnum</i> q_eAxisStatus
i_Port <i>T_AmsPort</i>	<i>AM_Diagnostic</i> q_astErrorDiagnostic
i_uiSlaveAddr <i>UINT</i>	<i>STRING</i> q_strAxisName
i_eFieldbus <i>AM_FieldbusEnum</i>	<i>REAL</i> q_rActualPosition_u
i_stData <i>AM_CommandDataType_Axia</i>	<i>REAL</i> q_rActualVelocity_u_s
i_rFactor_pos <i>REAL</i>	<i>REAL</i> q_rActualVelocity_vel
i_rFactor_vel <i>REAL</i>	<i>REAL</i> q_rActualTorque_Nm
i_uiStatusWord_PDO <i>UINT</i>	<i>BOOL</i> q_xHomed
i_siModeOfOperationDisplay_PDO <i>SINT</i>	<i>BOOL</i> q_xInPos
i_diActualPosition_PDO <i>DINT</i>	<i>BOOL</i> q_xInVel
i_diActualVelocity_u_s_PDO <i>DINT</i>	<i>BOOL</i> q_xWarning
i_iActualVelocity_VL_PDO <i>INT</i>	<i>AM_WarningDiagnosticType_Axia</i> q_stWarningDiagnostic
i_iActualTorque_PDO <i>INT</i>	<i>AM_TouchProbeStatusType</i> q_stTouchProbeStatus
i_usiDataSet <i>USINT</i>	<i>UINT</i> q_uiControlWord_PDO
i_usiAxisNumber <i>USINT</i>	<i>SINT</i> q_siModeOfOperation_PDO
	<i>DINT</i> q_diTargetPosition_PDO
	<i>DINT</i> q_diTargetVelocity_u_s_PDO
	<i>UDINT</i> q_udiProfileVelocity_u_s_PDO
	<i>INT</i> q_iTargetVelocity_VL_PDO
	<i>INT</i> q_iTargetTorque_PDO

### Description

This FB allows the user to easily manage Axia Move drive series (single or double axis) either in CANOpen or in EtherCAT. The motion profiles defined by the CiA402 specifications has been implemented, in combination with the manufacturer specific features. Unlike [AxisMgr\\_SM](#), the controller does not calculate and send the trajectory to the drive, but the drive's motion profile generator is used. This also means that the controller is not intended to have demanding real-time capabilities, since the control of the drive may not be deterministic.

AxisMgr\_AXM\_CAN\_ETC only works in combination with Axia Vert drive. The FB is designed to work in State Machine control only (*obj. 0x2200/0x2A00 = 3 - State Machine*). Depending on the actual drive configuration, some functions may not be supported.

Unlike [AxisMgr\\_SM](#), AxisMgr\_AXM\_CAN\_ETC is not able to automatically detect the axis number. In the case of an Axis Move Double Axis, *i\_usiAxisNumber* should be set to either 1 or 2 depending on the axis number that the FB will control.

AxisMgr\_AXM\_CAN\_ETC is based on the base architecture described in chapter [5](#). The base interface (chapter [5.2.2](#)) won't be described in this chapter.

The supported function are listed in chapter [5.3.3](#) and described in the following. This chapter won't give further information.

Management of the fieldbus (cyclic and acyclic datas), as well as specific precautions, has been described in chapter [5.2.4.1](#).



## Interface

In addition to the base interface (chapter [5.2.2](#)), the AxisMgr AXM CAN ETC's specific In/Out are listed below.

Input	Data Type	Initial	Description
i_NetId	T_AmsNetId		Identification of the EtherCAT master network (only relevant when EtherCAT is used).
i_Port	T_AmsPort		Identification of the CANOpen port (only relevant when CANOpen is used).
i_uiSlaveAddr	UINT		Physical slave address (EtherCAT) or the device Node ID (CANOpen).
i_eFieldBus	AM_FieldbusEnum		Fieldbus selection (either CANOpen or EtherCAT).
i_stData	<a href="#">AM CommandData Type Axia</a>		Structure of data relevant for the supported functions. Compile the data structures to configure the function parameters. Each structure is only relevant when using the related function.
i_rFactor_pos	REAL	65536 / 360.0	Conversion factor [increments/units]. It defines the amount of the drive's increment that represent 1u displacement in the user program. This value is relevant for the position-controlled functions (Profile Position, Profile Velocity, Homing, ecc...). See chapter <a href="#">5.3.1.2</a> .
i_rFactor_vel	REAL	1	Conversion factor [VL Units/vel. units]. It defines the amount of drive's VL units that represent 1u speed in the user program. This value is relevant for the speed-controlled functions (Velocity Mode). See chapter <a href="#">5.3.1.2</a> .
i_uiStatusWord_PDO	UINT		Cyclic information for drive's status word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6041</i> ). This object is always mandatory.
i_siModeOfOperationDisplay_PDO	SINT		Cyclic information for drive's mode of operation display. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6061</i> ). This object is always mandatory.
i_diActualPosition_PDO	DINT		Cyclic information for drive's actual position. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6064</i> ).
i_diActualVelocity_u_s_PDO	DINT		Cyclic information for drive's actual velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x606C</i> ).
i_iActualVelocity_VL_PDO	INT		Cyclic information for drive's actual velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6044</i> ).



i_iActualTorque_PDO	INT		Cyclic information for drive's actual torque [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6077</i> ).
i_usiDataSet	USINT		Not implemented yet
i_usiAxisNumber	USINT	1	In the case of an Axia Move Double Axis, it defines the axis number (either 1 or 2) that the AxisMgr should control.

Output	Data Type	Description
q_rActualPosition_u	REAL	Actual position of the axis [u], referred to the last home referencing. The value is updated only if <i>i_diActualPosition_PDO</i> has been provided, and scaled by <i>i_rFactorPos</i> .
q_rActualVelocity_u_s	REAL	Actual velocity of the axis [u/s]. The value is updated only if <i>i_diActualVelocity_u_s_PDO</i> has been provided, and scaled by <i>i_rFactorPos</i> .
q_rActualVelocity_vel	REAL	Actual velocity of the axis [vel. units]. The value is updated only if <i>i_iActualVelocity_VL_PDO</i> has been provided, and scaled by <i>i_rFactorVel</i> .
q_rActualTorque_Nm	REAL	Actual torque of the axis [Nm]. The value is updated only if <i>i_iActualTorque_PDO</i> has been provided, and it is scaled by the rated torque of the motor (read out during <i>Init</i> ).
q_xHomed	BOOL	TRUE: The axis has been successfully homed. It will be reset at each new home request and set again when done.
q_xInPos	BOOL	TRUE: the axis has reached the target position after a <i>Positioning</i> command. This output is affected by the parameters <i>rPositionWindow</i> and <i>uiPositionWindowTime_ms</i> in <i>i_stData.stProfilePos</i> . This output is reset as soon as a new <i>Positioning</i> command is triggered or if the axis leaves the <i>Standstill</i> state.
q_xInVel	BOOL	TRUE: the axis has reached the target speed during a <i>Velocity</i> command. This output is affected by the parameters <i>rVelocityWindow</i> and <i>uiVelocityWindowTime_ms</i> in <i>i_stData.stProfileVelocity</i> and <i>i_stData.stVelocityMode</i> . This output is reset as soon as the target speed changes or if the axis leaves the actual operating mode.
q_xWarning	BOOL	TRUE: a drive warning is active
q_stWarningDiagnostic	<a href="#">AM_WarningDiagnosticType_Axia</a>	It contains the warning informations
q_stTouchProbeStatus	<a href="#">AM_TouchProbeStatusType</a>	It contains the TouchProbe informations, when enabled (see <i>i_stData.stTouchProbe</i> )



q_uiControlWord_PDO	UINT	Cyclic information for drive's control word. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6040</i> ). This object is always mandatory.
q_siModeOfOperation_PDO	SINT	Cyclic information for drive's mode of operation. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6060</i> ). This object is always mandatory.
q_diTargetPosition_PDO	DINT	Cyclic information for drive's target position. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6074</i> ). Only relevant for Profile Position function.
q_diTargetVelocity_u_s_PDO	DINT	Cyclic information for drive's target velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x60FF</i> ). Only relevant for Profile Velocity function.
q_udiProfileVelocity_u_s_PDO	UDINT	Cyclic information for drive's target profile velocity [incr/s]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6081</i> ). Only relevant for Profile Position function.
q_iTargetVelocity_VL_PDO	INT	Cyclic information for drive's target velocity [VL units]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6042</i> ). Only relevant for Velocity Mode function.
q_iTargetTorque_PDO	INT	Cyclic information for drive's target torque [% x10]. It has to be linked/assigned directly to the I/O mapping of the drive's PDO ( <i>obj. 0x6071</i> ). Only relevant for Profile Torque function.

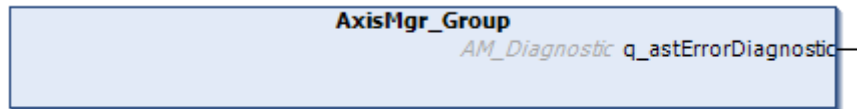


## 7.1.7 AxisMgr\_Group

### Overview

Type:	Function Block
Available as of:	v. 3.1.2.0

### FUNCTION\_BLOCK AxisMgr\_Group EXTENDS AxisMgr\_Group\_BASE



### Description

This Function Block provides an easy way to control a group of AxisMgr from a single point. It is meaningful to say that this FB represents a group of axes.

This FB must be configured. Thanks to the method *ADD\_AXISMGR*, it is possible to add an axis into the logic of the group. An axis can take part of several groups and receive the command from each group in which it has been added.

In this way the user can separate application, based on groups logic.

An instance of *AxisMgr\_Group* is already defined (chapter [9.1](#)). *gbl\_fbAxisGroup* groups all the AxisMgr defined in the application.

Once the AxisMgr are added, it is sufficient to call the *AxisMgr\_Group* instance in a program task to run all the AxisMgr added. Since *gbl\_fbAxisGroup* has been already provided, the user can eventually just call it in order to put all the AxisMgr in execution. This result in a less complex code and in a coding time reduction.

The I/O interface is replaced by properties. *AxisMgr\_Group* writes the command to the AxisMgrs added just when the properties are called. In the same way, it gets the AxisMgrs status signals once the property is evaluated in the program.

A maximum number of *\_AM\_K.k\_iMaxNumGroupsPerAxis* can be added per each group.

### Interface

Output	Data Type	Description
q_astErrorDiagnostic	AM_Diagnostic ( <a href="#">4.5</a> )	It groups the error diagnostic of the AxisMgr added. As soon as an group member reports an error, it is reported here at the same time, with the name of the AxisMgr instance. The first entry in the list is always the most recent error event. The size of the array can be adjusted modifying <i>k_iDiagnostic</i> from the Library Manager->Parameters. Keep it low to optimize the memory usage.



### ADD\_AXISMGR (Method)

This method has to be called in order to assign the specified AxisMgr to the *AxisMgr Group*. The method can be either called once at the beginning or cyclically in the program. The execution order of the AxisMgr inside the group is the same as they have been added. If more than one axis are supposed to be *Synchronized*, then they should be added after the master instance (see chapter [7.1.1](#)).

It should not be used by *gbl\_fbAxisGroup*, since all the AxisMgr instances have been already added implicitly.

Input	Data Type	Description
itfAxisMgr	ITF_AxisMgr	Interface to the AxisMgr. Assign here the instance of the AxisMgr.

### Properties

The following table will list all the available properties of *AxisMgr Group*. They represent the same command/signal of the AxisMgr base interface (see chapter [5.2.2](#)), in addition to some extra signals.

Name	Data Type	Access	Description
i_iCmd	INT	Set	Command selection input. It defines which functions should be executed by the group members. The possible values and the commands supported are listed in the FB description. The command will be taken into account only after a rising edge on <i>i_xLaunchCmd</i> .  The value set by this property will not overwrite AxisMgr.i_iCmd, but it will work separately.
i_xEnable	BOOL	Set	Set to TRUE to enable the execution of the AxisMgrs added. Execution of the initialization of the drive will start if not yet carried out.
i_xLaunchCmd	BOOL	Set	Rising Edge: launch the command selected in <i>i_iCmd</i> .  The value set by this property will not overwrite AxisMgr.i_xLaunchCmd, but it will work separately.
i_xPower	BOOL	Set	Rising edge: power the axes. Falling edge: disable the axes.
i_xQuickStop	BOOL	Set	Rising Edge: put the axes in <i>EmergencyStop</i> state and trigger the emergency ramp of the drive. As long as it remains TRUE, it prevents the axis to get powered. When using virtual axes, the stop ramp is immediate, regardless the settings in <i>i_stData.stQuickStop</i>
i_xReset	BOOL	Set	Rising edge: clear drive's fault and FB's diagnostic
q_xCmdDone	BOOL	Get	TRUE: the function indicated by <i>i_iCmd</i> has been finished by all the axes added. It remains TRUE as long as <i>i_xLaunchCmd</i> stays TRUE. In the case <i>i_xLaunchCmd</i> is set back to FALSE before the function reaches its end, then <i>q_xCmdDone</i> will raise just for a task cycle.  The group result of <i>q_xCmdDone</i> only relates to the function launched via the <i>AxisMgr Group</i> . The function result of the single AxisMgr won't be displayed here.
q_xEnabled	BOOL	Get	TRUE: All group members are enabled ( <i>i_xEnable</i> = TRUE) and initialization has finished.
q_xFault	BOOL	Get	TRUE: At least one of the group member reported an error.
q_xHomed	BOOL	Get	TRUE: All group members are homed
q_xInPos	BOOL	Get	TRUE: All group members are InPos
q_xInSync	BOOL	Get	TRUE: All the group members (that has got a master reference) are InSync.
q_xNotPowered	BOOL	Get	TRUE: None of the AxisMgrs added are powered
q_xOperational	BOOL	Get	TRUE: All the group members are Operational



q_xPowered	BOOL	Get	TRUE: All the group members are powered
q_xRemoteControl	BOOL	Get	TRUE: All the group members can be controlled from remote (ready to receive commands)
q_xStandstill	BOOL	Get	TRUE: All the group members are in state <i>Standstill</i>
q_xWarning	BOOL	Get	TRUE: At least one of the group member is reporting an hardware warning.

#### NOTE

Each properties, once called, may set/get the corresponding signal of the group members. The user should consider this especially when set commands; when setting a signal both from the single AxisMgr and from the group in the same cycle, the last value set before the execution of the group will take effect.



- All the properties can be eventually accessed with "Get" access, even if they are listed as "Set" only. Those properties that are "Set" only, does not get the corresponding value from all the group members, but only returns the last value set in the group.
- If one or more axis are being disabled (from the project tree), then, when evaluating one of the "Get" properties, the value of those axes is bypassed. This avoid the user to delete the axis from the group (changing the code); the evaluation logic will work anyway as if the axis is not infact present. This also applies if a "Get" signal is not supported by one or more AxisMgr added (e.g.: *q\_xInPos* of an *AxisMgr AGL <fieldbus>* will be considered TRUE, *q\_xInPos* should be considered as the overall signal for those which supports it in their interface).

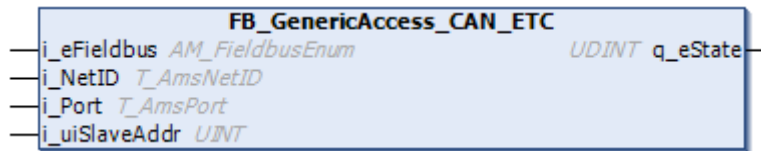


## 7.1.8 FB\_GenericAcces\_CAN\_ETC

### Overview

Type:	Function Block
Available as of:	v. 3.1.2.0

### FUNCTION\_BLOCK FB\_GenericAccess\_CAN\_ETC EXTENDS FB\_GenericAccess\_Common



### Description

The FB is very useful when it comes to manage several SDO requests, either with a CANOpen or an EtherCAT slave. The FB just put all the parameter in a queue, in such a way that when a request come from one parameter then the next has to wait for its response, avoiding that more than one request are performed at once.

See [ST\\_ParamType\\_Common](#) for further information about how to use the parameters.

It is designed to work either with CANOpen or EtherCAT.

### Interface

Input	Data Type	Description
i_eFieldbus	<a href="#">AM_FieldbusEnum</a>	Fieldbus selection
i_NetId	T_AmsNetID	It defines the EtherCAT master Net ID. Only relevant when <i>i_eFieldbus</i> = <i>EtherCAT</i>
i_Port	T_AmsPort	It defines the CANOpen port. Only relevant when <i>i_eFieldbus</i> = <i>CANOpen</i>
i_uiSlaveAddr	UINT	Depending on the value set in <i>i_eFieldbus</i> , it represents either the Node ID (CANOpen) or the Physical Address (EtherCAT).

Output	Data Type	Description
q_eState	<a href="#">EN_State_CAN_ETC</a>	Returns the NMT state of the slave.

### ADD\_PARAM (Method)

This method has to be called in order to configure the FB with the desired parameter. The method can be either called once at the beginning or cyclically. Once added, parameter is managed by the FB. Parameters that are not added, are useless.

Input	Data Type	Description
pstParam	POINTER TO <a href="#">ST_ParamType_Common</a>	Pointer to the parameter structure.

### DELETE\_PARAM (Method)

This method removes a previously added *pstParam* from the managed parameters (if present).

Input	Data Type	Description
pstParam	POINTER TO <a href="#">ST_ParamType_Common</a>	Pointer to the parameter structure.



### GET\_ABORTCODE\_MESSAGE (Method)

This method returns the message related to the specified *udiAbortInfoID*. The function will build the output message with the given *wIndex* and *bSubIndex*.

Input	Data Type	Description
wIndex	WORD	Identification of the object index
bSubIndex	BYTE	Identification of the object sub index
udiAbortInfoID	UDINT	Error identification

Return Data Type	Description
STRING(100)	Description of the error



## 7.2 Functions

This chapter contains the following topics:

Chapter	Chapter name	Page
<a href="#">7.2.1</a>	<a href="#">GetAppWarningMessage</a>	<a href="#">89</a>
<a href="#">7.2.2</a>	<a href="#">GetFaultMessage_ACTIVE_AGILE</a>	<a href="#">89</a>
<a href="#">7.2.3</a>	<a href="#">GetFaultMessage_AXIA</a>	<a href="#">90</a>
<a href="#">7.2.4</a>	<a href="#">GetWarningMessage</a>	<a href="#">90</a>
<a href="#">7.2.5</a>	<a href="#">GetWarningDiagnostic_Axia</a>	<a href="#">91</a>



## 7.2.1 GetAppWarningMessage

### Overview

Type:	Function
Available as of:	v. 3.1.2.0

### Description

This function returns the Application Warning message related to the specified Application warning code (displayed in *P. 273*). The output string will concatenate the messages of the active application warnings in a single string. Works with ACUx10, ANG, AGL application warnings codes.

Input	Data Type	Description
uiAppWarningCode	UINT	Value of the actual warning code

Return Data Type	Description
STRING	Application warning message

## 7.2.2 GetFaultMessage\_ACTIVE\_AGILE

### Overview

Type:	Function
Available as of:	v. 3.1.2.0

### Description

This function returns the Fault message related to the specified Fault code (displayed in *P. 259*). Works with ACUx10, ANG, AGL fault codes.

Input	Data Type	Description
uiFaultCode	UINT	Value of the actual fault code

Return Data Type	Description
STRING	fault message



## 7.2.3 GetFaultMessage\_AXIA

### Overview

Type:	Function
Available as of:	v. 3.1.2.0

### Description

This function returns the Fault message related to the specified Fault code (displayed in *obj. 0x4010.2*). Works with AXV, AXM fault codes.

Input	Data Type	Description
uiFaultCode	UINT	Value of the actual fault code

Return Data Type	Description
STRING	fault message

## 7.2.4 GetWarningMessage

### Overview

Type:	Function
Available as of:	v. 3.1.2.0

### Description

This function returns the Warning message related to the specified Warning code (displayed in *P. 269*). The output string will concatenate the messages of the active warnings in a single string. Works with ACUx10, ANG, AGL warnings codes.

Input	Data Type	Description
uiWarningCode	UINT	Value of the actual warning code

Return Data Type	Description
STRING	warning message



## 7.2.5 GetWarningDiagnostic\_Axia

### Overview

Type:	Function
Available as of:	v. 3.1.2.0

### Description

This function returns the complete Warning diagnostic related to the specified Warning groups and codes (displayed by *obj. 0x4015*). See [AM WarningDiagnosticType\\_Axia](#).

Input	Data Type	Description
i_usiWarnignGroups	USINT	Value of the actual warning groups that reported a warning ( <i>obj. 0x4015.1</i> )
i_udiPowerNumber	UDINT	Value of the actual Power warning code ( <i>obj. 0x4015.2</i> )
i_udiCommunicationNumber	UDINT	Value of the actual Communication warning code ( <i>obj. 0x4015.3</i> )
i_udiApplicationNumber	UDINT	Value of the actual Application warning code ( <i>obj. 0x4015.4</i> )
i_udiEncoderNumber	UDINT	Value of the actual Encoder warning code ( <i>obj. 0x4015.5</i> )

Return Data Type	Description
<a href="#">AM WarningDiagnosticType_Axia</a>	Warning diagnostic structure



## 8 Data Unit Types

This part contains the following chapters:

Chapter	Chapter name	Page
<a href="#">8.1</a>	<a href="#">Enumerations</a>	<a href="#">93</a>
<a href="#">8.2</a>	<a href="#">Structures</a>	<a href="#">106</a>



## 8.1 Enumerations

This chapter contains the following topics:

Chapter	Enumeration	Page
<a href="#">8.1.1</a>	<a href="#">AM_AxisMgrStatusEnum</a>	<a href="#">94</a>
<a href="#">8.1.2</a>	<a href="#">AM_ChangeProfileBehaviorEnum</a>	<a href="#">95</a>
<a href="#">8.1.3</a>	<a href="#">AM_ControllerMode</a>	<a href="#">95</a>
<a href="#">8.1.4</a>	<a href="#">AM_DataTypeEnum</a>	<a href="#">96</a>
<a href="#">8.1.5</a>	<a href="#">AM_DriveModelEnum</a>	<a href="#">96</a>
<a href="#">8.1.6</a>	<a href="#">AM_EndOfPositionBehaviorEnum</a>	<a href="#">97</a>
<a href="#">8.1.7</a>	<a href="#">AM_ErrorSourceEnum</a>	<a href="#">97</a>
<a href="#">8.1.8</a>	<a href="#">AM_FBErrorEnum</a>	<a href="#">98</a>
<a href="#">8.1.9</a>	<a href="#">AM_FieldbusEnum</a>	<a href="#">98</a>
<a href="#">8.1.10</a>	<a href="#">AM_HomingMethodsEnum</a>	<a href="#">99</a>
<a href="#">8.1.11</a>	<a href="#">AM_NewCommandFailedBehaviorEnum</a>	<a href="#">101</a>
<a href="#">8.1.12</a>	<a href="#">AM_PowerOffOptionEnum</a>	<a href="#">102</a>
<a href="#">8.1.13</a>	<a href="#">AM_PosTypeEnum</a>	<a href="#">103</a>
<a href="#">8.1.14</a>	<a href="#">AM_TouchProbeModeEnum</a>	<a href="#">103</a>
<a href="#">8.1.15</a>	<a href="#">AM_UserErrorEnum</a>	<a href="#">104</a>
<a href="#">8.1.16</a>	<a href="#">EN_State_CAN_ETC</a>	<a href="#">105</a>



## 8.1.1 AM\_AxisMgrStatusEnum

### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

The enumeration contains the status that the axis can reach according to the AxisManager state diagram (see chapter [5.1](#)). The enumeration is given as an output in each type of Axis Manager.

### Enumeration Elements

Name	Value	Description
Unknown	16#0001	Axis has not been initialized yet.
ErrorStop	16#0002	The axis is in error state. The possible causes are listed in <a href="#">AM_ErrorSourceEnum</a> .
Init	16#0004	Axis is initializing. This state will bring the axis from <i>Unknown</i> to <i>Disabled</i> state.
Disabled	16#0008	Axis not powered.
Standstill	16#0010	Axis powered and standstill
Homing	16#0020	Homing procedure is being carried out
Position	16#0040	The axis is performing the positioning with the given target data.
Velocity	16#0080	The axis is moving endless with the given target data.
Synchronized	16#0100	The axis is Synchronized with a Master Axis (i.e. in Gear, GearInPos or in Cam).
Stopping	16#0200	The axis is stopping.
EmergencyStop	16#0400	The axis is performing the emergency stop triggered by the <i>i_xQuickStop</i> input.
CN	16#0800	The axis is controlled by a CN kinematic. This state is reachable only thanks to a <i>AxisMgr_Group_CN_xx</i> .
Jog	16#1000	The axis is controlled in Jog mode. Specific Boolean are provided to activate the function, move forward and backward.
Torque	16#2000	The axis is torque controlled.



## 8.1.2 AM\_ChangeProfileBehaviorEnum

### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

The enumeration is relevant when using Profile Position function functions by AxisMgr "Light". It defines the behavior when a new Positioning command is triggered by the user. The selection finds its definition according to the CiA DSP402 in mode of operation 1 - Profile Position Mode.

This enumeration is relevant when a Positioning command is triggered during another one, defining a queue of movements.

### Enumeration Elements

Name	Value	Description
Immediately	0	When a Positioning is in progress and a new Positioning command is triggered, the new position profile (target position, velocity, ramps) is processed immediately
Immediate_OnSetPoint	1	When a Positioning is in progress and a new Positioning command is triggered, the axis is driven to the current target position at current speed. As soon as the position is reached, the new position profile is taken over.
Stop_OnSetPoint	2	When a Positioning is in progress and a new Positioning command is triggered, the axis is driven to the current target position and it will be stopped at. After the position is reached, the new position profile is taken over

## 8.1.3 AM\_ControllerMode

### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

It contains the allowed Mode of Operation. Relevant for the selection of the target controller mode when using the Set Controller Mode function ([AxisMgr\\_SM](#)).

### Enumeration Elements

Name	Value	Description
SMC_default	0	Default controller mode set for the axis (if specified)
SMC_torque	1	CST mode
SMC_velocity	2	CSV mode
SMC_position	3	CSP mode



## 8.1.4 AM\_DataTypeEnum

### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

The enumeration contains the available data types managed by the parameter generic access functionalities. The data type selected via this enumeration define the length of the information that are transferred/requested (e.g.: via acyclic data).

### Enumeration Elements

Name	Value	Description
_BYTE	1	1 Byte data, unsigned (0..255)
_WORD	2	2 Byte data, unsigned (0..65536)
_DWORD	3	4 Byte data, unsigned (0..4294967295)
_INT	4	2 Byte data, signed (-32768..32767)
_UINT	5	2 Byte data, unsigned (0..65535)
_SINT	6	1 Byte data, signed (-128..127)
_USINT	7	1 Byte data, unsigned (0..255)
_DINT	8	4 Byte data, signed (-2147483648..2147483647)
_UDINT	9	4 Byte data, unsigned (0..4294967295)
_LINT	10	8 Byte data, signed ( $-2^{63} .. 2^{63} - 1$ )
_ULINT	11	8 Byte data, unsigned ( $0 .. 2^{64} - 1$ )
_STRING	12	32 + 1 char
_REAL	13	4 Byte data, signed (-3.402823E+38 .. 3.402823E+38)

## 8.1.5 AM\_DriveModelEnum

### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

The enumeration establishes the inverter model detected by the Axis Mgr itself and it is used to correctly manage the differences between the Bonfiglioli's inverters and drives. Internal usage only.

### Enumeration Elements

Name	Value	Description
Unknown	-1	Drive not recognized or not defined
ANG	8	Active Next Generation
ACU	5	Active Cube
AGL	1	AgilE (not relevant for AxisMgr_SM)
AXIA	12	Axia (Vert or Move)



DGM	50	Decentralized Gear Motor (not relevant for AxisMgr_SM)
iBMD	60	iBMD

### 8.1.6 AM\_EndOfPositionBehaviorEnum

#### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

#### Description

It defines the ending state after the execution of a positioning movement. The user, by means of this enumeration, can choose to end Position function in *Standstill* state or to remain in *Positioning*. In the last case, the user can simply update target values without triggering the function again, and the axis will move towards it.

#### Enumeration Elements

Name	Value	Description
GoInStandstill	0	The axis ends the movement and reaches the <i>Standstill</i> state. <i>q_xCmdDone</i> can be evaluated to establish the end of the positioning.
RemainInPositioning	1	The axis remains in <i>Position</i> state. In this way the user can update the target parameters (e.g.: <i>IrTargetPosition</i> ) to trigger new movements, avoiding to set <i>i_xLaunchCmd</i> again. In this case <i>q_xCmdDone</i> is not relevant and will not be set at each position done, <i>q_xInPos</i> should be used instead.

### 8.1.7 AM\_ErrorSourceEnum

#### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

#### Description

The enumeration indicates the possible source of error that the Axis Mgr can experience.

#### Enumeration Elements

Name	Value	Description
No_Error	0	No error occurred
FB_Error	1	The error has been raised by a Tc FB or by a custom function. When using <a href="#">AxisMgr_SM</a> the univoke ErrorID is defined by the TwinCAT environment (NC Error List). When using an AxisMgr "Light", the meaning of the error is defined by the <a href="#">AM_FBErrorEnum</a> list.
User_Error	2	The error has been triggered by the user (e.g. because of a wrong parameter value or a command in an invalid state).
Drive_Error	3	A drive error has occurred
Comm_Error	4	A communication error has occurred (e.g. acyclic request failed).



## 8.1.8 AM\_FBErrorEnum

### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

Only relevant for AxisMgr "Light". The *q\_astErrorDiagnostic[x].udiErrorID* should be evaluated referring to this list only in the case *q\_astErrorDiagnostic[x].eErrorSource = FB\_Error*. The error is generated by the function itself.

### Enumeration Elements

Name	Value	Description
NO_ERROR	0	No error occurred.
READY_TO_SWITCH_ON_TIMEOUT	1	When trying to power the axis, the drive was not ready to be switched on in the expected time from when the voltage was enabled.
SWITCHED_ON_TIMEOUT	2	When trying to power the axis, the drive was not able to switch on in the expected time from when it was ready to switch on.
OPERATION_ENABLED_TIMEOUT	3	When trying to power the axis, the drive was not ready to get standstill in the expected time from when it was switched on.
TIMEOUT_RESET	4	When trying to acknowledge the fault, the drive was not able to be restored in the expected time.
TIMEOUT_MODE_OF_OPERATION	5	The function was not able to set the required mode of operation in the expected time
HOMING_ERROR	6	When during an home run, the drive has reported an error.
MAINS_SUPPLY_MISSING	7	When trying to power the axis, the drive does not have the mains supply.
UNABLE_TO_GET_FAULT_CODE	8	Axis Mgr was not able to get the actual fault code reported by the drive.

## 8.1.9 AM\_FieldbusEnum

### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

The enumeration indicates the selection of the fieldbus that is supposed to be used by the generic access in SDO. Depending on the selection (CANOpen or EtherCAT), the FB behaves accordingly.

### Enumeration Elements

Name	Value	Description
EtherCAT	0	The target remote device is an EtherCAT device.
CANOpen	1	The target remote device is a CANOpen device.



## 8.1.10 AM\_HomingMethodsEnum

### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

The enumeration groups all the possible choices for the Homing procedure that are supported by Bonfiglioli's drive (refer to the Application manual Positioning of the used drive for further information). The homing types are based on the CANOpen specification DSP402.

No.	Main destination		Fine destination (Ref. signal)	Limit Switch ?	
1	Left	limit switch	Ref. signal right	Left limit switch	
2	Right		Ref. signal left	Right limit switch	
3	Negative	home switch	Ref. signal left	Without limit switch	
4			Ref. signal right		
5	Positive		Ref. signal right		
6			Ref. signal left		
7	Left edge	home switch	Ref. signal left	Right limit switch	
8	Right edge		Ref. signal right		
9			Ref. signal left		
10			Ref. signal right		
11	Right edge		Ref. signal right	Left limit switch	
12	Left edge		Ref. signal left		
13			Ref. signal right		
14	Left edge		Ref. signal left		
15	Reserved				
16	Reserved				
17	Left	limit switch	Falling edge	Left limit switch	
18	Right		Falling edge	Right limit switch	
19	Negative	home switch	Falling edge	Without limit switch	
20			Rising edge		
21	Positive		Falling edge		
22			Rising edge		
23	Left edge	home switch	Falling edge	Right limit switch	
24	Right edge		Rising edge		
25			Rising edge		
26			Falling edge		
27	Right edge		Falling edge	Left limit switch	
28	Left edge		Rising edge		
29			Rising edge		
30	Left edge		Falling edge		
31	Reserved				
32	Reserved				
33	Left	ref. signal			
34	Right				
35	Current	position			

### Enumeration Elements

Name	Value	Description
No_Homing	0	No homig; the current position value is not changed. The current position value is the value saved upon last disconnection of power supply.
Neg_LimitSwitch_And_RefSignal	1	Homing to negative HW limit switch with detection of encoder ref. signal.



Pos_LimitSwitch_And_RefSignal	2	Homing to positive HW limit switch with detection of encoder ref. signal.
Pos_HomeSwitch_RefSignal_LeftOfEdge	3	Homing to positive home switch with detection of encoder ref. signal. Home position is the first encoder ref. signal to the left of the edge of the home switch signal.
Pos_HomeSwitch_RefSignal_RightOfEdge	4	Homing to positive home switch with detection of encoder ref. signal. Home position is the first encoder ref. signal to the right of the edge of the home switch signal.
Neg_HomeSwitch_RefSignal_RightOfEdge	5	Homing to negative home switch with detection of encoder ref. signal. Home position is the first encoder ref. signal to the left of the edge of the home switch signal.
Neg_HomeSwitch_RefSignal_LeftOfEdge	6	Homing to negative home switch with detection of encoder ref.signal. Home position is the first encoder ref. signal to the left of the edge of the home switch signal.
Pos_LimitSwitch_RefSignal_LeftOfLeftEdgeOfHomeSwitch	7	Homing to home switch with detection of encoder ref. signal. Homing direction positive (clockwise). Reversal of direction of rotation when positive HW limit switch is reached. Home position is the first encoder ref. signal to the left or right of the left or right edge of the home signal switch.
Pos_LimitSwitch_RefSignal_RightOfLeftEdgeOfHomeSwitch	8	
Pos_LimitSwitch_RefSignal_LeftOfRightEdgeOfHomeSwitch	9	
Pos_LimitSwitch_RefSignal_RightOfRightEdgeOfHomeSwitch	10	
Neg_LimitSwitch_RefSignal_LeftOfLeftEdgeOfHomeSwitch	11	Homing to home switch with detection of encoder ref. signal. Homing direction negative (anticlockwise). Reversal of direction of rotation when negative HW limit switch is reached. Home position is the first encoder ref. signal to the left or right of the left or right edge of the home signal switch.
Neg_LimitSwitch_RefSignal_RightOfLeftEdgeOfHomeSwitch	12	
Neg_LimitSwitch_RefSignal_LeftOfRightEdgeOfHomeSwitch	13	
Neg_LimitSwitch_RefSignal_RightOfRightEdgeOfHomeSwitch	14	
17..30: like 1..14 but without encoder ref. signal		
Neg_LimitSwitch	17	Homing to negative HW limit switch
Pos_LimitSwitch	18	Homing to positive HW limit switch
Pos_HomeSwitch_LeftOfEdge	19	Homing to positive home switch. Home position is at the left of the edge of the home switch signal.
Pos_HomeSwitch_RightOfEdge	20	Homing to positive home switch. Home position is at the right of the edge of the home switch signal.
Neg_HomeSwitch_RightOfEdge	21	Homing to negative home switch. Home position is at the right of the edge of the home switch signal.



Neg_HomeSwitch_LeftOfEdge	22	Homing to negative home switch. Home position is at the left of the edge of the home switch signal.
Pos_LimitSwitch_LeftOfLeftEdgeOfHomeSwitch	23	Homing to home switch. Homing direction positive (clockwise). Reversal of direction of rotation when positive HW limit switch is reached. Home position is at the left or right of the left or right edge of the home switch signal.
Pos_LimitSwitch_RightOfLeftEdgeOfHomeSwitch	24	
Pos_LimitSwitch_LeftOfRightEdgeOfHomeSwitch	25	
Pos_LimitSwitch_RightOfRightEdgeOfHomeSwitch	26	
Neg_LimitSwitch_LeftOfLeftEdgeOfHomeSwitch	27	Homing to home switch. Homing direction negative (anticlockwise). Reversal of direction of rotation when negative HW limit switch is reached. Home position is at the left or right of the left or right edge of the home switch signal.
Neg_LimitSwitch_RightOfLeftEdgeOfHomeSwitch	28	
Neg_LimitSwitch_LeftOfRightEdgeOfHomeSwitch	29	
Neg_LimitSwitch_RightOfRightEdgeOfHomeSwitch	30	
RefSignal_LeftOfActPosition	33	Home position is the first encoder ref. signal in negative (operation mode 33) or positive (operation mode 34) direction.
RefSignal_RightOfActPosition	34	
CurrentPosition	35	Current position is home position. Final position is taken over as actual position value.

### 8.1.11 AM\_NewCommandFailedBehaviorEnum

#### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

#### Description

It defines the behavior of an AxisMgr's function if failed during the STARTING state (refer to chapter [5.3](#)). The selection made in this enumeration will be stored at the end of *Init* state. Change of this value afterwards won't have any effect.

#### Enumeration Elements

Name	Value	Description
ErrorStop	0	The new command, that resulted in an error, puts the AxisMgr in <i>ErrorStop</i> state and any ongoing movement will be stopped accordingly.
Refuse	1	The failure on the request of a new command will not lead to a stop of the ongoing function. AxisMgr will only report the error in the <i>q_stErrorDiagnostic</i> and <i>q_xFault</i> will be set to <i>TRUE</i> . The ongoing function won't be aborted as if the new failed command was never been requested.



## 8.1.12 AM\_PowerOffOptionEnum

### Overview

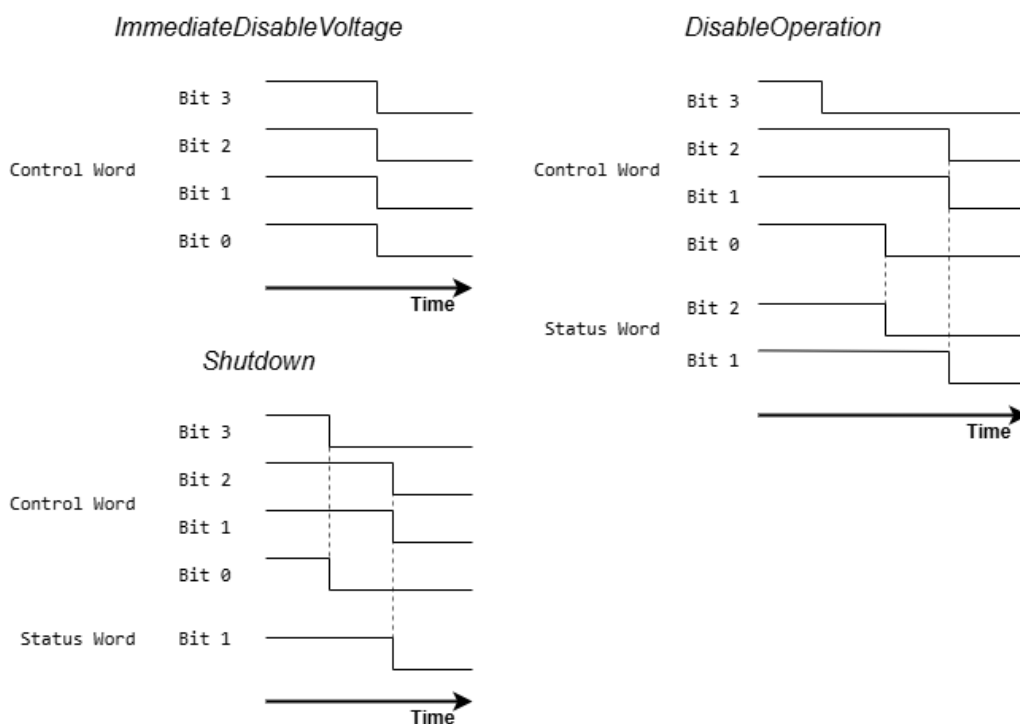
Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

It defines the behavior when a power-off command has been requested (i.e.:  $\bar{I\_xPower} \Rightarrow FALSE$ ). Depending on the selection, the transitions that brings the drive to state Disable Voltage are made differently. See also chapter [5.2.3.2](#).

### Enumeration Elements

Name	Value	Description
ImmediateDisableVoltage	0	Axis Mgr immediately set the Control Word to 0. In this way, the drive will immediately cut-off the voltage to the motor, without waiting any time. If the motor was running, it will be probably stop in a free-wheeling manner. In case of vertical axis with holding brake, the axis may fall without control as the brake takes a while to be effectively closed.
DisableOperation	1	Axis Mgr immediately Disables the operation of the drive. In this case, the value set in <i>obj. 0x605C Disable Operation Option Code</i> (Axia) or <i>P.392 State Transition 5</i> (ACU, ANG, AGL) is relevant. Once the motor is not controlled by the drive anymore, Axis Mgr will Disable the Voltage. In this way the user can manage the holding brake in the proper way, keeping the motor standstill while the brake is closing.
Shutdown	2	Axis Mgr immediately Shutdown the drive. In this case, the value set in <i>obj. 0x605B Shutdown Option Code</i> (Axia) is relevant. Once the motor is not controlled by the drive anymore, Axis Mgr will Disable the Voltage. In this way the user can manage the holding brake in the proper way, keeping the motor standstill while the brake is closing.





### 8.1.13 AM\_PosTypeEnum

#### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

#### Description

The enumeration defines the positioning mode, either absolute or relative.

#### Enumeration Elements

Name	Value	Description
Absolute	0	Target position relates to the fixed reference position which is determined by a homing operation. An absolute distance is covered, referred to the reference position.
Relative	1	A relative positioning operation relates always to the current position of the axis. New target position = current position + relative distance.

### 8.1.14 AM\_TouchProbeModeEnum

#### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

#### Description

When using Touch Probe function, it defines the operation behavior of the TP channel.

#### Enumeration Elements

Name	Value	Description
TriggerFirstEvent_SingleShot	0	Since TP channel was enabled, Touch Probe function detects the next TP event only (both negative and positive edges). Further TP input events won't be evaluated. The user should disable TP channel and re-enable again to start another evaluation.
Continuos	1	As long as TP channel was enabled, Touch Probe continuously evaluates the TP channel inputs. Information are updated at each TP event detected.



## 8.1.15 AM\_UserErrorEnum

### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

### Description

The enumeration lists the possible causes when an *User Error* occurs. The *q\_astErrorDiagnostic[x].udiErrorID* should be evaluated referring to this list only in the case *q\_astErrorDiagnostic[x].eErrorSource = User\_Error*.

### Enumeration Elements

Name	Value	Description
NO_ERROR	0	No error occurred.
POINTER_TO_AXIS_REF_EQUAL_TO_0	1	Reference to AXIS_REF not assigned or invalid.
INVALID_ACC_DEC_VALUE	2	Invalid acceleration and/or deceleration value. Value equal or less than 0 or too big.
INVALID_SPEED	3	Invalid speed value. Value equal or less than 0 or too big.
INVALID_POSITION	4	Invalid Position value.
COMMAND_NOT_AVAILABLE_IN_THIS_STATE	5	The given command is not possible in the current axis state. Refer to chapter 5.1 state diagram to have an overview of the possible command in each state.
TIMEOUT	6	Operation timed out
UNKNOWN_INVERTER_MODEL	7	Inverter model not recognized during Init. Third party inverters are not supported.
POINTER_TO_MASTER_REF_EQUAL_TO_0	8	Reference to AXIS_REF not assigned or invalid in <i>i_pstMasterAxis</i> .
INVALID_GEAR_RATIO	9	Invalid gear ratio. Numerator and/or Denominator equal to 0.
INVALID_CAM_REF	10	Invalid address to MC_CAM_REF.
AXIS_NOT_READY_FOR_MOTION	11	Axis not ready for motion. Probably hardware has not been released.
NOT_IN_OPERATIONAL_STATE	12	Axis is not operational. Check communication.
MISSING_REMOTE_DEVICE	13	Reference to remote device <i>i_itfRemoteDevice</i> not provided.
COMMAND_NOT_AVAILABLE_IN_THIS_CONFIGURATION	14	The requested command is not available with the current drive configuration (e.g.: Position not possible in speed-controlled configurations).
INVALID_FIELDBUS	15	Invalid Fieldbus
AXIS_NOT_READY_FOR_CN	16	When requesting CN function, the axis was not able to be set. A further error entry with the reason is probably displayed.
INVALID_POINTER_TO_QUEUE	17	When requesting CN function, the address to the path queue is missing.
SWITCH_ON_DISABLED	18	Switch On Disabled



INVALID_TARGET_TORQUE	19	The provided target torque is out of the valid range
INVALID_TORQUE_SLOPE	20	The provided torque slope is out of the valid range
CONTROLLER_MODE_NOT_SUPPORTED	21	The selected controller mode is not supported by the drive.

### 8.1.16 EN\_State\_CAN\_ETC

#### Overview

Type:	Enumeration
Available as of:	v. 3.1.2.0

#### Description

The enumeration lists the possible state that the CANOpen or EtherCAT slave device can reach.

#### Enumeration Elements

Name	Value	Description
INIT	0	Standard NMT state. Refer to CiA specification DS 301 for further information.
BOOT	1	
PRE_OPERATIONAL	2	
SAFE_OPERATIONAL	3	
OPERATIONAL	4	
UNKNOWN	5	



## 8.2 Structures

This chapter contains the following topics:

Chapter	Structure	Page
<a href="#">8.2.1</a>	<a href="#">AM_CamDataType</a>	<a href="#">107</a>
<a href="#">8.2.2</a>	<a href="#">AM_CommandDataType_ACTIVE</a>	<a href="#">108</a>
<a href="#">8.2.3</a>	<a href="#">AM_CommandDataType_AGL</a>	<a href="#">109</a>
<a href="#">8.2.4</a>	<a href="#">AM_CommandDataType_Axia</a>	<a href="#">109</a>
<a href="#">8.2.5</a>	<a href="#">AM_CommandDataType_SM</a>	<a href="#">110</a>
<a href="#">8.2.6</a>	<a href="#">AM_ErrorDiagnosticType</a>	<a href="#">111</a>
<a href="#">8.2.7</a>	<a href="#">AM_GearDataType</a>	<a href="#">111</a>
<a href="#">8.2.8</a>	<a href="#">AM_HomingDataType</a>	<a href="#">112</a>
<a href="#">8.2.9</a>	<a href="#">AM_InitDataType</a>	<a href="#">113</a>
<a href="#">8.2.10</a>	<a href="#">AM_InitParameterType</a>	<a href="#">114</a>
<a href="#">8.2.11</a>	<a href="#">AM_JogDataType</a>	<a href="#">114</a>
<a href="#">8.2.12</a>	<a href="#">AM_PositioningDataType</a>	<a href="#">115</a>
<a href="#">8.2.13</a>	<a href="#">AM_PowerDataType</a>	<a href="#">116</a>
<a href="#">8.2.14</a>	<a href="#">AM_ProfilePositionDataType_ACTIVE</a>	<a href="#">117</a>
<a href="#">8.2.15</a>	<a href="#">AM_ProfilePositionDataType_Axia</a>	<a href="#">120</a>
<a href="#">8.2.16</a>	<a href="#">AM_ProfileTorqueDataType_Axia</a>	<a href="#">122</a>
<a href="#">8.2.17</a>	<a href="#">AM_ProfileVelocityDataType_ACTIVE</a>	<a href="#">123</a>
<a href="#">8.2.18</a>	<a href="#">AM_ProfileVelocityDataType_Axia</a>	<a href="#">124</a>
<a href="#">8.2.19</a>	<a href="#">AM_QuickStopDataType</a>	<a href="#">124</a>
<a href="#">8.2.20</a>	<a href="#">AM_SetPosDataType</a>	<a href="#">125</a>
<a href="#">8.2.21</a>	<a href="#">AM_SetTorqueDataType</a>	<a href="#">125</a>
<a href="#">8.2.22</a>	<a href="#">AM_StopDataType</a>	<a href="#">126</a>
<a href="#">8.2.23</a>	<a href="#">AM_SuperImposedDataType</a>	<a href="#">127</a>
<a href="#">8.2.24</a>	<a href="#">AM_TouchProbeDataType</a>	<a href="#">128</a>
<a href="#">8.2.25</a>	<a href="#">AM_TouchProbeStatusType</a>	<a href="#">128</a>
<a href="#">8.2.26</a>	<a href="#">AM_TPChannelStatusType</a>	<a href="#">129</a>
<a href="#">8.2.27</a>	<a href="#">AM_TPChannelType</a>	<a href="#">130</a>
<a href="#">8.2.28</a>	<a href="#">AM_VelocityModeDataType_ACTIVE</a>	<a href="#">131</a>
<a href="#">8.2.29</a>	<a href="#">AM_VelocityModeDataType_AGL</a>	<a href="#">132</a>
<a href="#">8.2.30</a>	<a href="#">AM_VelocityModeDataType_Axia</a>	<a href="#">133</a>
<a href="#">8.2.31</a>	<a href="#">AM_VeloDataType_SM</a>	<a href="#">133</a>
<a href="#">8.2.32</a>	<a href="#">AM_WarningDiagnosticType_ACTIVE</a>	<a href="#">134</a>
<a href="#">8.2.33</a>	<a href="#">AM_WarningDiagnosticType_Axia</a>	<a href="#">135</a>
<a href="#">8.2.34</a>	<a href="#">AM_WarningGroupType_Axia</a>	<a href="#">136</a>
<a href="#">8.2.35</a>	<a href="#">ST_ParamType_Common</a>	<a href="#">137</a>



## 8.2.1 AM\_CamDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Camming command. With this data the command is being configured; nevertheless, the cam plate must be defined externally (e.g.: compiling the MC\_CAM\_REF object. Refer to the related documentation for further information about cam plates and its definition). Refer also to chapter [5.3.2.7](#).

### Structure Elements

Name	Data Type	Default	Description
pCamRef	POINTER TO <a href="#">MC_CAM_REF</a>		Pointer to the MC_CAM_REF. Must be instantiated and configured externally by the user.
xPeriodic	BOOL	TRUE	Selection of CAM management in periodic mode: → FALSE: ONE SHOT Cam → TRUE: periodic Cam
xMasterAbsolute	BOOL	TRUE	Selection of CAM management, assuming the position of the master as an absolute quota → FALSE: Relative master position. The current position of the master corresponds to the start of the cam → TRUE: Absolute master position. The current position of the master corresponds to the same position in the cam profile
lrMasterOffset	LREAL	0.0	Defines an offset of the master, in order to "move" the cam in both directions, on the X axis
lrMasterScaling	LREAL	1.0	Defines a master scale factor, in order to "stretch" the cam in both directions, on the X axis
xSlaveAbsolute	BOOL	TRUE	Selection of CAM management, assuming the position of the controlled axis as an absolute quote → FALSE: Relative slave position. The current position of the controlled axis corresponds to the start of the cam → TRUE: Absolute slave position. The current position of the controlled axis corresponds to the same position in the cam profile
lrSlaveOffset	LREAL	0.0	Defines an offset of the controlled axis, in order to "move" the cam in both directions, on the Y axis
lrSlaveScaling	LREAL	1.0	Defines a scale factor of the controlled axis, in order to "stretch" the cam in both directions, on the Y axis
enActivationMode	Tc2_MC2_Camming. MC_StartMode		Parameter that defines how the CAM is engaged



stOptions	Tc2_MC2_Camming. ST_CamInOptions		Further options
udiCamID	UDINT	1	Defines the CAM identifier that should be assigned. The value must be between 1 and 255

## 8.2.2 AM\_CommandDataType\_ACTIVE

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure is a container for all the data structure relevant for the related function. This data type contains the input data for the functions available in AxisMgr ACUx10 <fieldbus> and AxisMgr ANG <fieldbus>.

### Structure Elements

Name	Data Type	Description
stInit	<a href="#">AM_InitDataType</a>	Data relevant for Init function.
stPower	<a href="#">AM_PowerDataType</a>	Data relevant for a power-off command.
stProfilePos	<a href="#">AM_ProfilePositionDataType ACTIVE</a>	Data relevant for Profile Position function.
stProfileVelocity	<a href="#">AM_ProfileVelocityDataType ACTIVE</a>	Data relevant for Profile Velocity function.
stVelocityMode	<a href="#">AM_VelocityModeDataType ACTIVE</a>	Data relevant for Velocity Mode function.
stHoming	<a href="#">AM_HomingDataType</a>	Data relevant for Homing function.
stStop	<a href="#">AM_StopDataType</a>	Data relevant for Stop function.
stQuickStop	<a href="#">AM_QuickStopDataType</a>	Data relevant for Quick Stop function.
stTouchProbe	<a href="#">AM_TouchProbeDataType</a>	Data relevant for TouchProbe function.



## 8.2.3 AM\_CommandDataType\_AGL

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure is a container for all the data structure relevant for the related function. This data type contains the input data for the functions available in AxisMgr AGL <fieldbus>.

### Structure Elements

Name	Data Type	Description
stInit	<a href="#">AM_InitDataType</a>	Data relevant for Init function.
stPower	<a href="#">AM_PowerDataType</a>	Data relevant for a power-off command.
stVelocityMode	<a href="#">AM_VelocityModeDataType_AGL</a>	Data relevant for Velocity Mode function.
stStop	<a href="#">AM_StopDataType</a>	Data relevant for Stop function.
stQuickStop	<a href="#">AM_QuickStopDataType</a>	Data relevant for Quick Stop function.

## 8.2.4 AM\_CommandDataType\_Axia

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure is a container for all the data structure relevant for the related function. This data type contains the input data for the functions available in AxisMgr AXV <fieldbus> and AxisMgr AXM <fieldbus>.

### Structure Elements

Name	Data Type	Description
stInit	<a href="#">AM_InitDataType</a>	Data relevant for Init function.
stPower	<a href="#">AM_PowerDataType</a>	Data relevant for a power-off command.
stProfilePos	<a href="#">AM_ProfilePositionDataType_Axia</a>	Data relevant for Profile Position function.
stProfileVelocity	<a href="#">AM_ProfileVelocityDataType_Axia</a>	Data relevant for Profile Velocity function.
stVelocityMode	<a href="#">AM_VelocityModeDataType_Axia</a>	Data relevant for Velocity Mode function.
stHoming	<a href="#">AM_HomingDataType</a>	Data relevant for Homing function.
stStop	<a href="#">AM_StopDataType</a>	Data relevant for Stop function.
stQuickStop	<a href="#">AM_QuickStopDataType</a>	Data relevant for Quick Stop function.
stTouchProbe	<a href="#">AM_TouchProbeDataType</a>	Data relevant for TouchProbe function.
stTorque	<a href="#">AM_ProfileTorqueDataType_Axia</a>	Data relevant for Profile Torque function.



## 8.2.5 AM\_CommandDataType\_SM

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure is a container for all the data structure relevant for the related function. This data type contains the input data for the functions available in [AxisMgr SM](#).

### Structure Elements

Name	Data Type	Description
stInit	<a href="#">AM_InitDataType</a>	Data relevant for Init function.
stVelo	<a href="#">AM_VeloDataType SM</a>	Data relevant for Velocity function.
stStop	<a href="#">AM_StopDataType</a>	Data relevant for Stop function.
stQuickStop	<a href="#">AM_QuickStopDataType</a>	Data relevant for Quick Stop function.
stPos	<a href="#">AM_PositioningDataType</a>	Data relevant for Position function.
stSetPos	<a href="#">AM_SetPosDataType</a>	Data relevant for Set Pos function.
stGearing	<a href="#">AM_GearDataType</a>	Data relevant for Gearing function.
stCamming	<a href="#">AM_CamDataType</a>	Data relevant for Camming function.
stSuperImpos	<a href="#">AM_SuperImposedDataType</a>	Data relevant for Super Imposed function.
stTouchProbe	<a href="#">AM_TouchProbeDataType</a>	Data relevant for Touch Probe function.
stJog	<a href="#">AM_JogDataType</a>	Data relevant for Jog function
stSetTorque	<a href="#">AM_SetTorqueDataType</a>	Data relevant for Set Torque (CST) function
enControllerMode	<a href="#">AM_ControllerMode</a>	Data relevant for Controller Mode function



## 8.2.6 AM\_ErrorDiagnosticType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

The structure contains the relevant information about the pending errors.

### Structure Elements

Name	Data Type	Description
eErrorSource	<a href="#">AM_ErrorSourceEnum</a>	It describes the root cause of the error. The possible values can be: <ul style="list-style-type: none"><li>→ No_Error: no error reported</li><li>→ FB_Error: the error has been reported by the FB (e.g.: Time out, command sequence issue...)</li><li>→ User_Error: the user has launched the function in the wrong way (e.g.: wrong or invalid input parameters, unsupported state...)</li><li>→ Drive_Error: the error reported a fault.</li><li>→ Comm_Error: a communication error has been reported (e.g.: SDO abort...)</li></ul>
udiErrorID	UDINT	It returns the error identification of the error. The identification is unique only within the same Error Source. <i>eErrorSource</i> and <i>udiErrorID</i> should be evaluated together.
strMessage	STRING(150)	Error message. It contains the indication of the function that reported the error.
ldtTimeStamp	LDT	The time stamp when the error has been detected. The time refers to the actual UTC time of the controller (resolution in milliseconds).

## 8.2.7 AM\_GearDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for Gearing function.

### Structure Elements

Name	Data Type	Default	Description
diGearNum	DINT	1	Gear ratio numerator used during <i>Gearing</i> function.
uiGearDen	UINT	1	Gear ratio denominator used during <i>Gearing</i> function.
lrAcceleration	LREAL		Max acceleration [units/s <sup>2</sup> ] used for ramping up in.
lrDeceleration	LREAL		Max deceleration [units/s <sup>2</sup> ] used for ramping up in.



## 8.2.8 AM\_HomingDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Homing function.

### Structure Elements

Name	Data Type	Default	Description
eMode	<a href="#">AM_HomingMethodsEnum</a>	CurrentPosition	HOME mode desired. Not relevant for virtual axis/simulation mode, Homing on current position will be always performed in this case.
IrSearchCamVelocity	LREAL		Search speed [units/sec] of the reference cam. Not relevant for virtual axis/simulation mode.
IrLeaveCamVelocity	LREAL		Zero cam [units/sec] abandonment speed. Not relevant for virtual axis/simulation mode.
IrAcceleration	LREAL		Acceleration [units/sec <sup>2</sup> ] used during movement in <i>Homing</i> . Not relevant for virtual axis/simulation mode.
IrFinalPosition	LREAL		Position assigned to the axis at the end of the <i>Homing</i> command [units]



## 8.2.9 AM\_InitDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter used during the Init procedure.

### Structure Elements

Name	Data Type	Default	Description
astPar	ARRAY[0.._AM_K. k_iParameters] OF <a href="#">AM_InitParameter Type</a> ;		Array of the parameters that should be written during <i>Init</i> state. The dimension of array can be configured in the Library Manager>Parameters. Keep the value low to optimize memory usage.
eNewCommand FailedBehavior Enum	<a href="#">AM_NewCommand FailedBehaviorEnu m</a>	ErrorStop	Behavior of the Axis Manager when a new command request fails before to get running (i.e.: due to an <i>User Error</i> occurred because of invalid input parameters). → ErrorStop: the failure on the request of a new command will stop the ongoing function and put the axis in <i>Error Stop</i> → Refuse: the failure on the request of a new command will not lead to a stop of the ongoing function. The FB will only report the error in the <i>q_stErrorDiagnostic</i> and <i>q_xFault</i> will be set to TRUE consequently This selection will apply for all the available functions and it will be taken into account at the end of <i>Init</i> , changing it afterwards won't have any effect.
xInvertDirection	BOOL		→ TRUE: drive will be set to invert the actual direction of the motor, maintaining the sign of the target/actual values unchanged. If <a href="#">AxisMgr_SM</a> is used, the behavior is similar to the <i>Invert direction</i> check box available in <i>Scaling/Mapping</i> tab of the Motion Axis, but, in this case, the inversion is not made by the drive (i.e.: Homing and limits switches may not work as expected). → FALSE: the actual inversion setting will be maintained. It will be only take into during <i>Init</i> . NOT available for iBMD. Only relevant for real physical axis. Use the <i>Invert Direction</i> checkbox in the case of Virtual Axis.



## 8.2.10 AM\_InitParameterType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

The structure contains the information that defines each parameter of *astPar* in [AM\\_InitDataType](#).

### Structure Elements

Name	Data Type	Description
stParam	<a href="#">ST_ParamType_Common</a>	Parameter structure for init writing parameter
tDelay	TIME	Time delay that the init should wait before to write the next parameter.

## 8.2.11 AM\_JogDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Jog function.

### Structure Elements

Name	Data Type	Description
lrJogVelocity	LREAL	Target velocity [units/sec] when in <i>Jog</i> mode. Negative value will reverse the movement direction. Changing the value will automatically take effect on the movement.
lrJogAcceleration	LREAL	Acceleration [units/sec <sup>2</sup> ] when in <i>Jog</i> mode. Changing the value will automatically take effect on the movement.
lrJogDeceleration	LREAL	Deceleration [units/sec <sup>2</sup> ] when in <i>Jog</i> mode. Changing the value will automatically take effect on the movement.



## 8.2.12 AM\_PositioningDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Positioning function ([AxisMgr\\_SM](#)).

### Structure Elements

Name	Data Type	Default	Description
ePositioningType	<a href="#">AM_PosTypeEnum</a>	Absolute	Selection of the positioning type: either Absolute or Relative. Changing of this value during an ongoing positioning will trigger a new positioning command of different type.
IrTargetPosition	LREAL		Target position [units] that should be reached (for absolute positioning) or Distance from the actual position (for relative positioning). Changing of this value during an ongoing positioning will trigger a new positioning command with different target.
IrTargetVelocity	LREAL		Target velocity [units/s] used during positioning (not necessarily reached). Changing of this value during an ongoing positioning will trigger a new positioning command with different target (not in the case of relative positioning of modulo axis)
IrAcceleration	LREAL		Acceleration [units/s <sup>2</sup> ] used during positioning. Changing of this value during an ongoing positioning will trigger a new positioning command with different value (not in the case of relative positioning of modulo axis)
IrDeceleration	LREAL		Deceleration [units/s <sup>2</sup> ] used during positioning. Changing of this value during an ongoing positioning will trigger a new positioning command with different value (not in the case of for relative positioning of modulo axis)
IrPositionWindow	LREAL		It defines the symmetrical window [units] around the target position where the axis should stay in order to set <i>q_xInPos</i>
uiPositionWinodwTime_ms	UINT		It defines the time delay [ms] within which the axis should remain in <i>IrPositionWindow</i> before to raise <i>q_xInPos</i>



enEndOfPositionBehavior	<a href="#">AM_EndOfPositionBehaviorEnum</a>	GoInStandstill	<p>It defines the behavior of the Axis Manager when the actual positioning movement reaches its end.</p> <ul style="list-style-type: none"><li>→ GoInStandstill: the axis ends the movement and reaches the <i>Standstill</i> state. <i>q_xCmdDone</i> can be evaluated to establish the end of the positioning.</li><li>→ RemainInPosition: the axis remains in <i>Position</i> state. In this way the user can update the target parameters (e.g.: <i>IrTargetPosition</i>) to trigger new movements, avoiding to set <i>i_xLaunchCmd</i> again. In this case <i>q_xCmdDone</i> is not relevant and will not be set at each position done, <i>q_xInPos</i> should be used instead.</li></ul>
-------------------------	--	----------------	--

### 8.2.13 AM\_PowerDataType

#### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

#### Description

This structure contains the relevant parameter for the a power-off command (AxisMgr "Light").

#### Structure Elements

Name	Data Type	Default	Description
ePowerOffOptionCode	<a href="#">AM_PowerOffOptionEnum</a>	ImmediateDisable Voltage	It defines the transition behavior of the drive for the motor power cut-off



## 8.2.14 AM\_ProfilePositionDataType\_ACTIVE

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Profile Position function (AxisMgr ACUx10 <fieldbus>, AxisMgr ANG <fieldbus>).

### Structure Elements

Name	Data Type	Default	Description
ePosType	<a href="#">AM_PosTypeEnum</a>		Selection of the positioning type: either Absolute or Relative. Changing of this value will be considered if a new position has been requested (e.g.: by means of changing profile target values).
rTargetPosition	REAL		Target position [units] that should be reached (for absolute positioning) or Distance from the actual position (for relative positioning). Changing of this value during an ongoing positioning will trigger a new positioning command with different target (not in the case of relative positioning).
rTargetVelocity	REAL		Target velocity [units/s] used during positioning (not necessarily reached). Changing of this value during an ongoing positioning will trigger a new positioning command with different target (not in the case of relative positioning).
rAcceleration	REAL		Acceleration [units/s <sup>2</sup> ] used during positioning. Changing of this value during an ongoing positioning will trigger a new positioning command with different value (not in the case of relative positioning).
rDeceleration	REAL		Deceleration [units/s <sup>2</sup> ] used during positioning. Changing of this value during an ongoing positioning will trigger a new positioning command with different value (not in the case of for relative positionin).
iRampRiseTime_ms	INT	-1	Axis Mgr will copy this value into <i>P.1176 Ramp Rise Time</i> of the drive (RAM). The value will be used to limit to jerk of the position movement, applying this time [ms] at the beginning and at the end of the acceleration phase. If it is set to -1, the value stored in the drive will take effect. It only works when <i>obj. 0x6086 Motion profile type = 3</i> .
iRampFallTime_ms	INT	-1	Axis Mgr will copy this value into <i>P.1176 Ramp Fall Time</i> of the drive (RAM). The value will be used to limit the jerk of the position movement, applying this time [ms] at the beginning and at the end of the deceleration phase. If it is set to -1, the value stored in the drive will take effect. It only works when <i>obj. 0x6086 Motion profile type = 3</i> .



rPositionWindow	REAL		It defines the symmetrical window [units] around the target position where the axis should stay in order to set <i>q_xInPos</i> . If it is greater than 0, Axis Mgr will copy the value into <i>obj. 0x6067 Position window</i> and used by the drive to establish the whether the axis is in the window. A value of 0 won't be copied into the drive, the value stored in <i>obj. 0x6067</i> will be used.
uiPositionWinodwTime_ms	UINT		It defines the time delay [ms] within which the axis should remain in <i>rPositionWindow</i> before to raise <i>q_xInPos</i> . If it is greater than 0, Axis Mgr will copy the value into <i>obj. 0x6068 Position window time</i> and used by the drive to establish whether the axis is in the window for the time set. A value of 0 won't be copied into the drive, the value stored in <i>obj. 0x6068</i> will be used.
eChangeProfileBehavior	<a href="#">AM_ChangeProfileBehaviorEnum</a>		<p>It defines the way that the target values are considered when updated:</p> <ul style="list-style-type: none"> <li>➔ Immediately: as soon as target position, velocity, acceleration and/or deceleration are updated during an ongoing positioning, the function writes the information into the drive's object. When all the information are written successfully, a new positioning command is triggered and the new target values will be applied.</li> <li>➔ Immediate_OnSetPoint: as soon as target position, velocity, acceleration and/or deceleration are updated during an ongoing positioning, the function writes the information into the drive's object. The new target values will be applied only once the previous movement reaches the target. The axis <b>won't stop</b> on the target but it will maintain the same operating speed, then the new profile position will be applied immediately "on-the-fly".</li> <li>➔ Stop_OnSetPoint: as soon as target position, velocity, acceleration and/or deceleration are updated during an ongoing positioning, the function writes the information into the drive's object. The new target values will be applied only once the previous movement reaches the target. The axis <b>will stop</b> on the target, then the new profile position will immediately start from 0 speed.</li> </ul>



enEndOfPositionBehavior	<a href="#">AM_EndOfPositionBehaviorEnum</a>	GoInStandstill	<p>It defines the behavior of the Axis Manager when the actual positioning movement reaches its end.</p> <ul style="list-style-type: none"><li>→ GoInStandstill: the axis ends the movement and reaches the <i>Standstill</i> state. <i>q_xCmdDone</i> can be evaluated to establish the end of the positioning.</li><li>→ RemainInPosition: the axis remains in <i>Position</i> state. In this way the user can update the target parameters (e.g.: <i>lrTargetPosition</i>) to trigger new movements, avoiding to set <i>i_xLaunchCmd</i> again.</li></ul> <p>In this case <i>q_xCmdDone</i> is not relevant and will not be set at each position done, <i>q_xInPos</i> should be used instead.</p>
-------------------------	--	----------------	---



## 8.2.15 AM\_ProfilePositionDataType\_Axia

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Profile Position function (AxisMgr AXV <fieldbus>, AxisMgr AXM <fieldbus>).

### Structure Elements

Name	Data Type	Default	Description
ePosType	<a href="#">AM_PosTypeEnum</a>		Selection of the positioning type: either Absolute or Relative. Changing of this value will be considered if a new position has been requested (e.g.: by means of changing profile target values).
rTargetPosition	REAL		Target position [units] that should be reached (for absolute positioning) or Distance from the actual position (for relative positioning). Changing of this value during an ongoing positioning will trigger a new positioning command with different target (not in the case of relative positioning).
rTargetVelocity	REAL		Target velocity [units/s] used during positioning (not necessarily reached). Changing of this value during an ongoing positioning will trigger a new positioning command with different target (not in the case of relative positioning).
rAcceleration	REAL		Acceleration [units/s <sup>2</sup> ] used during positioning. Changing of this value during an ongoing positioning will trigger a new positioning command with different value (not in the case of relative positioning).
rDeceleration	REAL		Deceleration [units/s <sup>2</sup> ] used during positioning. Changing of this value during an ongoing positioning will trigger a new positioning command with different value (not in the case of for relative positionin).
rPositionWindow	REAL		It defines the symmetrical window [units] around the target position where the axis should stay in order to set <i>q_xInPos</i> . If it is greater than 0, Axis Mgr will copy the value into <i>obj. 0x6067 Position window</i> and used by the drive to establish the whether the axis is in the window. A value of 0 won't be copied into the drive, the value stored in <i>obj. 0x6067</i> will be used.
uiPositionWinodwTime_ms	UINT		It defines the time delay [ms] within which the axis should remain in <i>rPositionWindow</i> before to raise <i>q_xInPos</i> . If it is greater than 0, Axis Mgr will copy the value into <i>obj. 0x6068 Position window time</i> and used by the drive to establish whether the axis is in the window for the time set. A value of 0 won't be copied into the drive, the value stored in <i>obj. 0x6068</i> will be used.



eChangeProfileBehavior	<a href="#">AM_ChangeProfileBehaviorEnum</a>		<p>It defines the way that the target values are considered when updated:</p> <ul style="list-style-type: none"><li>➔ Immediately: as soon as target position, velocity, acceleration and/or deceleration are updated during an ongoing positioning, the function writes the information into the drive's object. When all the information are written successfully, a new positioning command is triggered and the new target values will be applied.</li><li>➔ Immediate_OnSetPoint: as soon as target position, velocity, acceleration and/or deceleration are updated during an ongoing positioning, the function writes the information into the drive's object. The new target values will be applied only once the previous movement reaches the target. The axis <b>won't stop</b> on the target but it will maintain the same operating speed, then the new profile position will be applied immediately "on-the-fly".</li><li>➔ Stop_OnSetPoint: as soon as target position, velocity, acceleration and/or deceleration are updated during an ongoing positioning, the function writes the information into the drive's object. The new target values will be applied only once the previous movement reaches the target. The axis <b>will stop</b> on the target, then the new profile position will immediately start from 0 speed.</li></ul>
enEndOfPositionBehavior	<a href="#">AM_EndOfPositionBehaviorEnum</a>	GoInStandstill	<p>It defines the behavior of the Axis Manager when the actual positioning movement reaches its end.</p> <ul style="list-style-type: none"><li>➔ GoInStandstill: the axis ends the movement and reaches the <i>Standstill</i> state. <i>q_xCmdDone</i> can be evaluated to establish the end of the positioning.</li><li>➔ RemainInPosition: the axis remains in <i>Position</i> state. In this way the user can update the target parameters (e.g.: <i>IrTargetPosition</i>) to trigger new movements, avoiding to set <i>i_xLaunchCmd</i> again.</li></ul> <p>In this case <i>q_xCmdDone</i> is not relevant and will not be set at each position done, <i>q_xInPos</i> should be used instead.</p>



## 8.2.16 AM\_ProfileTorqueDataType\_Axia

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Profile Torque function (AxisMgr AXV <fieldbus>, AxisMgr AXM <fieldbus>).

### Structure Elements

Name	Data Type	Description
rTargetTorque_Nm	REAL	Target torque [Nm] used during torque control. Changes of this value during an ongoing torque control immediately affect the axis.
rTorqueSlope_Nm_s	REAL	Torque Slope [Nm/s] used during torque control. Changes of this value during an ongoing torque control immediately affect the axis.



## 8.2.17 AM\_ProfileVelocityDataType\_ACTIVE

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Profile Velocity function (AxisMgr ACUx10 <fieldbus>, AxisMgr ANG <fieldbus>).

### Structure Elements

Name	Data Type	Default	Description
rTargetVelocity	REAL		Target velocity [units/s] used during velocity. Changing of this value during an ongoing positioning will take effect.
rAcceleration	REAL		Acceleration [units/s <sup>2</sup> ] used during velocity. Changing of this value during an ongoing positioning will take effect.
rDeceleration	REAL		Deceleration [units/s <sup>2</sup> ] used during velocity. Changing of this value during an ongoing positioning will take effect.
iRampRiseTime_ms	INT	-1	Axis Mgr will copy this value into <i>P.1176 Ramp Rise Time</i> of the drive (RAM). The value will be used to limit the jerk of the velocity movement, applying this time [ms] at the beginning and at the end of the acceleration phase. If it is set to -1, the value stored in the drive will take effect. It only works when <i>obj. 0x6086 Motion profile type = 3</i> .
iRampFallTime_ms	INT	-1	Axis Mgr will copy this value into <i>P.1176 Ramp Fall Time</i> of the drive (RAM). The value will be used to limit the jerk of the velocity movement, applying this time [ms] at the beginning and at the end of the deceleration phase. If it is set to -1, the value stored in the drive will take effect. It only works when <i>obj. 0x6086 Motion profile type = 3</i> .
rVelocityWindow	REAL		It defines the symmetrical velocity window [pos. units/s] around the target velocity where the axis speed should operate in order to set <i>q_xInVel</i> .
uiVelocityWindowTime_ms	UINT		It defines the time delay [ms] within which the axis should operate in <i>rVelocityWindow</i> before to raise <i>q_xInVel</i> .



## 8.2.18 AM\_ProfileVelocityDataType\_Axia

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Profile Velocity function (AxisMgr AXV <fieldbus>, AxisMgr AXM <fieldbus>).

### Structure Elements

Name	Data Type	Description
rTargetVelocity	REAL	Target velocity [units/s] used during velocity. Changing of this value during an ongoing positioning will take effect.
rAcceleration	REAL	Acceleration [units/s <sup>2</sup> ] used during velocity. Changing of this value during an ongoing positioning will take effect.
rDeceleration	REAL	Deceleration [units/s <sup>2</sup> ] used during velocity. Changing of this value during an ongoing positioning will take effect.
rVelocityWindow	REAL	It defines the symmetrical velocity window [pos. units/s] around the target velocity where the axis speed should operate in order to set <i>q_xInVel</i> .
uiVelocityWindowTime_ms	UINT	It defines the time delay [ms] within which the axis should operate in <i>rVelocityWindow</i> before to raise <i>q_xInVel</i>

## 8.2.19 AM\_QuickStopDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Quick Stop function.

### Structure Elements

Name	Data Type	Description
rQuickStopDeceleration	REAL	Deceleration used when in <i>EmergencyStop</i> (i.e.: <i>i_xQuickStop</i> is triggered or <i>ErrorStop</i> state occurred). Depending on the AxisMgr used and to the actual Mode of Operation, the dimension of this value will change: <ul style="list-style-type: none"><li>→ AxisMgr_SM: units/s<sup>2</sup>. If it is set to 0, the last value used for a Move command will take effect.</li><li>→ AxisMgr "Light", Mode Of Operation ≠ 2: pos. units/s<sup>2</sup> (<i>obj. 0x6085</i> is used). If it is set to 0, the value stored in the drive will take effect. Value of <i>rQuickStopDeceleration</i> is only relevant when <i>obj. 0x605A</i> = 2.</li><li>→ AxisMgr "Light", Mode Of Operation = 2: vel. units/s (<i>obj. 0x604A</i> is used). If it is set to 0, the value stored in the drive will take effect. Value of <i>rQuickStopDeceleration</i> is only relevant when <i>obj. 0x605A</i> = 2.</li></ul>



## 8.2.20 AM\_SetPosDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the SetPos function.

### Structure Elements

Name	Data Type	Description
lrPosition	LREAL	New position [units] that the axis will assume right after a <i>Set Pos</i> command.
eSetPosMode	<a href="#">AM_PosTypeEnum</a>	Selection of the reference type for the <i>Set Pos</i> command. If Relative then <i>lrPosition</i> represents a distance from the actual position.

## 8.2.21 AM\_SetTorqueDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Set Torque function ([AxisMgr SM](#)).

### Structure Elements

Name	Data Type	Description
rTargetTorque_Nm	REAL	Torque in Nm (for rotary motors) or N (for linear motors)



## 8.2.22 AM\_StopDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Stop function.

### Structure Elements

Name	Data Type	Description
IrDeceleration	LREAL	<p>Deceleration used to stop the axis. Depending on the AxisMgr used and to the actual Mode of Operation, the dimension of this value and the behavior will change:</p> <ul style="list-style-type: none"><li>→ AxisMgr_SM: units/s<sup>2</sup>. If it set to 0, the last value used for a Move command will take effect.</li><li>→ AxisMgr "Light", Mode Of Operation = 2 or = 4: vel. units/s (<i>obj. 0x6049</i> is used). If it is set to 0, the value stored in the drive will take effect. Only relevant if <i>obj. 0x605C</i> = 1.</li><li>→ AxisMgr "Light", Mode Of Operation = 3: pos. units/s<sup>2</sup> (<i>obj. 0x6084</i> is used). If it is set to 0, the value stored in the drive will take effect.</li><li>→ AxisMgr "Light", Mode Of Operation = 1: pos. units/s<sup>2</sup> (<i>obj. 0x6084</i> is used). If it is set to 0, the value stored in the drive will take effect. Only relevant if <i>obj. 0x605D</i> = 1.</li><li>→ AxisMgr "Light", Mode Of Operation = 6: not used. <i>i_stData.stHoming.IrAcceleration</i> will be used instead.</li></ul>



## 8.2.23 AM\_SuperImposedDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the SuperImposed function.

### Structure Elements

Name	Data Type	Description
IrDistance	LREAL	Requested distance to be recovered [user unit].
IrMaxVelocity	LREAL	Maximum velocity during the superimposed profile [u/s].
IrMaxAcceleration	LREAL	Maximum acceleration during the superimposed profile [u/s <sup>2</sup> ].
IrMaxDeceleration	LREAL	Maximum deceleration during the superimposed profile [u/s <sup>2</sup> ] .
IrLength	LREAL	Refer to Tc2_MC2 documentation
IrVelocityProcess	LREAL	
eSuperImposedMode	Tc2_MC2.E_SuperpositionMode	



## 8.2.24 AM\_TouchProbeDataType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Touch Probe function.

### Structure Elements

Name	Data Type	Default	Description
stTP1	<a href="#">AM_TPChannelType</a>		Touch Probe data for channel 1
stTP2	<a href="#">AM_TPChannelType</a>		Touch Probe data for channel 2
tRefreshTouchProbeStatusTimer	TIME	T#50ms	Refresh update timer. AxisMgr refreshes the Touch probe status information every <i>tRefreshTouchProbeStatusTimer</i> time. Reducing this time will lead to a faster polling of the Touch probe information
IrScaleFactor	LREAL	1	Scale factor that establish the relationship between user unit and drive's increments. It is used for the TouchProbe position calculation. It should correspond to what was defined in the NC encoder parameters (see <a href="#">5.3.1.2</a> ). It is only relevant for AxisMgr_SM. AxisMgr "Light" will use the <i>i_rFactor_pos</i> input.

## 8.2.25 AM\_TouchProbeStatusType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the actual Touch Probe channel status information.

### Structure Elements

Name	Data Type	Description
stTP1	<a href="#">AM_TPChannelStatusType</a>	It contains the Touch probe status information of the first channel
stTP2	<a href="#">AM_TPChannelStatusType</a>	It contains the Touch probe status information of the second channel



## 8.2.26 AM\_TPChannelStatusType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains each TP channel information data.

### Structure Elements

Name	Data Type	Description
xPosEdgeValueValid	BOOL	<p><i>xPosEdgeValueValid</i> should be evaluated on the edges:</p> <ul style="list-style-type: none"><li>➔ Negative edge: a positive edge on the TP channel has been detected. From this moment, AxisMgr will read out the Pos edge position and the Pos edge counter registers stored in the drive. These data are to be considered invalid (not up to date) and should not be evaluated yet.</li><li>➔ Positive edge: the register has been read successfully, <i>IrPosEdgePositionValue</i> and <i>uiPosEdgeCounter</i> can be now considered as valid (up to date, related to the last Pos edge of the TP channel).</li></ul> <p>The time delay between the physical TP channel event and the negative edge of <i>xPosEdgeValueValid</i> is affected at maximum by <i>i_stData.tRefreshTouchProbeStatusTimer</i> + time for the SDO response + task cycle</p>
xNegEdgeValueValid	BOOL	<p><i>xNegEdgeValueValid</i> should be evaluated on the edges:</p> <ul style="list-style-type: none"><li>➔ Negative edge: a negative edge on the TP channel has been detected. From this moment, AxisMgr will read out the Neg edge position and the Neg edge counter registers stored in the drive. These data are to be considered invalid (not up to date) and should not be evaluated yet.</li><li>➔ Positive edge: the register has been read successfully, <i>IrNegEdgePositionValue</i> and <i>uiNegEdgeCounter</i> can be now considered as valid (up to date, related to the last Neg edge of the TP channel).</li></ul> <p>The time delay between the physical TP channel event and the indication of <i>xNegEdgeValueValid</i> is affected at maximum by <i>i_stData.tRefreshTouchProbeStatusTimer</i> + time for the SDO response + task cycle</p>
IrPosEdgePositionValue	LREAL	It returns the position value [units] stored in the drive when the positive edge of the TP channel was detected by the drive. It should be evaluated at the rising edge of <i>xPosEdgeValueValid</i> .
IrNegEdgePositionValue	LREAL	It returns the position value [units] stored in the drive when the negative edge of the TP channel was detected by the drive. It should be evaluated at the rising edge of <i>xNegEdgeValueValid</i> .
uiPosEdgeCounter	UINT	It returns the number of the positive edges of the TP channel detected by the inverter (since the last power-on). It should be evaluated at the rising edge of <i>xPosEdgeValueValid</i> .
uiNegEdgeCounter	UINT	It returns the number of the positive edges of the TP channel detected by the inverter (since the last power-on). It should be evaluated at the rising edge of <i>xNegEdgeValueValid</i> .



## 8.2.27 AM\_TPChannelType

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the actual TP channel specific configuration data.

### Structure Elements

Name	Data Type	Default	Description
xTouchProbeEnable	BOOL		TRUE: Enables the touch probe channel. Touch probe status for this channel is updated.
eTouchProbeMode	<a href="#">AM_TouchProbeModeEnum</a>	Continuos	Selection for Touch Probe mode.
xTP_input	BOOL		Used only for virtual/simulation. When using real axis, the TP input must be connected on the drive side (in the corresponding TP channel terminal).



## 8.2.28 AM\_VelocityModeDataType\_ACTIVE

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Velocity Mode function (AxisMgr ACUx10 <fieldbus>, AxisMgr ANG <fieldbus>).

### Structure Elements

Name	Data Type	Default	Description
rTargetVelocity	REAL		Target velocity [vel. units] used during velocity. Changing of this value during an ongoing positioning will take effect. In addition to this value, the inverter reference speed may also be affected by further internal parametrization (e.g.: <i>P.434 Ramp Setpoint</i> ).
rAcceleration	REAL		Acceleration [vel. units/s] used during velocity. Changing of this value during an ongoing positioning will take effect.
rDeceleration	REAL		Deceleration [vel. units/s] used during velocity. Changing of this value during an ongoing positioning will take effect.
iRampRiseTime_CW_ms	INT	-1	Axis Mgr will copy this value into <i>P.430 Ramp Rise Time Clockwise</i> of the drive (RAM). The value will be used to limit the jerk of the velocity movement, applying this time [ms] at the beginning and at the end of the clockwise acceleration phase. If it is set to -1, the value stored in the drive will take effect.
iRampFallTime_CW_ms	INT	-1	Axis Mgr will copy this value into <i>P.431 Ramp Fall Time Clockwise</i> of the drive (RAM). The value will be used to limit the jerk of the velocity movement, applying this time [ms] at the beginning and at the end of the clockwise deceleration phase. If it is set to -1, the value stored in the drive will take effect.
iRampRiseTime_CCW_ms	INT	-1	Axis Mgr will copy this value into <i>P.432 Ramp Rise Time Anticlockwise</i> of the drive (RAM). The value will be used to limit the jerk of the velocity movement, applying this time [ms] at the beginning and at the end of the counter-clockwise acceleration phase. If it is set to -1, the value stored in the drive will take effect.
iRampFallTime_CCW_ms	INT	-1	Axis Mgr will copy this value into <i>P.433 Ramp Fall Time Anticlockwise</i> of the drive (RAM). The value will be used to limit the jerk of the velocity movement, applying this time [ms] at the beginning and at the end of the counter-clockwise deceleration phase. If it is set to -1, the value stored in the drive will take effect.
rVelocityWindow	REAL		It defines the symmetrical velocity window [vel. units/s] around the target velocity where the axis speed should operate in order to set <i>q_xInVel</i> .
uiVelocityWindowTime_ms	UINT		It defines the time delay [ms] within which the axis should operate in <i>rVelocityWindow</i> before to raise <i>q_xInVel</i> .



## 8.2.29 AM\_VelocityModeDataType\_AGL

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant parameter for the Velocity Mode function (AxisMgr AGL <fieldbus>).

### Structure Elements

Name	Data Type	Default	Description
rTargetVelocity	REAL		Target velocity [vel. units] used during velocity. Changing of this value during an ongoing positioning will take effect. In addition to this value, the inverter reference speed may also be affected by further internal parametrization (e.g.: <i>P.475 Reference Frequency Source 1</i> ).
rAcceleration	REAL		Acceleration [vel. units/s] used during velocity. Changing of this value during an ongoing positioning will take effect.
rDeceleration	REAL		Deceleration [vel. units/s] used during velocity. Changing of this value during an ongoing positioning will take effect.
iRampRiseTime_ms	INT	-1	Axis Mgr will copy this value into <i>P.430 Ramp Rise Time</i> of the drive (RAM). The value will be used to limit the jerk of the velocity movement, applying this time [ms] at the beginning and at the end of the CW/CCW acceleration/deceleration phase. If it is set to -1, the value stored in the drive will take effect.
rVelocityWindow	REAL		It defines the symmetrical velocity window [vel. units/s] around the target velocity where the axis speed should operate in order to set <i>q_xInVel</i> .
uiVelocityWindowTime_ms	UINT		It defines the time delay [ms] within which the axis should operate in <i>rVelocityWindow</i> before to raise <i>q_xInVel</i> .



### 8.2.30 AM\_VelocityModeDataType\_Axia

#### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

#### Description

This structure contains the relevant parameter for the Velocity Mode function ([AxisMgr AXV <fieldbus>](#), [AxisMgr AXM <fieldbus>](#)).

#### Structure Elements

Name	Data Type	Description
rTargetVelocity	REAL	Target velocity [vel. units] used during velocity. Changing of this value during an ongoing positioning will take effect.
rAcceleration	REAL	Acceleration [vel. units/s] used during velocity. Changing of this value during an ongoing positioning will take effect.
rDeceleration	REAL	Deceleration [vel. units/s] used during velocity. Changing of this value during an ongoing positioning will take effect.
rVelocityWindow	REAL	It defines the symmetrical velocity window [vel. units/s] around the target velocity where the axis speed should operate in order to set <i>q_xInVel</i> .
uiVelocityWindowTime_ms	UINT	It defines the time delay [ms] within which the axis should operate in <i>rVelocityWindow</i> before to raise <i>q_xInVel</i>

### 8.2.31 AM\_VeloDataType\_SM

#### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

#### Description

This structure contains the relevant parameter for the Velocity function. This command data structure is specific for [AxisMgr SM](#).

#### Structure Elements

Name	Data Type	Description
lrTargetVelocity	LREAL	Target velocity in units/second. Negative value will reverse the movement direction. Changing the value will automatically take effect on the movement.
lrAcceleration	LREAL	Acceleration used when increasing speed in units/second <sup>2</sup> . Negative values are not allowed. Changing the value will automatically take effect on the movement.
lrDeceleration	LREAL	Deceleration used when decreasing speed in units/second <sup>2</sup> . Negative values are not allowed. Changing the value will automatically take effect on the movement.



## 8.2.32 AM\_WarningDiagnosticType\_ACTIVE

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant information about the active warning of ACU, ANG and AGL drive series. The structure is automatically cleared once the warning is not pending anylonger (acknowledge is not required).

### Structure Elements

Name	Data Type	Description
uiWarningID	UINT	Warning ID got from the drive.
strWarningMessage	STRING	Message related to the Warning ID
uiAppWarningID	UINT	Application Warning got from the drive.
strAppWarningMessage	STRING	Message related to the Application Warning ID.



## 8.2.33 AM\_WarningDiagnosticType\_Axia

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the relevant information about the active warning of Axia drive series. The structure is automatically cleared once the warning is not pending anylonger (acknowledge is not required).

### Structure Elements

Name	Data Type	Description
usiWarningGroups	USINT	Identification of the groups in which at least one active warning is present. More than one group may be active at the same time. The information is bit-coded and it is as follows: <ul style="list-style-type: none"><li>→ Bit 0: Power warnings active</li><li>→ Bit 1: Communication warnings active</li><li>→ Bit 2: Application warnings active</li><li>→ Bit 3: Encoder warnings active.</li></ul>
strWarningMessageGroups	STRING	Message related to the active Warning group.
stPowerWarningsGroup	<a href="#">AM_WarningGroupType_Axia</a>	It contains the information about the Power warnings, relevant when the first bit of <i>usiWarningGroups</i> is set.
stCommunicationWarningsGroup	<a href="#">AM_WarningGroupType_Axia</a>	It contains the information about the Communication warnings, relevant when the second bit of <i>usiWarningGroups</i> is set.
stApplicationWarningsGroup	<a href="#">AM_WarningGroupType_Axia</a>	It contains the information about the Application warnings, relevant when the third bit of <i>usiWarningGroups</i> is set.
stEncoderWarningsGroup	<a href="#">AM_WarningGroupType_Axia</a>	It contains the information about the Encoder warnings, relevant when the fourth bit of <i>usiWarningGroups</i> is set.



## 8.2.34 AM\_WarningGroupType\_Axia

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

This structure contains the warning information of each warning group. More than one warning per group can be active at the same time.

### Structure Elements

Name	Data Type	Description
udiNumber	UDINT	ID number of the actual warning got by the drive.
strMessage	STRING	Message related to the actual warning ID.



## 8.2.35 ST\_ParamType\_Common

### Overview

Type:	Structure
Available as of:	v. 3.1.2.0

### Description

The structure is used for **acyclic data** requests. The structure has to be used in combination with FB GenericAccess <fieldbus>. The FB uses this data structure to manage the request from the user program, then, it provides the result directly in it. In order to use it properly, the user should only provide the data information (*wIndex*, *bSubIndex* ...) and control the requests (*xStartRead*, *xStartWrite*). The output results (*xParamReady*, *xError*...) are managed by FB GenericAccess <fieldbus> and the user is supposed to evaluate them (overwrite of this signal may lead to unexpected behavior).

When managing several acyclic data objects, the user can declare as many ST\_ParamType\_Common as the number of object to be accessed (both reading and writing), providing the object definition (address, data type, ecc...). Once added to the FB GenericAccess <fieldbus> (by means of *ADD\_PARAM()*), the FB will carry out the requests once at time, in a cyclic manner. The user can set *xStartRead* or *xStartWrite* and threat ST\_ParamType\_Common as it was a cyclic data.

Instead, if the requests occurs only in some occasions, the user can eventually define only one ST\_ParamType\_Common with a dynamic configuration of address, data type, ecc..., requesting the information and evaluating the result.

Both the approaches are valid and can coexist per each FB GenericAccess <fieldbus>.

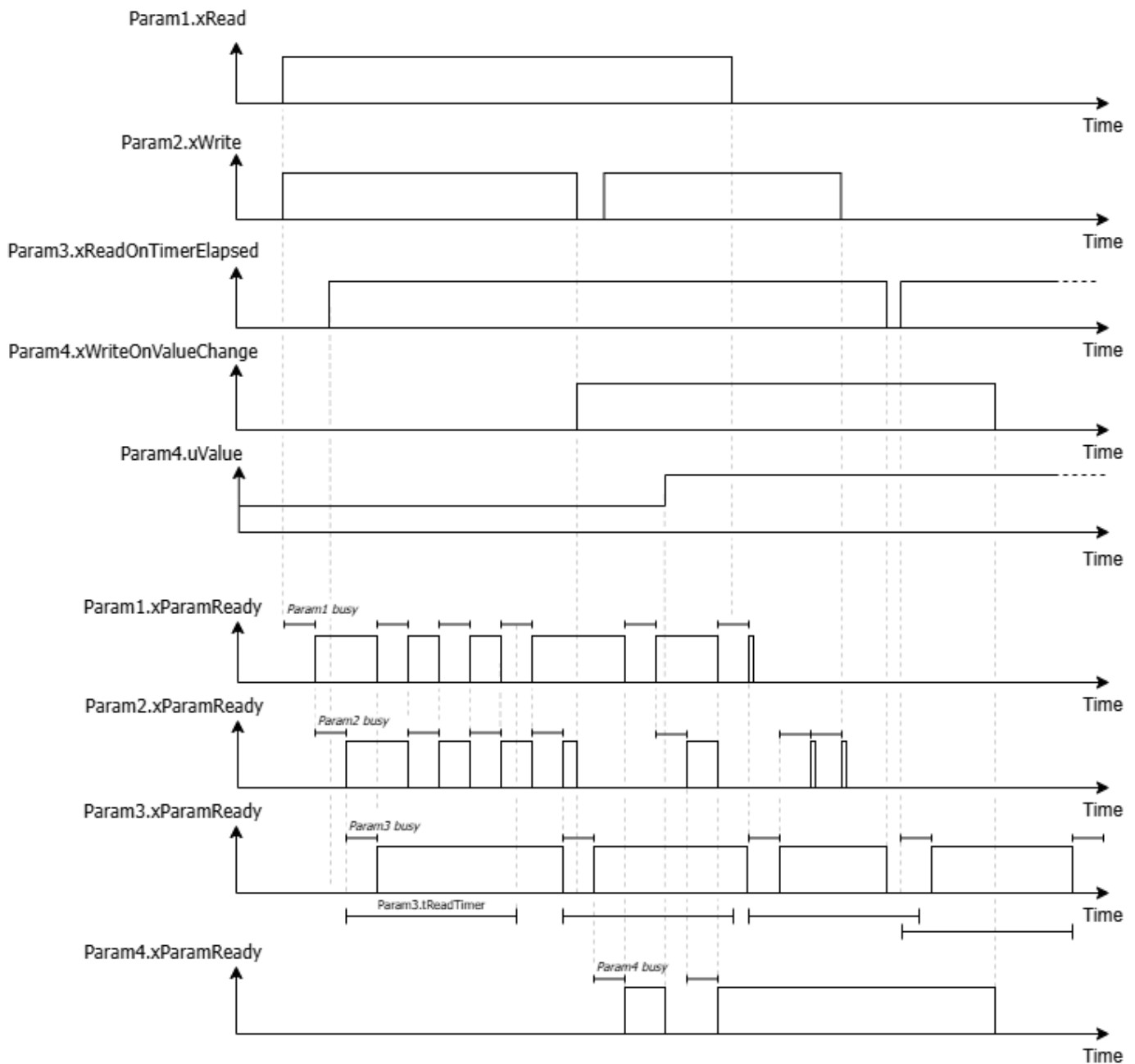
When using Axis Mgr, the input *i\_stGenericAccess* is probably provided, as well as the corresponding FB GenericAccess <fieldbus> interface. The user should use it when accessing objects of the Axis Mgr's device, otherwise, a simultaneity of requests may occur.

### Structure Elements

Name	Data Type	Description
uValue	RW_Data (4.6)	It contains the value to write/read. The union is defined in all the supported data types. In this way <i>uValue</i> will cover all the basic data types. Depending on the object to be read/written, the corresponding <i>uValue</i> 's member should be used.
wIndex	WORD	Object index. Depending on the drive and on the fieldbus used, the definition of this value may vary.
bSubIndex	BYTE	Object subindex. Depending on the drive and on the fieldbus used, the definition of this value may vary.
eDataType	<a href="#">AM_DataTypeEnum</a>	Data type (size lenght of the object)
xWrite	BOOL	Set to TRUE to write the value into the defined object. Transfer requests will be performed as long as <i>xWrite</i> is set. Read request cannot be set at the same time, no requests will be performed in this case.
xWriteOnValueChange	BOOL	Set to TRUE if the object must be transferred only if the value has changed since last time it was successfully transferred. <i>xWrite</i> has the priority over this command. A transfer request will be performed, regardless the value, once <i>xWriteOnValueChange</i> is set. Read request cannot be set at the same time, no requests will be performed in this case.
xRead	BOOL	Set to TRUE to read the value from the defined object. Receive requests will be performed as long as <i>xRead</i> is set. Write request cannot be set at the same time, no requests will be performed in this case.
xReadOnTimeElapsed	BOOL	Set to TRUE if the read requests should be delayed at predefined time interval. The time interval is defined in <i>tReadTimer</i> . <i>xRead</i> has the priority over this command. A read request will be performed once <i>xReadOnTimeElapsed</i> becomes TRUE, without waiting any time. Write request cannot be set at the same time, no requests will be performed in this case.



tReadTimer	TIME	Time interval for reading requests. Only relevant with <i>xReadOnTimeElapsed</i> = <i>TRUE</i> . Timer restarts each time the parameter access has started.
xParamReady	BOOL	TRUE (for at least one task cycle) as soon as the object has been successfully transferred/received (for at least one task cycle). This is reset if <i>xRead</i> and <i>xWrite</i> are reset. Do not overwrite this member, it should be only evaluated.
xError	BOOL	TRUE if an error occurred. Further information in <i>udiError</i> and <i>udiErrorInfo</i> . Do not overwrite this member, it should be only evaluated.
udiError	UDINT	The value identification depends on the fieldbus used. Refer to the FB GenericAccess <fieldbus> for further information. Do not overwrite this member, it should be only evaluated.
udiErrorInfo	UDINT	The value identification depends on the fieldbus used. Refer to the FB GenericAccess <fieldbus> for further information. Do not overwrite this member, it should be only evaluated.





---

## 9 Global Variables

This part contains the following chapters:

Chapter	Chapter Name	Page
<a href="#">9.1</a>	<a href="#">GVL</a>	<a href="#">140</a>



## 9.1 GVL

### Overview

Type:	Global Variable List
Available as of:	v. 3.1.2.0

### Description

This Global Variable List provides elements that are intended for the user.

### List Elements

Name	Data Type	Value	Description
gbl_fbAxisGroup	<a href="#">AxisMgr_Group</a>		This element represents a global instance of a AxisMgr_Group. All the AxisMgr declared in the project are added to this instance (both SM and "Light"). In this way the user has got an already-configured instance to control all the AxisMgr. This comes very useful when the user has to launch commands and have a single point diagnostic of all the axis together.



---

## 10 Parameters

This part contains the following chapters:

Chapter	Chapter Name	Page
<a href="#">10.1</a>	<a href="#">_AM_K</a>	<a href="#">142</a>



## 10.1 \_AM\_K

### Overview

Type:	Parameters
Available as of:	v. 3.1.2.0

### Description

This Parameters list contains the array dimension used in the library's FBs and Data Types. Those dimension are constant value and cannot be changed at runtime (during the execution of the program cycle). Nevertheless, the user can modify the value (by means of the Library Manager→Library Parameters... dialog box) before to build and download the program. The initial value is intentionally low to keep a lower memory usage. If needed, the value can be increased.

### List Elements

Name	Data Type	Value	Description
k_iParameters	INT	5	Maximum dimension of the array <i>astPar</i> used in <a href="#">AM_InitDataType</a> . It defines the maximum number of parameters that can be written in the <i>Init</i> procedure.
k_iDiagnostic	INT	4	Maximum dimension of the error diagnostic array of structure <i>q_astErrorDiagnostic</i> . It defines the maximum number of pending error that can be shown at the same time per axis.
k_iMaxNumGroupsPerAxis	INT	5	Maximum number of groups that each axis can be part of (considering the already-defined <i>gbl_fbAxisGroup</i> ).





12 REVISION INDEX (R)

BR_IOM_AXMG-CAT_STD_ENG_R01_0			
Revision	Change	Date	Author
1.0	First draft	24/09/2025	A. Campostrini

This publication supersedes and replaces any previous edition and revision. We reserve the right to implement modifications without notice.  
This catalogue cannot be reproduced, even partially, without prior consent.

*To the extent permitted by the applicable law, the Seller shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for damages to the integrated system, business interruption, reduced usability, availability and completeness, loss of profits, loss of stored information or other economic losses arising from the use of the Software, even if the Buyer has been advised of the possibility of such damages.*

*In any case, and always to the extent permitted by law, the Seller's liability to the Buyer shall be limited to an amount corresponding to the price paid for the purchase of the Software, which the parties predetermine as a penalty in accordance with article 1382 of Italian Civil Code.*

*By using the application examples the Buyer acknowledges that the Seller cannot be held liable for any damage beyond the liability provisions described above.*



We have a relentless commitment to excellence, innovation & sustainability. Our team creates, distributes and services world-class power transmission & drive solutions to keep the world in motion.

#### **HEADQUARTERS**

##### **Bonfiglioli S.p.A**

Via Cav. Clementino Bonfiglioli, 1  
40012 Calderara di Reno - Bologna (Italy)  
Tel. +39 051 6473111

