

Premium and Atrium using Unity Pro

Motion control for SERCOS® motion
User manual

07/2012

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information that is contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2012 Schneider Electric. All rights reserved.

Table of Contents



	Safety Information	11
	About the Book	13
Part I	The SERCOS® package in Premium PLCs	15
Chapter 1	SERCOS® Architecture	17
	Introduction	18
	Introduction to SERCOS® Architecture	20
	Development of the TSX CSY •4 Modules	22
Chapter 2	Implementation methodology	25
	Introduction	26
	Implementing an Independent Axis	28
	Implementation of a co-ordinated or slave axis group	30
	Installation Phase Overview	31
Chapter 3	Quick Start	33
	Description of the example	34
	Trajectory description	36
	Description of commands in manual mode	37
	Requirements and methodology	38
	Variable Speed Controller Configuration	39
	TSX CSY 84 Module Configuration	40
	Creation of IODDTs and Project Variables	41
	Symbolization of discrete input/output modules	42
	Description of the IODDTs of module TSX CSY 84	43
	Programming of preliminary processing	46
	SFC programming	47
	Transition programming	48
	Actions programming	50
	Programming of subsequent processing	53
	Program transfer	57
	Adjusting the Parameters	58
	Introduction to the Debug Mode	59

Part II	Hardware installation for the modules	
	TSX CSY 84/85/164	61
Chapter 4	Presentation of the TSX CSY 84 / 164 modules	63
	Introduction to the modules in their environment	64
	Physical presentation of the modules	66
Chapter 5	Installing the modules	69
	Mounting modules in a PLC station rack	70
	Installing the modules in a PLC station	71
	Number of application specific channels managed by a TSX CSY 84 or CSY 164 Station	72
	Installation precautions	73
	Module displays	74
	Initialization of the module following an internal error	78
	TSX CSY 84 and TSX CSY 164 module operating mode	79
Chapter 6	Description of the multi-axis control system	81
	Multi-axis control system for SERCOS® Premium PLC	82
	SERCOS® ring network	84
Chapter 7	Fiber Optic Cables	87
	Pre-Assembled Fiber Optic Cables	88
	Kits for creating custom fiber optic cables	89
Chapter 8	Characteristics, Standards and service conditions	91
	Characteristics of the module	92
	Characteristics of the SERCOS® network	93
	Standards and Operating Conditions	94
Chapter 9	Compatible servodrives	95
	List of variable speed controllers	95
Part III	Motion control common to TSX CSY 84 and TSX CSY 164 modules	97
Chapter 10	Functional Description	99
	Module Functions	100
	"SERCOS®" Function	102
	"Real Axis" Function	103
	"Imaginary Axis" Function	104
	"Remote Axis" Function	105
	Concept of the Group	107
	"Coordinated Axis Group" Function	108
	"Slave Axis Group" Function	110
	"Cam Profile" Function	113
Chapter 11	Configuration	115
	IODDT Association	116
	Module Configuration	117
	Configuration of the Channels	121
	Configuration of the SERCOS® Function (Channel 0)	122

	Configuring an Individual Axis (Channels 1 to 12)	123
	Configuration of an External Measurement Axis (Channels 13 to 16) ...	127
	Configuration of a Coordinated Set (Channels 17 to 20)	129
	Configuration of a Slave Axis Group (Channels 21 to 24).....	130
	Configuration of a Cam Profile (Channels 25 to 31)	133
Chapter 12	Programming the TSX CSY 84 and TSX CSY 164	
	modules	137
12.1	Motion Status Bits	138
	RAMPING Bit (%Ir.m.c.0)	140
	STEADY Bit (%Ir.m.c.1)	141
	STOPPING Bit (%Ir.m.c.2)	142
	PROFILE_END Bit (%Ir.m.c.3)	143
	IN_POSITION Bit (%Ir.m.c.4)	144
	AXIS_HOMING Bit (%Ir.m.c.5)	145
	AXIS_HOMED Bit (%Ir.m.c.6).....	146
	AXIS_NOT_FOLLOWING bit (%Ir.m.c.7).....	147
	HOLDING Bit (%Ir.m.c.8)	148
	RESUMING Bit (%Ir.m.c.9).....	149
	DRIVE_ENABLED Bit (%Ir.m.c.10)	150
	DRIVE_DIAG Bit (%Ir.m.c.11)	151
	DRIVE_WARNING Bit (%Ir.m.c.12)	152
	DRIVE_FAULT Bit (%Ir.m.c.13)	153
	DRIVE_DISABLED bit (%Ir.m.c.14)	154
	AXIS_SUMMARY_FLT Bit (%Ir.m.c.15).....	155
	AXIS_COMM_OK Bit (%Ir.m.c.16).....	156
	AXIS_IS_LINKED Bit (%Ir.m.c.17)	157
	AXIS_IN_CMD Bit (%Ir.m.c.18)	158
	AXIS_AT_TARGET Bit (%Ir.m.c.20).....	159
	AXIS_POS_LIMIT Bit (%Ir.m.c.21).....	160
	AXIS_NEG_LIMIT Bit (%Ir.m.c.22).....	161
	AXIS_WARNING Bit (%Ir.m.c.23).....	162
	BIAS_REMAIN Bit (%Ir.m.c.24)	163
	AXIS_MANUAL_MODE BIT (%Ir.m.c.25).....	164
	DRIVE_REALTIME_BIT1 Bit (%Ir.m.c.26)	165
	DRIVE_REALTIME_BIT2 Bit (%Ir.m.c.27)	166
	AXIS_HOLD Bit (%Ir.m.c.28)	167
	AXIS_HALT Bit (%Ir.m.c.29)	168
	AXIS_FASTSTOP Bit (%Ir.m.c.30).....	169
	AXIS_READY Bit (%Ir.m.c.31)	170
	CONF_OK Bit (%Ir.m.c.32).....	171
12.2	Movement Control Functions	172
	Motion Control Functions: General	173
	Management of ALLOW and CONTROL Bits	174
	MovementControl Bits Modifiable on Edge	176
	Control Bits Modifiable on Change of State	177

	Status Bits and Words	178
	Manual Mode	179
	Set_Functional_Mode Function	181
	Home Function.	182
	Reference Point with the Home Function	184
	ForcedHomed Function	185
	Unhome Function.	186
	SetPosition Function	187
12.3	Movement Functions	188
	About Movement Functions	189
	MoveType Data Type.	191
	MoveImmed Function	192
	MoveQueue Function.	194
	GetMoveQueueLength Function	196
	GetSpeedOverride and SetSpeedOverride Function.	197
	EnableRealTimeCtrlBit Function	198
12.4	Motion Function Following Deactivation of the Sequence	199
	Overview of the Function	200
	"MoveImmed" Function	201
	"MoveImmed" Function with ABS_MOVE_NO_ROLLOVER.	203
	"MoveQueue" Function	205
	"MoveQueue" Function with ABS_MOVE_NO_ROLLOVER.	207
12.5	Functions of actual speed/position.	209
	GetActualSpeedFunction.	210
	GetCommandSpeed Function	211
	Modulo counter service	212
	GetUnrolledPosition Function	213
	GetUnrolledCommandedPosition Function	214
	GetCommandedPosition Function.	215
12.6	Following Functions	216
	About Following Functions.	217
	"Target sequence activation" function	219
	"Estimation of Remaining Time Before Activation of Sequence" Function	221
	Sequence stop (Unlink) of the slave group set off by a trigger	223
	Movement Function for coordinated group compatible with the MMS	
	Quantum function	226
	FollowOn 415 Function Description.	228
Chapter 13	Adjusting the TSX CSY 84 and TSX CSY 164 Modules	231
13.1	SERCOS® Module Adjustment: General	232
	Description of the SERCOS® Module Adjustment Screens: General	233
	Adjustment Principles	236

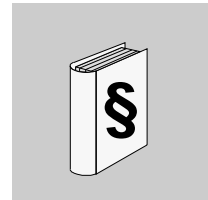
13.2	Adjustment of the Parameters of a Real, Imaginary or Remote Axis	237
	Description of the Adjustment Screen of a SERCOS® Module: Real or Imaginary Axes	238
	Description of the Adjustment Screen of a SERCOS® Module: Remote Axes	240
	Description of the TRF_RECIPE Screen for Real Axes	242
	TRF_RECIPE	244
	TRF_RECIPE Examples	246
	Speed Parameter: DefaultSpeed	247
	Speed Parameter: SpeedLimit	248
	About Acceleration/Deceleration Parameters	249
	Acceleration/Deceleration Parameter: Accel.	250
	Acceleration/Deceleration Parameter: Decel	251
	Acceleration/Deceleration Parameter: AccelMax	252
	Acceleration/Deceleration Parameter: DecelMax	253
	Acceleration/Deceleration Parameter: AccelType.	254
	Data type AccelerationType	256
	Resolution Parameters: GearRatio	257
	Position Limit and Modulo Parameter: PositionLimit.	259
	About the Modulo	261
	Position Limit and Modulo Parameter: RolloverLimit.	263
	Position Limit and Modulo Parameter: EnableRollover.	265
	About the In-Position Band	266
	Position Limit Parameter: InPositionBand.	267
	About Enable Mode.	268
	Position Limit Parameter: EnableMode.	269
	Behavior Slave Axis Disable	270
13.3	Parameters of a Group of Slave Axes.	271
	Description of the Adjustment Screen of a SERCOS® Module: Slave Axes	272
	About a Group of Slave Axes	274
	SlaveMode Data	277
	Slave Group Parameters: FollowerMode	278
	Slave Group Parameter: FollowerConfig.	279
	Slave Group Parameter: FollowerRatio.	280
	About the Master Offset	282
	Slave Group Parameter: MasterOffset	283
	Slave Group Parameter: MasterTriggerPosition	285
	Slave Group Parameter: AbsFollowerBias	287
	Slave Group Parameter: FollowerBias	290
	Propagation of stops in a slave group.	291
	Alignment Function of the Emergency Stop Inclines.	292
13.4	Cam Profile Adjustment.	294
	About Cam Profiles	295
	Cam Profile Parameter: Length.	296
	Cam Profile Parameter: LookUpFollowerPosition Function	297

	Cam Profile Parameter: Coord.	298
	Cam Profile Parameter: InterpType	300
	Cam Profile Parameter: ProfileId	301
	Description of the TRF_RECIPE Screen for Cam Profiles	302
	TRF_RECIPE Instruction.	304
13.5	Sercos® ring parameters.	306
	Description of the Adjustment Screen of a SERCOS® Module:	
	SERCOS® ring	307
	Adjustment graph for optical power	310
	Sercos® ring parameter: OpticalPower	311
13.6	SERCOS® IDN Read/Write Functions	312
	Servodrive Parameters: IDN_S and IDN_P	313
	Servodrive Parameters: IDN_US and IDN_UP	315
	Standard SERCOS Identification Numbers (IDN)	317
	IDNs for customized telegrams	320
Chapter 14	Debugging the TSX CSY 84 / 164 modules	321
	Description of the debug screen: General	322
	Debug Screen User Interface	325
	Debug: Enable	326
	Debug: Diagnostics	327
	Debug: Movement	330
	Debug: Sending commands (Auto mode)	332
	Debug: Manual command (Manual mode).	333
	Debug: Reference	334
	Debug: Following	335
	Debug: Drive	336
	Debug: Position	337
	Module diagnostic.	338
	Channel Diagnostics	340
Chapter 15	Diagnostics and Maintenance	341
15.1	SERCOS® Phase Diagnostics Introduction	342
	About SERCOS® Phases Standard	343
	GetActualPhase Function	344
	GetCommandedPhase Function	345
15.2	Reading Actual Data for the Ring Introduction.	346
	GetSercosAddress Function	347
	GetNumberOfDrivesInRing function	348
	IsLoopUp function	349
15.3	Axis Identification.	350
	GetNumberInSet Function.	350

15.4	Status and fault information.	351
	Implicit Exchange Objects	352
	Faults Accessible via the GetMotionFault Command	353
	GetMotionFault Function	355
	Faults Accessible via the GetMotionWarning Command	356
	GetMotionWarning Function	358
15.5	Faults and Warnings	359
	Module faults which can be accessed by explicit reading	360
	Channel faults which can be accessed by explicit reading	361
	List of Configuration and Adjustment Errors	362
	Faulty Registers	365
	List of Error Codes for a WRITE_CMD Command	369
	WRITE_CMD Command: Programming Errors.	370
	WRITE_CMD Command: Communication errors	372
	WRITE_CMD Command: System errors.	373
	WRITE_CMD Command: System warnings	375
	TRF_RECIPE Command Explicit Writing Faults.	376
	Operating modes.	377
	Logic diagrams of faults	378
	Servodrive faults	380
15.6	Performances of the TSX CSY 84.	382
	Performances of the TSX CSY 84.	382
Chapter 16	SERCOS language objects	385
16.1	The language objects and IODDTs of the SERCOS module	386
	Introduction to IODDTs associated with TSX CSY 84 modules	387
	Implicit Exchange Language Objects Associated with the Application-Specific Function.	388
	Explicit Exchange Language Objects Associated with the Application-Specific Function.	389
	Management of Exchanges and Reports with Explicit Objects	391
	Interface Language	396
	Parameter Management	398
	WRITE_PARAM and READ_PARAM	400
	SAVE_PARAM and RESTORE_PARAM	401
	WRITE_CMD.	402
	WRITE_CMD Examples	404
	READ_STS	406
	TRF_RECIPE and MOD_PARAM.	407
16.2	Language Objects and IODDTs Associated Specifically with the SERCOS Module.	408
	Details concerning T_CSY_CMD-type IODDT implicit exchange objects.	409
	Details concerning T_CSY_CMD-type IODDT explicit exchange objects.	411
	Details concerning T_CSY_RING-type IODDT implicit exchange objects	413
	Details concerning T_CSY_RING-type IODDT explicit exchange objects	416
	Details concerning T_CSY_TRF-type IODDT explicit exchange objects	419

	Details concerning T_CSY_IND-type IODDT implicit exchange objects .	421
	Details concerning T_CSY_IND-type IODDT explicit exchange objects .	425
	Details concerning T_CSY_FOLLOW-type IODDT implicit exchange objects	429
	Details concerning T_CSY_FOLLOW-type IODDT explicit exchange objects	432
	Details concerning T_CSY_COORD-type IODDT implicit exchange objects	440
	Details concerning T_CSY_COORD-type IODDT explicit exchange objects	443
	Details concerning T_CSY_CAM-type IODDT implicit exchange objects	446
	Details concerning T_CSY_CAM-type IODDT explicit exchange objects	447
16.3	The IODDT Type T_GEN_MOD Applicable to All Modules.	450
	Details of the Language Objects of the T_GEN_MOD-Type IODDT. . . .	450
Chapter 17	Appendices.	453
17.1	List of Accessible Functions.	453
	Accessible Functions for SERCOS® Channel 0	454
	Accessible Functions for a Real Axis.	456
	Accessible Functions for an Imaginary Axis	460
	Accessible Functions for a Remote Axis	463
	Accessible Functions for a Group of Co-ordinated Axes	465
	Accessible Functions for a Group of Slave Axes.	467
	Accessible Functions for a Cam Profile	469
Part IV	Functions specific to the TSX CSY 164.	471
Chapter 18	Compatibility	473
	Compatibility	473
Chapter 19	Configuration of the TSX CSY 164 module	475
	Configuration of the TSX CSY 164 module	476
	Configuring channels.	477
Chapter 20	Functions specific to the TSX CSY 164	481
	Dynamic reconfiguration of groups	482
	Monitoring Function of the Position Deviations Between Axes	484
Chapter 21	Performances of the TSX CSY 164.	489
	Performances of the TSX CSY 164 Module	489
Index		493

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of this symbol to a Danger safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.



DANGER

DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.



WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, **can result in** death or serious injury.

CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

<i>NOTICE</i>

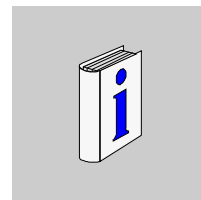
<i>NOTICE</i> is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book



At a Glance

Document Scope

This manual describes the installation of motion control using the SERCOS® TSX CSY 84/164 modules on Premium and Atrium PLCs.

Validity Note

This documentation is valid from Unity Pro v7.0.

Product Related Information

WARNING

UNINTENDED EQUIPMENT OPERATION

The application of this product requires expertise in the design and programming of control systems. Only persons with such expertise should be allowed to program, install, alter, and apply this product.

Follow all local and national safety codes and standards.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

The SERCOS® package in Premium PLCs



Subject of this Part

This part introduces the SERCOS® architecture and describes the methodology for implementing the SERCOS® motion control in Premium PLCs.

NOTE: In this section, all the data relating to TSX CSY 84 mounting is also applicable to the TSX CSY 164. Moreover, the TSX CSY 84 is taken as an example.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	SERCOS® Architecture	17
2	Implementation methodology	25
3	Quick Start	33

Subject of this Chapter

This chapter gives a brief introduction to the digital link between master and slaves, as defined by the standard EN 61491 "Serial data link for real time communication between command units and training devices".

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Introduction	18
Introduction to SERCOS® Architecture	20
Development of the TSX CSY •4 Modules	22

Introduction

Digital Link

The digital link between an axis control module (master) and intelligent servodrives (slaves) is defined by European standard EN 61491 relating to electrical equipment for industrial machines.

The use of a distributed architecture enables the project I/Os (position encoder, emergency stop, etc.) to be connected directly to the servodrives, reducing cabling costs.

The fiber optic link enables high speed exchanges (2 or 4 MHz) and ensures immunity against electromagnetic interference.

Exchanged Data

There are two types of data exchanged via the digital link:

- cyclic data exchanged from master to slaves (position control, etc.) or from slaves to master (position measurement, etc.). Cyclic data exchange between the master and each slave is limited to 8 read and 8 write objects per SERCOS® cycle.
- non-cyclic data: complex commands, parameter write or read, etc.
For each cycle, these exchanges are made via 2 reserved bytes in read mode and 2 reserved bytes in write mode. Several cycles are therefore required to exchange an object (e.g. to complete a parameter read).

Identification of Exchanged Objects

All objects may be accessed using an identification number: IDN

The standard can identify 31,768 objects and specify about 300 (e.g. IDN 40 = speed value).

All objects consist of the following fields: Name (maximum 64 characters), attribute, unit, maximum value, minimum value, value.

Operating Modes

The bus operating modes follow 5 phases as follows:

At start up

Phase	Operating mode
Phase 0	Bus ring test. The servodrives are in repeater mode.
Phase 1	Determination of slaves on the bus.
Phase 2	Servodrive system configuration.
Phase 3	Cyclic exchange programming. Servodrive parametering.

In normal operation

Phase	Operating mode
Phase 4	Active cyclic exchanges.

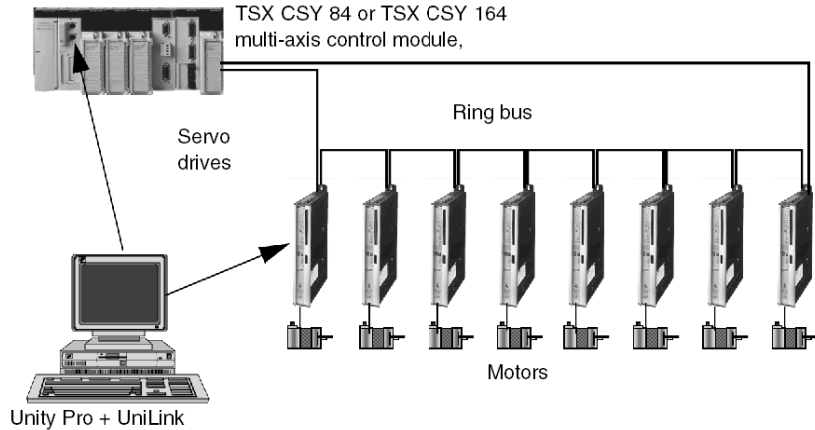
Because each servodrive acts as a repeater on the bus, the system switches to phase 0 in the event of a power supply cut, a communication error, a fault on one of the servodrives or the disconnection of the power supply to the bus.

NOTE: Certain parameters (IDN) may only be accessed in phase 3 (see standard EN 61491). The following functions GetActualPhase, GetCommandedPhase and SetCommandedPhase indicate the active phase and set it in phase 3.

Introduction to SERCOS® Architecture

Overview

The overview of the SERCOS® architecture is as follows:



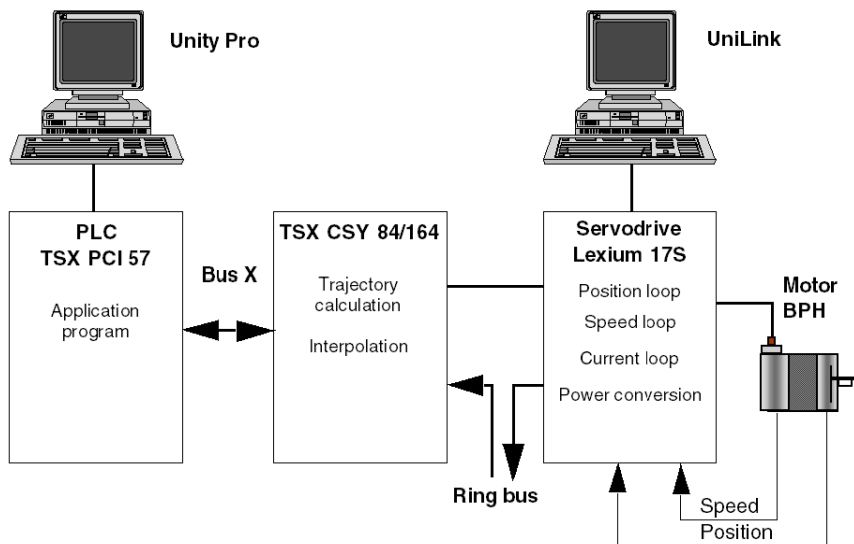
Introduction to the Premium Package

The Premium package consists of:

- the Premium/Atrium PLC range
- the axis control module TSX CSY 84 which can control up to 8 servodrives via the bus ring,
- the axis control module TSX CSY 164 which can control up to 16 servodrives via the bus ring,
- the range of Lexium 17S servodrives (4 A to 56 A): LXM 17S,
- the range of motors: BPH...,
- Unity Proware is used to configure the TSX CSY 84/164 module and program the motion project.
- UniLink software which can be used to enter parameters and adjust servodrives.

Functional Diagram

The following diagram shows the different functions performed by the motion controller and servodrives:



NOTE: It is possible to use a servodrive that is not in the Premium/Atrium range. In this case, it will be configured by its own configuration software and not by UniLink.

Functions Performed by the Various Modules

The axis control module TSX CSY 84/164 performs the trajectory calculation and interpolates several axes.

The servodrive controls the position, speed and torque loops. It also provides the power conversion to control the motor. The encoder information is sent to the servodrive (current position, current speed).

Exchanges between the PLC and the motion controller are made via the bus X at the bottom of the rack.

Exchanges between the motion controller and the servodrives are made via the SERCOS® ring bus.

Development of the TSX CSY •4 Modules

At a Glance

The version of the module is displayed:

- on the label located on the side of the module
- in the module zone of the debug screens in online mode

	CSY 84 V1.1	CSY 84 V1.2	CSY 84 V1.3	CSY 84 V1.6	CSY 164 V1.0	CSY 164 V1.3	CSY 164 V1.6
Option to close the position loop via an incremental encoder or SSI on Lexium 17S and 17HP servodrives (1) (Not documented)	X	X	X	X	X	X	X
Option to switch the command position to torque (1) and (2) (Not documented)	X	X	X	X	X	X	X
Option to switch the command position to speed command (1) and (2) (Not documented)	X	X	X	X	X	X	X
Read or write the servodrive parameters	X	X	X	X	X	X	X
Position loop with external encoder (1)	X	X	X	X	X	X	X
Torque mode (1)	X	X	X	X	X	X	X
Speed mode (1)	X	X	X	X	X	X	X
Manual mode (4)		X	X	X	X	X	X
Behavior upon fault or independent axis disable (FREEWHEEL_STOP) (3)			X	X	X	X	X
Propagation of stops in a slave group			X	X	X	X	X
Alignment function of the emergency-stop inclines			X	X	X	X	X
Selection of stop behavior following latching command			X	X	X	X	X
Stop of a member axis of a slave group in an absolute position following an Unlink			X	X		X	X
Stop of a member axis of a slave group in an absolute position			X	X		X	X
"Target sequence activation" function			X	X		X	X
Modulo counter service			X	X		X	X
Switching ACC and DEC parameters in the MOVE			X	X		X	X

	CSY 84 V1.1	CSY 84 V1.2	CSY 84 V1.3	CSY 84 V1.6	CSY 164 V1.0	CSY 164 V1.3	CSY 164 V1.6
Sequence stop (Unlink) of the slave group set off by a trigger			X	X		X	X
Dynamic configuration of individual channels				X	X	X	X
Dynamic reconfiguration of groups				X	X	X	X
Monitoring function of the deviations between axes				X	X	X	X
Absolute move above rollover limit for independent axes				X			X
Movement functions for coordinated axis compatible with MMS Quantum				X			X
Function GetPositionCounter and SetPositionCounter				X			X
New adjustment functions for real axis: EnableDriveWarning and DisableDriveWarning				X			X

(1) Requires modification of files for the TSX CSY 84/164 (Sercos.cfg) modules.

(2) Requires a minimum version for the Lexium drives (MHDS xxxxN 00 ≥ SV 1.3, MHDA xxxx ≥ SV 1.4) no restrictions for Lexium HP and Lexium option AS servodrives.

(3) Requires a minimum version for the Lexium MHDx 5.51 servodrives.

Requires the Unity V2.1 version or higher in order to access debug screens.

Implementation methodology



Subject of this Chapter

This chapter describes the global methodology for implementing movement of an independent axis or interpolated axes.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Introduction	26
Implementing an Independent Axis	28
Implementation of a co-ordinated or slave axis group	30
Installation Phase Overview	31

Introduction

TSX CSY 84 Module Features

The TSX CSY 84 module can be used to perform the following functionalities:

- 8 real axes connected to a servodrive (channels 1 – 8),
- 4 imaginary axes (channels 9 - 12),
- 4 axes with remote input (channels 13 – 16),
- 4 groups of coordinated axes (channels 17 – 20). Each group may be used for linear interpolation of between 2 and 8 axes,
- 4 groups of slave axes (channels 21 – 24). Each group may contain a maximum of 7 axes: 1 master axis and 6 slave axes.

NOTE: The TSX CSY 84 module also provides 7 cam profiles (channels 25 - 31).

TSX CSY 164 Module Features

The TSX CSY 164 module can be used to perform the following functionalities:

- 16 individual axes (channels 1 to 16),
channels 1 to 16 can each support either a real axis function, an imaginary axis function, or an external setpoint:
- 4 groups of coordinated axes (channels 17 – 20). Each group may be used for linear interpolation of between 2 and 8 axes,
- 4 groups of slave axes (channels 21 – 24). Each group may contain a maximum of 7 axes: 1 master axis and 6 slave axes.
- 7 cam profiles (channels 25 to 31).

NOTE: The TSX CSY 84/164 module also provides 7 cam profiles (channels 25 - 31).

Real axis

A real axis is a physical axis, which controls a servodrive via the SERCOS® ring bus.

Imaginary Axis

An imaginary axis is not a physical axis. It can, however, be used to coordinate the movements of several physical axes. For example, an imaginary axis may be the master axis in a slave group.

An imaginary axis can also be used in the adjustment or debugging phase to simulate a real master axis independently of the process.

Remote axis

A remote axis can be used to upload an item of remote position data to the module. Typically, the TSX CSY 84/164 module must take up a sequential position from an encoder controlled by an external mechanism and connected to the output of the auxiliary position of the servodrive.

Group of co-ordinated axes

A group of coordinated axes consists of axes, which move in coordination with one another. One of the axes in the set, defined as the coordination master, acts as a speed reference for the movement of the set.

The acceleration and speed of other coordinated axes will be calculated so that all the axes move at the same time.

Slave Axis Group

A group of slave axes consists of a master axis and slave axes, which follow the movement of the master axis. There are 2 ways of following the master axis:

- In Ratio mode: Each slave axis follows the master axis according to a relationship that has been defined in configuration, known as the Slave relation. For example, the position of the slave equates to that of the master multiplied by a ratio x ,
- In Cam mode: The slave axes follow the master axes according to a cam profile.

A cam profile can be used to implement an electronic cam in order to simplify the task of programming complex movements. A points table is used to define the position of the slave according to the master's position.

Implementing an Independent Axis

Introduction

An individual axis can be either a real axis connected to a servodrive, an imaginary axis, or a remote axis.

The groups of slave or coordinated axes consist of a collection of individual axes (real, imaginary or remote).

Implementation Methodology for a Real Axis

Before a real axis may be implemented, channel 0 must be enabled (all the channel 0 ALLOW bits: %Qr.m.0.18 and %Qr.m.0.26 at %Qr.m.0.31 are at status 1).

Implementation methodology for a real axis occurs in 3 phases:

- Phase 1: configuration of the servodrive using UniLink software,
- Phase 2: configuring the TSX CSY 84 module, using the Unity Pro configuration editor (module declaration and parameter configuration for all axes used),
- Phase 3: project program writing, transferred from this program to the PLC and project debugging.

NOTE: Enabling the servodrive via the Unilink software inhibits module commands to the servodrive. The servodrive must therefore be disabled before exiting the Unilink software.

Implementation Methodology for an Imaginary Axis

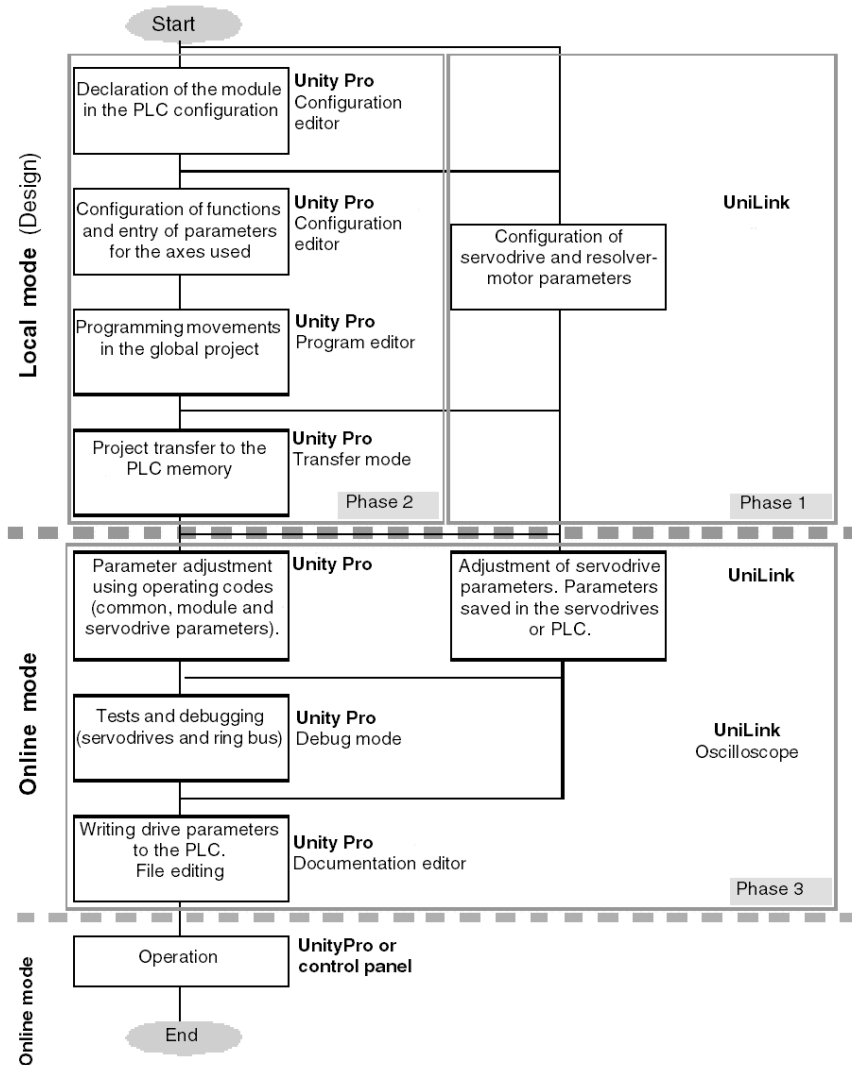
An imaginary axis is not connected to a servodrive (it is not a physical axis). Except for operations linked to a servodrive that does not exist, an imaginary axis is implemented in the same way as a real axis.

Implementation Methodology for a Remote Input

A remote input has far fewer functions than other kinds of individual axes. Its implementation is identical to the implementation of a real or imaginary axis in which only the position data needs to be configured. In this case, all operations linked to the servodrive or the programming of a movement do not exist.

Implementation Methodology for a Real Axis

Overview



Implementation of a co-ordinated or slave axis group

Group of co-ordinated axes

Co-ordinated axes are real or imaginary axes. A group of co-ordinated axes may be implemented by first implementing the real axes (methodology described on previous pages) and then the group.

Slave axis group

In a slave axis group, the master axis may be real, imaginary or external and the slave axes may be real or imaginary. Implementing a slave axis group first entails implementing the independent axes which make up this group, following the methodology described on the previous pages, and then the group.

Installation Phase Overview

Introduction

The software installation of the application-specific modules is carried out from the various Unity Pro editors:

- in offline mode
- in online mode

If you do not have a processor to connect to, Unity Pro allows you to carry out an initial test using the simulator. In this case the installation (*see page 32*) is different.

The following order of installation phases is recommended but it is possible to change the order of certain phases (for example, starting with the configuration phase).

Installation Phases with Processor

The following table shows the various phases of installation with the processor:

Phase	Description	Mode
Declaration of variables	Declaration of IODDT-type variables for the application-specific modules and variables of the project.	Offline (1)
Programming	Project programming.	Offline (1)
Configuration	Declaration of modules.	Offline
	Module channel configuration.	
	Entry of configuration parameters.	
Association	Association of IODDTs with the channels configured (variable editor).	Offline (1)
Generation	Project generation (analysis and editing of links).	Offline
Transfer	Transfer project to PLC.	Online
Adjustment/De-bugging	Project debugging from debug screens, animation tables.	Online
	Modifying the program and adjustment parameters.	
Documentation	Building documentation file and printing miscellaneous information relating to the project.	Online (1)
Operation/Diag-nostic	Displaying miscellaneous information necessary for supervisory control of the project.	Online
	Diagnostic of project and modules.	
Key:		
(1)	These various phases can also be performed in the other mode.	

Implementation Phases with Simulator

The following table shows the various phases of installation with the simulator.

Phase	Description	Mode
Declaration of variables	Declaration of IODDT-type variables for the application-specific modules and variables of the project.	Offline (1)
Programming	Project programming.	Offline (1)
Configuration	Declaration of modules.	Offline
	Module channel configuration.	
	Entry of configuration parameters.	
Association	Association of IODDTs with the modules configured (variable editor).	Offline (1)
Generation	Project generation (analysis and editing of links).	Offline
Transfer	Transfer project to simulator.	Online
Simulation	Program simulation without inputs/outputs.	Online
Adjustment/De-bugging	Project debugging from debug screens, animation tables.	Online
	Modifying the program and adjustment parameters.	
Key:		
(1)	These various phases can also be performed in the other mode.	

NOTE: The simulator is only used for the discrete or analog modules.

Quick Start

3

Aim of this Chapter

This chapter provides an example of the implementation of a motion control application using the SERCOS® package for Premium PLCs.

NOTE: The TSX CSY 84 module is used as an example. All procedures for the TSX CSY 84 configuration are identical for the TSX CSY 164 module.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description of the example	34
Trajectory description	36
Description of commands in manual mode	37
Requirements and methodology	38
Variable Speed Controller Configuration	39
TSX CSY 84 Module Configuration	40
Creation of IODDTs and Project Variables	41
Symbolization of discrete input/output modules	42
Description of the IODDTs of module TSX CSY 84	43
Programming of preliminary processing	46
SFC programming	47
Transition programming	48
Actions programming	50
Programming of subsequent processing	53
Program transfer	57
Adjusting the Parameters	58
Introduction to the Debug Mode	59

Description of the example

Introduction

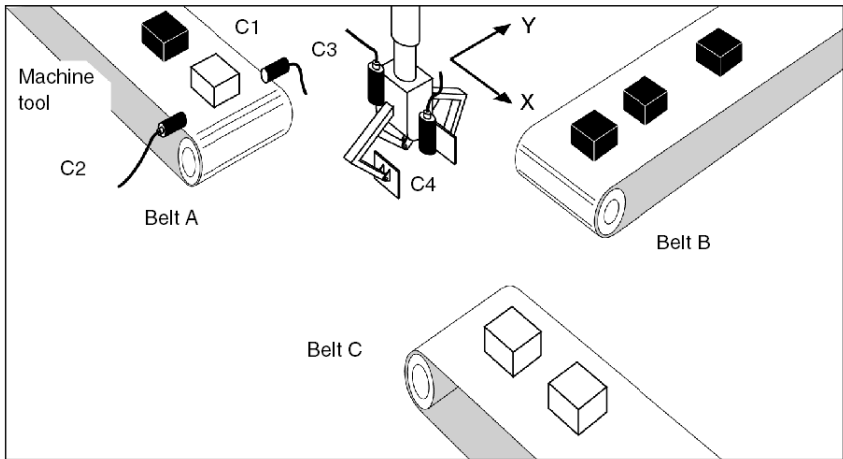
This example is given for teaching purposes only. It will enable you to follow each phase of implementation for a TSX CSY 84 axis control module, without having to read the whole manual.

Transfer Device

A transfer device ensures that machining outputs are removed. This device is made up of a claw which can move in space on an axis (X and Y axes) parallel to the ground.

When an item appears on the removal conveyor belt A, the claw automatically locates it in order to place it on belt B or C depending on the type of item. The claw then returns to position, ready to grab as soon as a new machined item is detected.

Illustration



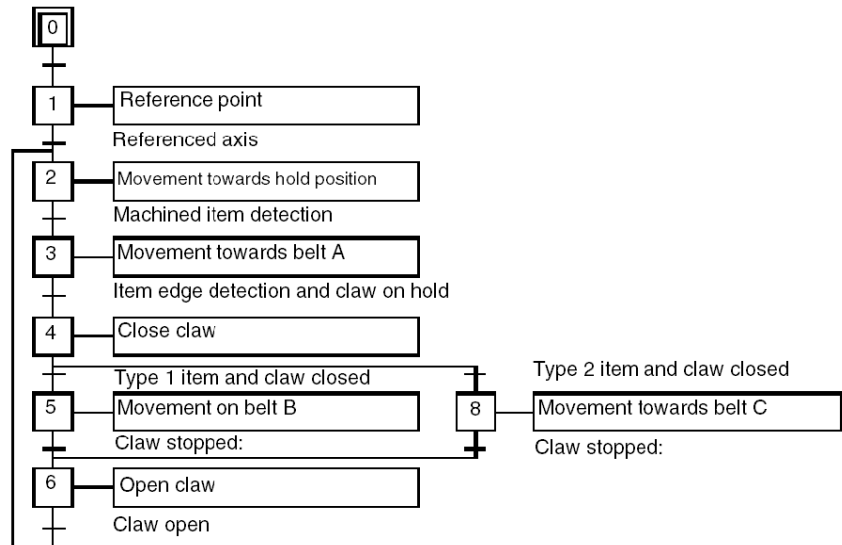
Inputs/Outputs

The inputs/outputs are as follows:

C1	machined item detection cell
C2	identification sensor for item type
C3	claw open/closed detection sensor
C4	item edge detection cell (in the claw), connected to the connector event input
O/F	claw: claw open/close command

Project Grafcet

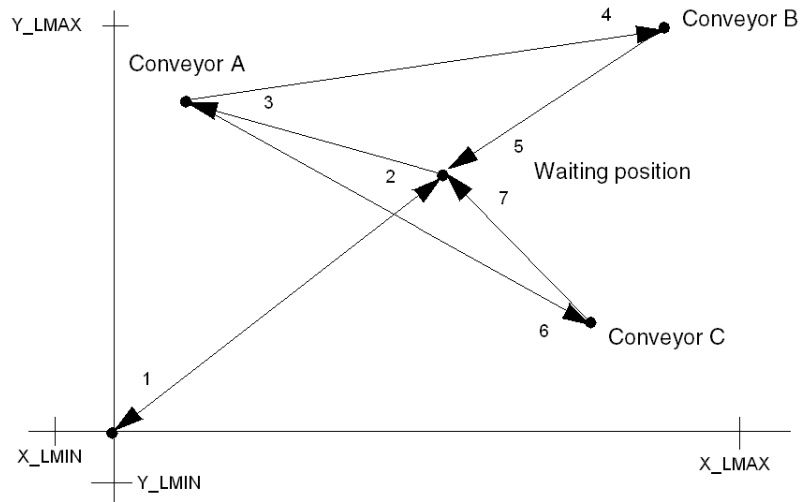
The project Grafcet is as follows:



Trajectory description

Trajectory illustration

The following diagram describes the trajectory:



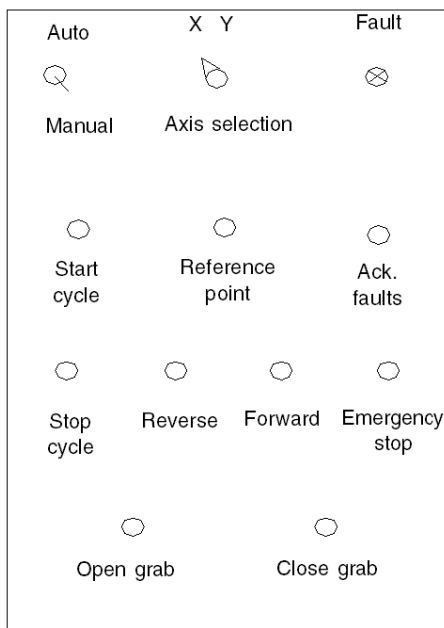
Description

1	Homing at speed V_{p0}
2	Movement at speed V_{ret} to the waiting position (X_{att} , Y_{att}) with stop
3	Movement to the A conveyor belt (X_A , Y_A) until the machined item is detected at speed V_A
4	Movement at speed V_B on belt B (X_B , Y_B) with stop
6	Movement at speed V_C on belt C (X_C , Y_C) with stop
5, 7	Movement at speed V_{ret} to the waiting position (X_{att} , Y_{att}) with stop

Description of commands in manual mode

Front Panel of the Human-Machine Interface

The following commands are regrouped on a front panel in order to control the moving part manually whilst there is an installation error. The commands and LEDs are run by discrete input and output modules



Description of the Commands

- **Auto / Man:** operating mode selector switch
- **Start cycle:** automatic execution of the cycle
- **Stop cycle:** automatic stopping of the cycle
- **X / Y axis selection:** Axis selection for control in manual mode.
- **Reference point:** manual reference point on the selected axis.
- **Forward / Reverse:** positive or negative manual motion control for the selected axis.
- **Fault:** LED grouping all hardware and project faults.
- **Ack. fault:** command to acknowledge faults present.
- **Emergency stop:** immediate stop of the moving part whatever mode is selected.
- **Open grab:** claw open command
- **Close grab:** command to close the grab

Requirements and methodology

Requirements

In order to restrict this description to the specific functions for the axis command, it should be assumed that the following operations have already been performed:

- Unity Proware is installed,
- UniLink software is installed,
- hardware is installed.

Methodology

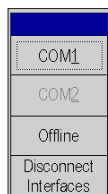
Implementation of the TSX CSY 84 module requires:

- configuration of the Lexium 17S variable speed controller, using UniLink software
- configuration of the TSX CSY 84 module, using Unity Proware.

Variable Speed Controller Configuration

Connection to the Variable Speed Controller

Access to the UniLink software displays the following screen:



Click on COM1 or COM2 to connect to the variable speed controller (this depends on the serial port used on the programming terminal for connection to the variable speed controller).

Configuration of the Motor and the SERCOS® Address

Please consult the UniLink software user manual on how to configure the motor and the SERCOS® address.

TSX CSY 84 Module Configuration

Software Declaration of the PLC Configuration

Launch the Unity Proware, select the command File / New, and choose a Premium processor.

Access the configuration editor from the project browser. To do this:

- double-click on the tab **X-Bus**,

Then select each component in the PLC configuration. The following items have been selected in this project:

- rack 0 and rack 1: TSX RKY 8E,
- processor: TSX P57 304,
- supply modules: TSX PSY 2600 for rack 0 and TSX PSY 5500 for rack 1,
- 32 inputs module: TSX DEY 32D2K in position 2 of rack 0,
- 32 outputs module: TSX DSY 32T2K in position 3 of rack 0,
- axis control module: TSX CSY 84 in position 2 of rack 1.

The processor cycle time should exceed 8ms (2 x the duration of the module cycle).

Axis Configuration

Double click on the TSX CSY 84 module (position 2 of rack 1), which displays the configuration screen.

- In the channel level zone select: Real 1, then the Real Axis function,
- Enter the parameter values,

in accordance with the following screen for a real-type axis 1:

Limits	Units	Movement
☒ Position checking	Type: Linear	☐ Modulo
Max. position: 5.000000e+005	Position: mm	Max. modulo: 0.000000e+000
Min. position: -5.000000e+005	Speed: mm/min	Min. modulo: 0.000000e+000
Max. speed: 1.200000e+004	Acceleration: mm/s²	In-position band: 1.000000e+005
Max. Acceleration: 1.200000e+004	Scale factor: Numerator: 2.000000e+000	Acceleration: 1.000000e+004
Max. Deceleration: 1.200000e+004	Denominator: 1.000000e+000	Deceleration: 1.000000e+004
☐ Enable position band		Acceleration type: Rectangle 100 %
Tolerance: 0.000000e+000		

- Enter the same parameters for the second axis (Real 2).
- Confirm the entry using the Edit / Confirm command or the icon
- On the main screen of the configuration editor, confirm the configuration using the Edit /Confirm command or the icon:

NOTE: Maximum speed (VMax) is set for a 6000 revs/min motor, with movement of 2mm per rev.

Creation of IODDTs and Project Variables

Creation and Association of IODDTs

We are going to create two IODDTs of the type `T_CSX_IND` (independent axes) that will then be associated with channels 1 and 2 of the TSX CSX 84 module. If you do not know how to carry out this operation, please refer to the corresponding section.

The two IODDTs are declared and associated as follows:

IODDT	Associated channel	Comment
Axis_1	%CH1.2.1	
Axis_2	%CH1.2.2	

Internal Variables

The following internal variables are created in the variable editor:

Symbol	Address	Comment
Speed_reached	%MF102	
Man_speed	%MF104	

Symbolization of discrete input/output modules

Discrete input module symbols

The discrete input module is positioned in slot 2 of rack 0. The symbols are as follows

Symbol	Address	Comment
Sensor_2	%I0.2.1	Item identification sensor (0= type 2, 1 = type 1)
Sensor_3	%I0.2.2	Grab open / grab closed sensor
Auto_man	%I0.2.3	Select mode switch (0 = auto; 1 = manual)
Start_cycle	%I0.2.4	
Stop_cycle	%I0.2.5	
x_y_selection	%I0.2.6	Select axis manually to move or HOME (1 = x; 0 = y)
Home_man	%I0.2.7	
Forward	%I0.2.8	
Reverse	%I0.2.9	
Ack_fault	%I0.2.10	
Fault_type	%I0.2.11	
Enable_axes	%I0.2.12	Confirm axes
Open_grab	%I0.2.13	
Close_grab	%I0.2.14	

Output Module Symbols

The discrete input module is positioned in slot 3 of rack 0. The symbols are as follows

Symbol	Address	Comment
Grab	%Q0.3.0	Actuator command to open / close grab
Fault	%Q0.3.1	Fault indication

Description of the IODDTs of module TSX CSY 84

At a Glance

We have declared two IODDTs:

- Axis_1: associated with channel 1 of TSX CSY 84
- Axis_2: associated with channel 2 of TSX CSY 84

Once the two IODDTs have been associated, we obtain the correspondence described in the following paragraphs.

IODDT Axis_1

Elements of this IODDT used in this example are the following:

Symbol	Address	Comment
Axis_1.Ramping	%I1.2.1	
Axis_1.Steady	%I1.2.1.1	
Axis_1.Stopping	%I1.2.1.2	
Axis_1.Profile_end	%I1.2.1.3	
Axis_1.In_position	%I1.2.1.4	
Axis_1.Axis_homing	%I1.2.1.5	
Axis_1.Axis_homed	%I1.2.1.6	
Axis_1.Axis_not_following	%I1.2.1.7	
Axis_1.Holding	%I1.2.1.8	
Axis_1.Resuming	%I1.2.1.9	
Axis_1.Drive_enable	%I1.2.1.10	
Axis_1.Drive_diag	%I1.2.1.11	
Axis_1.Drive_warning	%I1.2.1.12	
Axis_1.Drive_flt	%I1.2.1.13	
Axis_1.Drive_disable	%I1.2.1.14	
Axis_1.Axis_summary_flt	%I1.2.1.15	
Axis_1.Axis_comm_ok	%I1.2.1.16	
Axis_1.Axis_is_linked	%I1.2.1.17	
Axis_1.Axis_in_command	%I1.2.1.18	
Axis_1.Axis_capture	%I1.2.1.19	
Axis_1.Axis_at_target	%I1.2.1.20	
Axis_1.Axis_pos_limit	%I1.2.1.21	
Axis_1.Axis_neg_limit	%I1.2.1.22	

Symbol	Address	Comment
Axis_1.Drive_realtime_bit1	%I1.2.1.26	
Axis_1.Drive_realtime_bit2	%I1.2.1.27	
Axis_1.Axis_hold	%I1.2.1.28	
Axis_1.Axis_halt	%I1.2.1.29	
Axis_1.Axis_faststop	%I1.2.1.30	
Axis_1.Axis_ready	%I1.2.1.31	
Axis_1.Position	%IF1.2.1	
Axis_1.Control_acquire	%Q1.2.1.2	
Axis_1.Control_enable	%Q1.2.1.10	
Axis_1.Control_resume	%Q1.2.1.12	
Axis_1.Control_clear_fit	%Q1.2.1.15	
Axis_1.Allow_capture	%Q1.2.1.16	
Axis_1.Allow_acquire	%Q1.2.1.18	
Axis_1.Allow_enable	%Q1.2.1.26	
Axis_1.Allow_follow	%Q1.2.1.27	
Axis_1.Allow_resume	%Q1.2.1.28	
Axis_1.Error_cmd	%MW1.2.1.19	
Axis_1.Return_1	%MD1.2.1.20	
Axis_1.Param_cmd_1	%MD1.2.1.27	
Axis_1.Param_cmd_2	%MD1.2.1.29	
Axis_1.Return_2	%MF1.2.1.22	
Axis_1.Return_3	%MF1.2.1.24	
Axis_1.Param_cmd_3	%MF1.2.1.31	
Axis_1.Param_cmd_4	%MF1.2.1.33	

IODDT Axis_2

Elements of this IODDT used in this example are the following:

Symbol	Address	Comment
Axis_2.Ramping	%I1.2.2	
Axis_2.Steady	%I1.2.2.1	
Axis_2.Stopping	%I1.2.2.2	
Axis_2.Profile_end	%I1.2.2.3	
Axis_2.In_position	%I1.2.2.4	
Axis_2.Axis_homing	%I1.2.2.5	

Symbol	Address	Comment
Axis_2.Axis_homed	%I1.2.2.6	
Axis_2.Axis_not_following	%I1.2.2.1	
Axis_2.Holding	%I1.2.2.8	
Axis_2.Resuming	%I1.2.2.9	
Axis_2.Drive_enable	%I1.2.2.10	
Axis_2.Drive_diag	%I1.2.2.11	
Axis_2.Drive_warning	%I1.2.2.12	
Axis_2.Drive_flt	%I1.2.2.13	
Axis_2.Drive_disable	%I1.2.2.14	
Axis_2.Axis_summary_flt	%I1.2.2.15	
Axis_2.Axis_comm_ok	%I1.2.2.16	
Axis_2.Axis_is_linked	%I1.2.2.17	
Axis_2.Axis_in_command	%I1.2.2.18	
Axis_2.Axis_capture	%I1.2.2.19	
Axis_2.Axis_at_target	%I1.2.2.20	
Axis_2.Axis_pos_limit	%I1.2.2.21	
Axis_2.Axis_neg_limit	%I1.2.2.22	
Axis_2.Drive_realtime_bit1	%I1.2.2.26	
Axis_2.Drive_realtime_bit2	%I1.2.2.27	
Axis_2.Axis_hold	%I1.2.2.28	
Axis_2.Axis_halt	%I1.2.2.29	
Axis_2.Axis_faststop	%I1.2.2.30	
Axis_2.Axis_ready	%I1.2.2.31	
Axis_2.Position	%IF1.2.2	
Axis_2.Return_1	%MD1.2.2.20	
Axis_2.Param_cmd_1	%MD1.2.2.27	
Axis_2.Param_cmd_2	%MD1.2.2.29	
Axis_2.Return_2	%MF1.2.2.22	
Axis_2.Return_3	%MF1.2.2.24	
Axis_2.Param_cmd_3	%MF1.2.2.31	
Axis_2.Param_cmd_4	%MF1.2.2.47	

Programming of preliminary processing

Preliminary section

This section, placed at the start of the task is used to initialize the chart:

```
! (*Chart initialization*)
```

```
IF NOT Start_cycle OR %S1 THEN SET %S21; END_IF;
```

```
! (*Initialization of movement values*)
```

```
%MF50 := %MF52 := 1000.0; (*Ready position*)
```

```
%MF54 := 2000.0; (*Move to belt B axis X*)
```

```
%MF56 := 1500.0; (*Move to belt B axis Y*)
```

```
%MF58 := 2000.0; (*Move to belt C axis X*)
```

```
%MF60 := 500.0; (*Move to belt C axis Y*)
```

```
%MF102 := 10000.0; (*Auto speed*)
```

```
%MF104 := 2000.0; (*Manual speed*)
```

```
%MF106:= 1000.0; (*Reference point speed*)
```

```
(*Initialization of channel 0*)
```

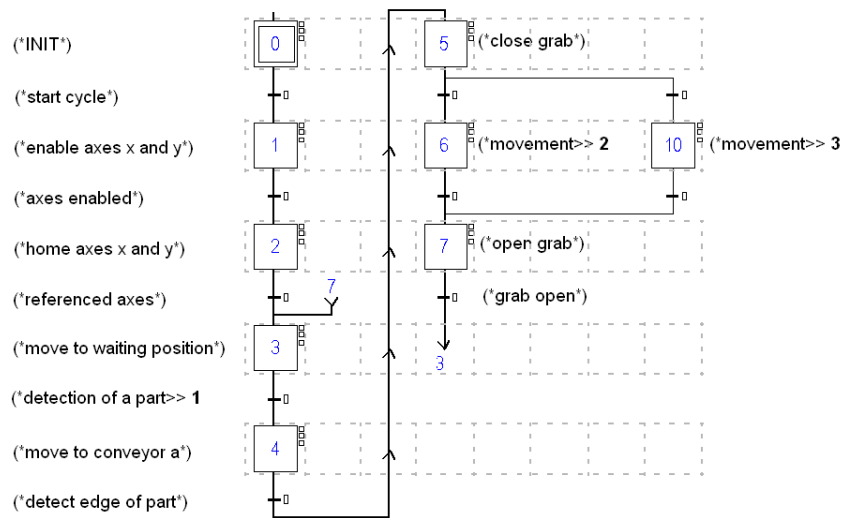
```
%Q1.2.26 := NOT %S0;
```

```
%L10:
```

SFC programming

SFC section

Description of chart in the example



List of connectors:

1	(*detection of a machined item*)
2	(*belt b movement*)
3	(*belt c movement*)

Transition programming

Step 0 -> 1

- Contents of the transition
! (*dcy and auto mode and no error *)
%I0.2.4 AND NOT %I0.2.3 AND NOT Axis_1.Axis_summary_flt AND
NOT Axis_2.Axis_summary_flt

Step 1 -> 2

- Contents of the transition
! (%m1 manual control // %mw2.x.3 = 1 -> ERR WRITE_CMD*)
Axis_1.Drive_enable AND Axis_2.Drive_enable

Step 2 -> 3

- Contents of the transition
! (*verification if drives are referenced and the command made*)
Axis_1.Axis_homed AND Axis_2.Axis_homed
AND (Axis_1.Err_write_cmd = 0) AND (Axis_2.Error_cmd = 0)
AND NOT Axis_1.Cmd_err
AND NOT Axis_2.Cmd_err (*check exchanges*)

Step 3 -> 4

- Contents of the transition
! (*verification => end of instruction and in position*)
Axis_1.Profile_end AND Axis_2.Profile_end AND
Axis_1.In_position AND Axis_2.In_position
(*check axes*)
AND %I3.0 (*part present*)
AND (Axis_1.Error_cmd = 0) AND (Axis_2.Error_cmd = 0) AND
NOT Axis_1.Cmd_err
AND NOT Axis_2.Cmd_err (*check exchanges*)

Step 4 -> 5

- Contents of the transition
! (*verification => end of instruction and in position*)
Axis_1.Profile_end AND Axis_2.Profile_end AND
Axis_1.In_position AND Axis_2.In_position
(*check axes*)
AND %I3.0 (*part present*)
AND (Axis_1.Error_cmd = 0) AND (Axis_2.Error_cmd = 0) AND
NOT Axis_1.Cmd_err
AND NOT Axis_2.Cmd_err (*check exchanges*)

Step 5 -> 6

- Contents of the transition
! (*machined item identification*)
%I0.2.1 AND %I0.2.2

Step 6 -> 7

- %X(6) -> %X(7)
! (*verification => end of instruction and in position*)
Axis_1.Profile_end AND Axis_2.Profile_end AND
Axis_1.In_position AND Axis_2.In_position
(*check axes*)
AND %I3.0 (*part present*)
AND (Axis_1.Error_cmd = 0) AND (Axis_2.Error_cmd = 0) AND
NOT Axis_1.Cmd_err
AND NOT Axis_2.Cmd_err (*check exchanges*)

Step 7 -> 3

- %X(7) -> %X(3)
! (*open grab ?*)
NOT %I0.2.2

Step 5 -> 10

- %X(5) -> %X(10)
! (*machined item identification*)
NOT %I0.2.1 AND %I0.2.2

Step 10 -> 7

- %X(10) -> %X(7)
! (*verification => end of instruction and in position*)
Axis_1.Profile_end AND Axis_2.Profile_end AND
Axis_1.In_position AND Axis_2.In_position
(*check axes*)
AND %I3.0 (*part present*)
AND (Axis_1.Error_cmd = 0) AND (Axis_2.Error_cmd = 0) AND
NOT Axis_1.Cmd_err
AND NOT Axis_2.Cmd_err (*check exchanges*)

Actions programming

Step 1

- Contents of action at step deactivation
! (*confirmation of axis channels 1 and 2*)
SET Axis_1.Control_enable; SET Axis_1.Control_acquire; SET
Axis_1.Control_enable; SET Axis_1.Control_acquire;
- Contents of action at step activation
! (*Initialization of values*)
RESET Axis_1.Control_enable; RESET Axis_1.Control_acquire;
RESET Axis_1.Control_enable; RESET Axis_1.Control_acquire;

Step 2

- Contents of action at step activation
! (*channel 2 reference point*)
Axis_2.Action_cmd := 6034; (**Reference point**)
Axis_2.Param_cmd_1 := 0; (**Param 1**)
Axis_2.Param_cmd_2 := 0; (**Param 2**)
Axis_2.Param_cmd_3 := 1.0; (**1 => positive direction, -1 negative
direction**)
Axis_2.Param_cmd_4 := %MF106; (**speed**)
WRITE_CMD(Axis_2);
! (*channel 1 reference point*)
Axis_1.Action_cmd := 6034; (**Reference point**)
Axis_1.Param_cmd_1 := 0; (**Param 1**)
Axis_1.Param_cmd_2 := 0; (**Param 2**)
Axis_1.Param_cmd_3 := 1.0; (**1 => positive direction, -1 => negative
direction**)
Axis_1.Param_cmd_4 := %MF106; (**speed**)
WRITE_CMD(Axis_1);

Step 3

- Contents of action at step activation
! (*channel 2 absolute movement*)
Axis_2.Action_cmd := 513; (**Move Immed**)
Axis_2.Param_cmd_1 := 0; (**Move type: -1 error, 0 ABS_MOVE, 1
INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG**)
Axis_2.Param_cmd_2 := 0; (**Param 2**)
Axis_2.Param_cmd_3 := %MF50; (Direction / Cont 1.0, Position / abs**)
Axis_2.Param_cmd_4 := %MF102; (**Speed**)
WRITE_CMD(Axis_2);
! (*channel 1 absolute movement*)
Axis_1.Action_cmd := 513; (**Move Immed**)

```

Axis_1.Param_cmd_1 := 0; (**Move type: -1 error, 0 ABS_MOVE, 1
INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG**)
Axis_1.Param_cmd_2 := 0; (**Param_2**)
Axis_1.Param_cmd_3 := %MF52; (Direction / Cont 1.0, Position / abs**)
Axis_1.Param_cmd_4 := %MF102; (**Speed**)
WRITE_CMD(Axis_1);

```

Step 4

- Contents of action at step activation

```

!(*channel 1 absolute movement*)
Axis_1.Action_cmd := 513; (**Move Immed**)
Axis_1.Param_cmd_1 := 0; (**Move type: -1 error, 0 ABS_MOVE, 1
INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG**)
Axis_1.Param_cmd_2 := 0; (**Param_2**)
Axis_1.Param_cmd_3 := 0.0; (Direction / Cont 1.0, Position / abs**)
Axis_1.Param_cmd_4 := %MF102; (**Speed**)
WRITE_CMD(Axis_1);
!(*channel 2 absolute movement*)
Axis_2.Action_cmd := 513; (**Move Immed**)
Axis_2.Param_cmd_1 := 0; (**Move type: -1 error, 0 ABS_MOVE, 1
INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG**)
Axis_2.Param_cmd_2 := 0; (**Param_2**)
Axis_2.Param_cmd_3 := 1000.0; (Direction / Cont 1.0, Position / abs**)
Axis_2.Param_cmd_4 := %MF102; (**Speed**)
WRITE_CMD(Axis_2);

```

Step 5

- Contents of action at step activation

```

!(*close grab*)
SET %Q0.3.0;

```

Step 6

- Contents of action at step activation

```

!(*channel 1 absolute movement*)
Axis_1.Action_cmd := 513; (**Move Immed**)
Axis_1.Param_cmd_1 := 0; (**Move type: -1 error, 0 ABS_MOVE, 1
INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG**)
Axis_1.Param_cmd_2 := 0; (**Param_2**)
Axis_1.Param_cmd_3 := %MF54; (Direction / Cont 1.0, Position / abs**)
Axis_1.Param_cmd_4 := %MF102; (**Speed**)
WRITE_CMD(Axis_1);
!(*channel 2 absolute movement*)
Axis_2.Action_cmd := 513; (**Move Immed**)

```

```
Axis_2.Param_cmd_1 := 0; (**Move type: -1 error, 0 ABS_MOVE, 1
INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG**)
Axis_2.Param_cmd_2 := 0; (**Param_2**)
Axis_2.Param_cmd_3 := %MF56; (Direction / Cont 1.0, Position / abs**)
Axis_2.Param_cmd_4 := %MF102; (**Speed**)
WRITE_CMD(Axis_2);
```

Step 7

- Contents of action at step activation
(*open grab*)

```
RESET %Q0.3.0;
```

Step 10

- Contents of action at step activation

```
! (*channel 1 absolute movement*)
```

```
Axis_1.Action_cmd := 513; (**Move Immed**)
```

```
Axis_1.Param_cmd_1 := 0; (**Move type: -1 error, 0 ABS_MOVE, 1
INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG**)
```

```
Axis_1.Param_cmd_2 := 0; (**Param_2**)
```

```
Axis_1.Param_cmd_3 := %MF58; (Direction / Cont 1.0, Position / abs**)
```

```
Axis_1.Param_cmd_4 := %MF102; (**Speed**)
```

```
WRITE_CMD(Axis_1);
```

```
! (*channel 2 absolute movement*)
```

```
Axis_2.Action_cmd := 513; (**Move Immed**)
```

```
Axis_2.Param_cmd_1 := 0; (**Move type: -1 error, 0 ABS_MOVE, 1
INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG**)
```

```
Axis_2.Param_cmd_2 := 0; (**Param_2**)
```

```
Axis_2.Param_cmd_3 := %MF60; (Direction / Cont 1.0, Position / abs**)
```

```
Axis_2.Param_cmd_4 := %MF102; (**Speed**)
```

```
WRITE_CMD(Axis_2);
```

Programming of subsequent processing

Posterior section

The posterior section is a section which completes the task and processes the actions to be performed.

It is described below

! (*-----RESET error management-----*)

```
IF %I0.2.10 THEN SET Axis_1.Control_clear_flg;
                SET Axis_2.Control_clear_flg;
                SET Axis_1.Control_acquire;
                SET Axis_2.Control_acquire;
ELSE RESET Axis_1.Control_clear_flg;
      RESET Axis_2.Control_clear_flg;
      RESET Axis_1.Control_acquire;
      RESET Axis_2.Control_acquire;

END_IF;
```

! (*-----Manual mode management-----*)

```
(*Initialize G7*)
IF %I0.2.3 THEN SET %S21;
END_IF;
(*Enable axes for manual mode*)
IF %I2.10 THEN SET Axis_1.Control_enable;
                SET Axis_2.Control_enable;
                SET Axis_1.Control_acquire;
                SET Axis_2.Control_acquire;
ELSE RESET Axis_1.Control_enable;
      RESET Axis_2.Control_enable;
      RESET Axis_1.Control_acquire;
      RESET Axis_2.Control_acquire;

END_IF;
```

(*****Positive x channel manual mode*****)

```
IF RE%I0.2.8 AND %I0.2.6 THEN
  Axis_1.Action_cmd := 513; (**Move Immed**)
  Axis_1.Param_cmd_1 := 2; (**Move type 2 = CONT_MOVE**)
  Axis_1.Param_cmd_2 := 0; (**Param_2**)
  Axis_1.Param_cmd_3 := 0.0; (**Direction**)
  Axis_1.Param_cmd_4 := %MF04; (**Speed**)
  WRITE_CMD (Axis_1);
END_IF;
```

(*****Negative x channel manual mode*****)

```
IF RE%IO.2.9 AND %IO.2.6 THEN
    Axis_1.Action_cmd := 513; (**Move Immed**)
    Axis_1.Param_cmd_1 := 2; (**Move type 2 = CONT_MOVE**)
    Axis_1.Param_cmd_2 := 0; (**Param_2**)
    Axis_1.Param_cmd_3 := -1.0; (**Direction**)
    Axis_1.Param_cmd_4 := %MF_04; (**Speed**)
    WRITE_CMD (Axis_1);
END_IF;
IF FE %IO.2.9 AND %IO.2.6 THEN Axis_1.Action_cmd := 407;
WRITE_CMD (Axis_1);
END_IF;
```

(*****Positive y channel manual mode*****)

```
IF RE%IO.2.8 AND NOT %IO.2.6 THEN
    Axis_2.Action_cmd := 513; (**Move Immed**)
    Axis_2.Param_cmd_1 := 2; (**Move type 2 = CONT_MOVE**)
    Axis_2.Param_cmd_2 := 0; (**Param_2**)
    Axis_2.Param_cmd_3 := 0.0; (**Direction**)
    Axis_2.Param_cmd_4 := %MF_04; (**Speed**)
    WRITE_CMD (Axis_2);
END_IF;
IF FE %IO.2.8 AND NOT %IO.2.6 THEN Axis_2.Action_cmd := 407;
WRITE_CMD (Axis_2);
END_IF;
```

(*****Negative y channel manual mode*****)

```

IF RE%IO.2.9 AND NOT %IO.2.6 THEN
    Axis_2.Action_cmd := 513; (**Move Immed**)
    Axis_2.Param_cmd_1 := 2; (**Move type 2 = CONT_MOVE**)
    Axis_2.Param_cmd_2 := 0; (**Param_2**)
    Axis_2.Param_cmd_3 := -1.0; (**Direction**)
    Axis_2.Param_cmd_4 := %MF04; (**Speed**)
    WRITE_CMD (Axis_2);
END_IF;
IF FE %IO.2.9 AND NOT %IO.2.6 THEN Axis_2.Action_cmd := 407;
WRITE_CMD (Axis_2);
END_IF;

```

(*****Manual homing of channel x*****)

```

IF RE %IO.2.7 AND %IO.2.6 THEN
    Axis_1.Action_cmd := 6034; (**Reference point**)
    Axis_1.Param_cmd_1 := 0; (**Param_1**)
    Axis_1.Param_cmd_2 := 0; (**Param_2**)
    Axis_1.Param_cmd_3 := 1.0; (**Direction**)
    Axis_1.Param_cmd_4 := %MF06; (**Speed**)
    WRITE_CMD (Axis_1);
END_IF;

```

(*****Manual homing of channel y*****)

```

IF RE %IO.2.7 AND NOT %IO.2.6 THEN
    Axis_2.Action_cmd := 6034; (**Reference point**)
    Axis_2.Param_cmd_1 := 0; (**Param_1**)
    Axis_2.Param_cmd_2 := 0; (**Param_2**)
    Axis_2.Param_cmd_3 := 1.0; (**Direction**)
    Axis_2.Param_cmd_4 := %MF06; (**Speed**)
    WRITE_CMD (Axis_2);
END_IF;

```

! (*-----Stop cycle during HALT instruction-----*)

```

IF RE%IO.2.5 THEN
    Axis_1.Action_cmd := 407; WRITE_CMD (Axis_1);
    Axis_2.Action_cmd := 407; WRITE_CMD (Axis_2);
END_IF;

```

! (*-----Error type read-----*)

```
IF RE%IO.2.11 THEN
    Axis_1.Action_cmd := 5510; WRITE_CMD (Axis_1);
    Axis_2.Action_cmd := 5510; WRITE_CMD (Axis_2);
END_IF;
```

! (*-----Channel disablement-----*)

```
(*disable drive on channel 2 (y axis)*)
IF %M4 THEN Axis_2.Action_cmd := 402; WRITE_CMD (Axis_2); END_IF;
(*disable drive on channel 1 (x axis)*)
IF %M4 THEN Axis_1.Action_cmd := 402; WRITE_CMD (Axis_1); END_IF;
```


Program transfer

Procedure for transferring a program

Once you have configured your project and entered your program, you must transfer these into the memory of the PLC processor in accordance with the following procedure:

Step	Action
1	Choose the standard operating mode by clicking on the Standard mode icon from the toolbar.
2	Connect the terminal to the PLC using the command PLC →Connect .
3	Activate the command PLC →Transfer Project to PLC .
4	Confirm the transfer.

Adjusting the Parameters

Introduction

If needed, it can be important to be able to adjust the configuration parameters.

There are several possible ways of doing this:

- you can return to offline mode and modify the parameters directly from the configuration screen,
- Unity Pro also allows you to modify the parameters either in online mode or using functions in the project adjustment parameters.

Restore/Save parameters

This service is available for the Network channel, the real, imaginary and remote axes as well as for the slave axes.

Introduction to the Debug Mode

Introduction

Once the application has been performed then transferred, it is often necessary to debug it.

Unity Pro will show you how to do this with the help of tools such as:

- debugging screens that:
 - have numerous indicators of the status of the application
 - give access to the change of status of certain variables

NOTE: The debug function has to be in online mode.

Debug procedure

You can perform the program debugging in the following way:

Step	Action
1	Set the PLC to RUN.
2	Display the debug screen of the TSX CSY 84/164 module. Note: You can simultaneously display a language editor in order to follow the development of the application.
3	Begin executing the chart.

Hardware installation for the modules TSX CSY 84/85/164



Subject of this Part

This part introduces hardware installation for the TSX CSY 84, 85 and 164 modules.

NOTE: In this section, all the data relating to TSX CSY 84 mounting is also applicable to the TSX CSY 164 and TSX CSY 85 . Moreover, the TSX CSY 84 is taken as an example.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	Presentation of the TSX CSY 84 / 164 modules	63
5	Installing the modules	69
6	Description of the multi-axis control system	81
7	Fiber Optic Cables	87
8	Characteristics, Standards and service conditions	91
9	Compatible servodrives	95

Presentation of the TSX CSY 84 / 164 modules

4

Subject of this Chapter

This chapter presents the TSX CSY 84 and TSX CSY 164 multi-axis control modules.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Introduction to the modules in their environment	64
Physical presentation of the modules	66

Introduction to the modules in their environment

Introduction

The TSX CSY 84 and TSX CSY 164 modules are double format application-specific modules from the Premium range that can be mounted in the TSX RKY** rack of a TSX/PCI 57 PLC station. They are one of the elements of the SERCOS® offer for Premium/Atrium PLCs enabling the creation of a multi-axis control system.

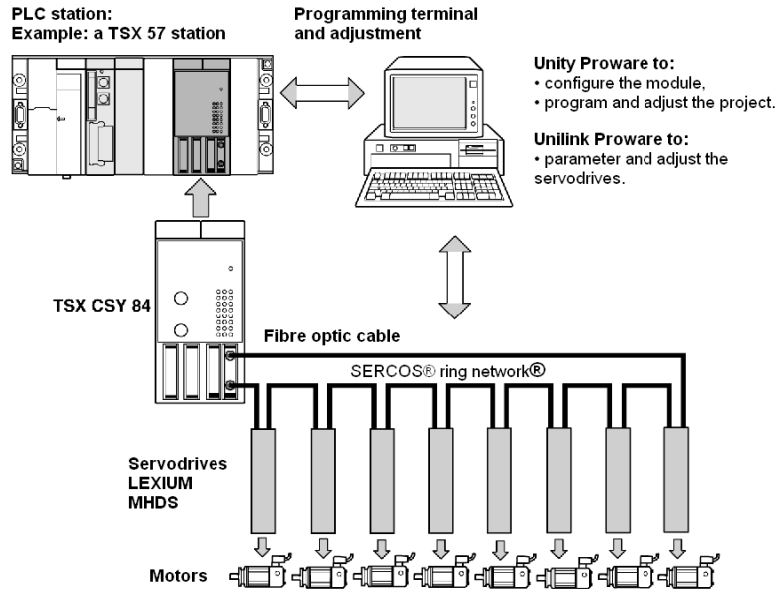
Introduction of the elements of the SERCOS® offer for Premium PLCs

The SERCOS® offer for the Premium PLC is composed of the following elements:

- a TSX/PCI 57 PLC station composed of:
 - one or more racks,
 - power supply modules,
 - a processor module,
 - the different modules required by the project.
- one or more TSX CSY 84 or TSX CSY 164 multi-axis control modules enabling each one to control up to 8 servodrives within a SERCOS® network.
- a range of 5 LEXIUM MHDS servodrives,
- a range of motors,
- plastic fiber optic cables (0.3 to 16.5 meters long):
 - that ensure physical connection of the modules/servodrives and servodrives/servodrives in a ring network structure,
 - that servoserve as media for the digital link between the TSX CSY 84 or TSX CSY 164 module (master) and the servodrives (slaves); this digital link is defined by European standard EN61491.
- Unity Proware is used to configure the TSX CSY 84 and TSX CSY 164 modules and program the motion project.
- UniLink LXM 17 software for setting up and making adjustments to servodrives.

Installation overview

The diagram below shows an example of a SERCOS® multi-axis control installation with the TSX CSY 84 module



Physical presentation of the modules

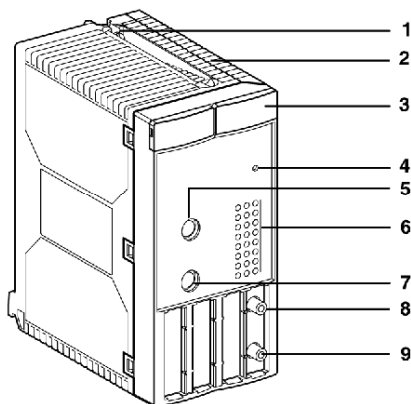
Description of the front panel

The following can be found on the front panel of the modules:

- a display block comprising 6 LEDs for viewing and diagnosing the module status,
- a group of 24 LEDs for viewing and diagnosing the various module channels,
- two connectors for connecting the fiber optic cables which make up the link between the module and variable speed controllers,
- two 8-pin Mini DIN connectors.

Appearance of the module

The diagram below shows the TSX CSY 84 module and its various elements.



Elements and their functions

Number	Elements	Functions
1	Screw	Attaches the module to the TSX RKY ** rack
2	Module casing	Does the following: • Supports and protects the electronic boards, • Locks the module into its slot.
3	Display block comprising 6 LEDs: • green RUN LED • red ERR LED • red I/O LED • yellow SER LED • yellow COM LED • yellow INI LED	These enable the user to visualize the status and errors of the module. Indicates the operating mode of the module. Indicates an internal module error. Indicates an external module error or an application error. Indicates that traffic on the SERCOS® network is flowing correctly. Not significant Indicates that the module is in re-initialization phase.
4	Recessed button	Used to initialize the module.
5	COM2 8-pin Mini DIN connector	Reserved.
6	24 LED indicators	Used to visualize and diagnose the module channels.
7	COM1 8-pin Mini DIN connector	Reserved.
8	TX send SMA connector	Used to connect the send fiber optic cable of the SERCOS® ring network.
9	RX receive SMA connector.	Used to connect the receive fiber optic cable of the SERCOS® ring network.

Installing the modules



Subject of this Chapter

This chapter describes the installation operations for the TSX CSY 84 / 164 multi-axis control modules.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Mounting modules in a PLC station rack	70
Installing the modules in a PLC station	71
Number of application specific channels managed by a TSX CSY 84 or CSY 164 Station	72
Installation precautions	73
Module displays	74
Initialization of the module following an internal error	78
TSX CSY 84 and TSX CSY 164 module operating mode	79

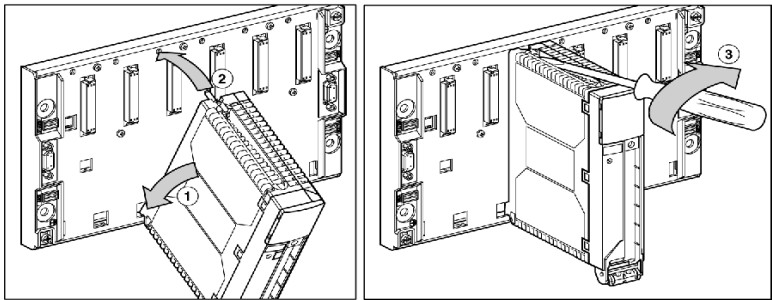
Mounting modules in a PLC station rack

Introduction

The TSX CSY 84/164 modules can be mounted in all available slots on a TSX RKY •• rack for a TSX 57/PCX 57/PCIX 57 PLC, with the exception of slots reserved for power supply modules and the processor. This double-format module occupies 2 slots on a TSX RKY •• rack.

Illustration

The diagrams below introduce the mounting procedure for a standard-format Premium range module on the TSX RKY •• rack. The procedure is the same for a double-format module.



Procedure

The following table describes the operations to be carried out.

Steps	Actions
1	Position the pins on the reverse side of the module in the centering holes located on the bottom part of the rack.
2	Rotate the module so that it comes into contact with the rack.
3	Connect the module to the rack using the screw located on the upper part of the module. Maximum tightening torque: 2.0 N. m.

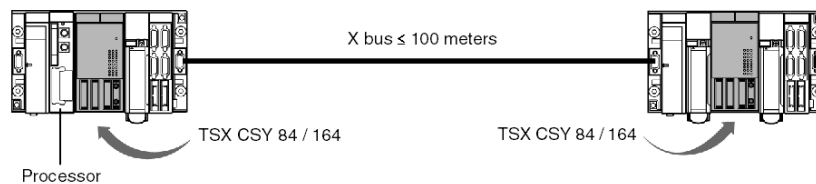
Installing the modules in a PLC station

Introduction

The TSX CSY 84 / 164 modules can be installed in all the racks located on the main X bus segment of a PLC station.

Installing the module

The diagram below represents the installation of a TSX CSY module in a rack belonging to the main X bus segment. The module can be installed in the rack supporting the processor and in all other racks present on the X bus. The distance between the rack supporting the TSX CSY module and the rack supporting the processor must not exceed 100 meters.



NOTE: The TSX CSY 84 / 164 modules cannot be installed in a rack belonging to a remote X bus segment by a TSX REY 200 module.

Number of application specific channels managed by a TSX CSY 84 or CSY 164 Station

Definition of an Application-Specific Channel

An application-specific module (TSX CTY• counting modules, TSX CAY• axis control modules, TSX CFY • step-by-step control modules, the TSX YSP Y• weighing module, the TSX CCY 1128 electronic cam module, TSX CSY 84 or TSX CSY 164 motion control modules, etc.) has a number of channels which may vary from 1 to n depending on the type of module (see *Premium and Atrium using Unity Pro, Processors, racks and power supply modules, Implementation manual*); these channels are called application-specific channels.

Counting the Number of Application-Specific Channels on the Station

To define:

- the power of the processor to be installed,
- the maximum number of application-specific modules that can be installed on the station.

Number of Application-Specific Modules for a TSX CSY 84 Module

A TSX CSY 84 module may have up to 32 application-specific channels. Only the configured application-specific channels should be counted.

The TSX CSY 84 module manages a maximum of 8 real axes (channels 1 to 8), which are associated with servodrives. In addition to these real axes, the module can manage:

- 4 imaginary axes (channels 9 to 12),
- 4 axes with remote input (channels 13 to 16),
- 4 groups of coordinated axes (channels 17 to 20).
- 4 groups of slave axes (channels 21 to 24).
- 7 cam profiles (channels 25 to 31).

Channel 0 (SERCOS®) is the digital bus management channel,

Number of Application-Specific Modules for a TSX CSY 164 Module

- channels 1 to 16 can each support either a real axis function, an imaginary axis function, or an external setpoint:
- 4 groups of coordinated axes (channels 17 to 20).
- 4 groups of slave axes (channels 21 to 24).
- 7 cam profiles (channels 25 to 31).

Channel 0 (SERCOS®) is the digital bus management channel,

Installation precautions

Introduction

In order to guarantee optimum operating conditions, it is necessary to take certain precautions when installing and removing a module, plugging and unplugging connectors to and from the front panel of the module, and adjusting its fixing screws.

Installing and removing a module

Installing and removing a module should be done with the power off. However, a module can be installed or removed without cutting the rack's power and without any risk of damaging the module.

Screwing in and unscrewing the fiber optic connectors on the front panel of the module

It is possible to screw in and unscrew the fiber optic connectors on the front panel of the module with the module powered up without any risk of damaging it.

WARNING

UNEXPECTED APPLICATION BEHAVIOR

Before connecting / disconnecting the fiber optic connectors, ensure that the consequences on the application are acceptable or power off the equipment.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Tightening torque of the module's fixing screw

Tightening torque: 2.0 Nm.

Module displays

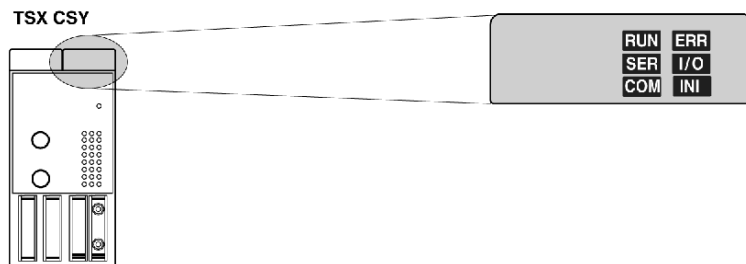
Role

The TSX CSY 84 or TSX CSY 164 module is equipped with two display units:

- A display block in standard Premium composed of 6 LEDs, whose function is to inform the user of:
 - the operating mode of the module: module operation normal, in error or switched off,
 - internal or external module operating errors.
- A set of 24 LEDs for displaying the status of the application-specific channels of the module (real axes, imaginary axes, etc.).

Presentation of the display block

The diagram below shows the physical appearance of the module's display block and the location of its six LEDs.



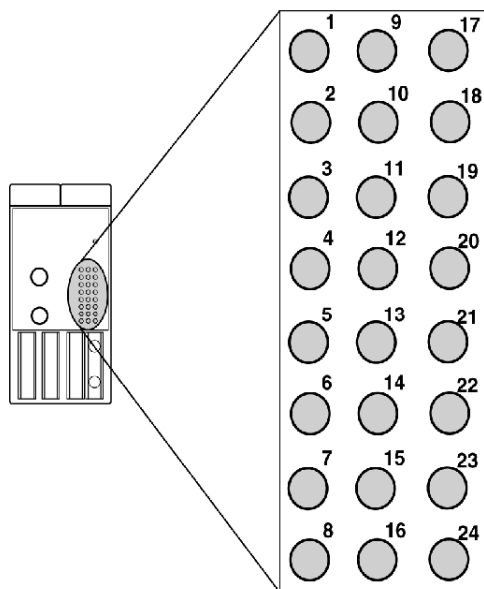
Statuses of the various display block LEDs and their meaning

The following tables show the different statuses for each LED on the display block and their meaning.

LED	Color	Status	Meaning
RUN	Green	On	Module in normal operating mode
		Off	Faulty module, module powered down, in initialization phase or project missing
ERR	Red	On	Internal module error: • module has broken down.
		Blinking	• Start-up of the module, • Communication error, • Project missing, invalid or execution error.
		Off	Operating normally, no faults
I/O	Red	On	External module error: • Cabling error
		Blinking	Not significant
		Off	Operating normally, no faults
INI	Yellow	On	Not significant
		Blinking	The module is in Reset phase
		Off	Operating normally
SER	Yellow	On	Not significant
		Blinking	Traffic on the SERCOS® network in normal operating mode
		Off	No traffic on the SERCOS network
COM	Yellow	-	Unused

Presentation of the display LEDs of the application-specific channels

The diagram below shows the physical characteristics of the 24 display LEDs for certain application-specific channels.



TSX CSY 84

LEDs 1 to 8: display of the 8 imaginary axes

LEDs 9 to 12: display of the 4 imaginary axes

LEDs 13 to 16: display of the 4 external measurement axes

LEDs 17 to 20: display of the 4 coordinated sets of axes

LEDs 21 to 24: display of the 4 follower sets of axes

TSX CSY164

LEDs 1 to 16: display of independent axes (real and imaginary axes, and external setpoint)

LEDs 17 to 20: display of the 4 coordinated sets of axes

LEDs 21 to 24: display of the 4 follower sets of axes

Statuses and meanings of the application-specific channel display LEDs

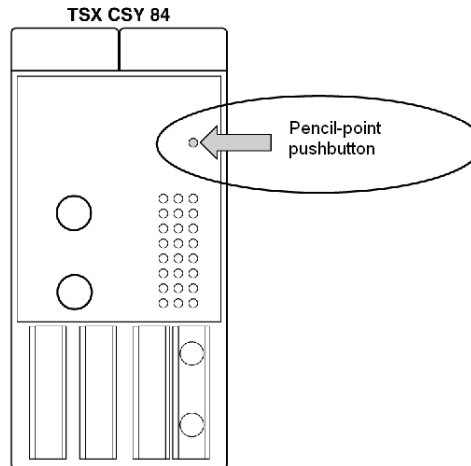
The table below shows the different statuses of the application-specific channel display LEDs:

LEDs	CSY 84 Assignment	CSY 164 Assignment	Statuses	Meanings
1 2 3 4 5 6 7 8	Real axis 1 Real axis 2 Real axis 3 Real axis 4 Real axis 5 Real axis 6 Real axis 7 Real axis 8	either real axis, imaginary axis, external setpoint	On	Axis in normal operating mode
9 10 11 12	Imaginary axis 1 Imaginary axis 2 Imaginary axis 3 Imaginary axis 4		Blinking	Axis in configuration mode or in error
13 14 15 16	External setpoint 1 External setpoint 2 External setpoint 3 External setpoint 4			
17 18 19 20	Coordinated set of axes 1 Coordinated set of axes 2 Coordinated set of axes 3 Coordinated set of axes 4		Off	Axis not configured or configuration error
21 22 23 24	Follower set of axes 1 Follower set of axes 2 Follower set of axes 3 Follower set of axes 4			

Initialization of the module following an internal error

How to initialize the module

The module is initialized using the pencil-point button located on the front panel as shown in the diagram below.



NOTE: Recommendations concerning physical action on the pencil-point button
Pressure applied to this push button should be moderate.

The point of the tool must be held perpendicular to the front panel of the module and centered in the access window.

Non-observation of this recommendation can cause damage to the push button.

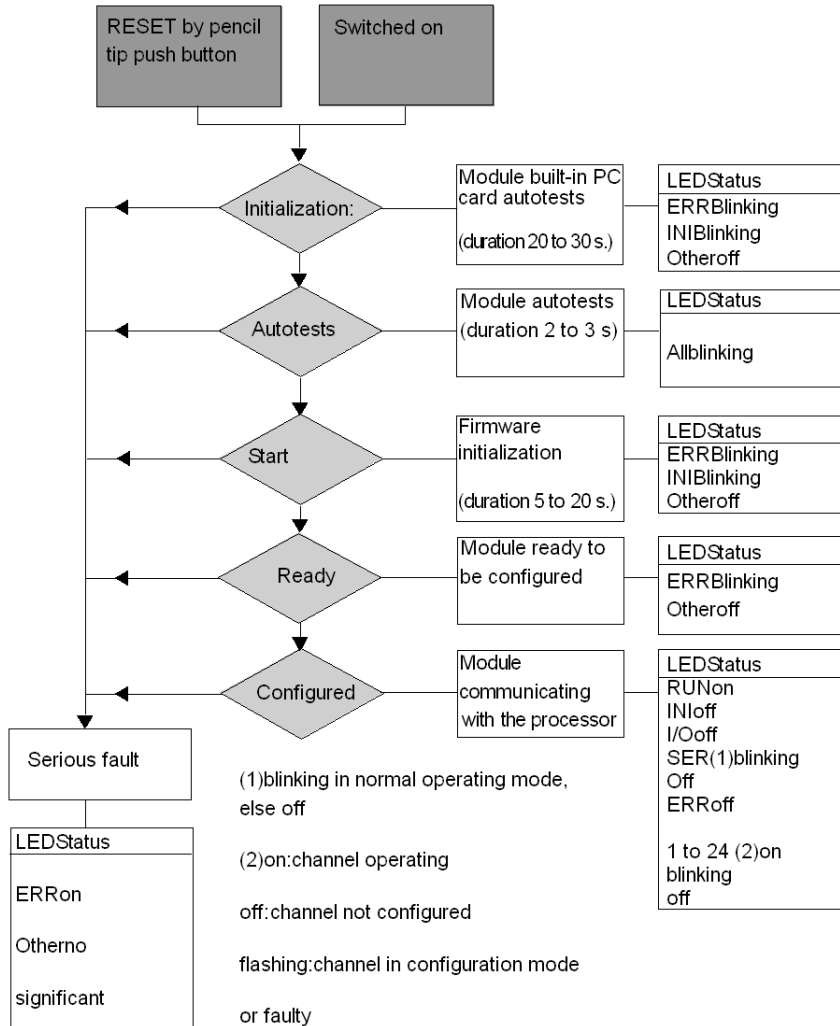
Consequence of an initialization

The module restarts in initialization phase in the same way as when it is powered up. The operating modes are described in what follows.

TSX CSY 84 and TSX CSY 164 module operating mode

Overview of the module operating mode

The overview below describes the different operating mode stages and gives the status of the indicator lamps on the front panel of the module for each step.



Description of the multi-axis control system



Subject of this Chapter

This chapter describes how the TSX CSY 84 and TSX CSY 164 modules interface with the variable speed controllers in a SERCOS® network configuration to form a multi-axis control.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Multi-axis control system for SERCOS® Premium PLC	82
SERCOS® ring network	84

Multi-axis control system for SERCOS® Premium PLC

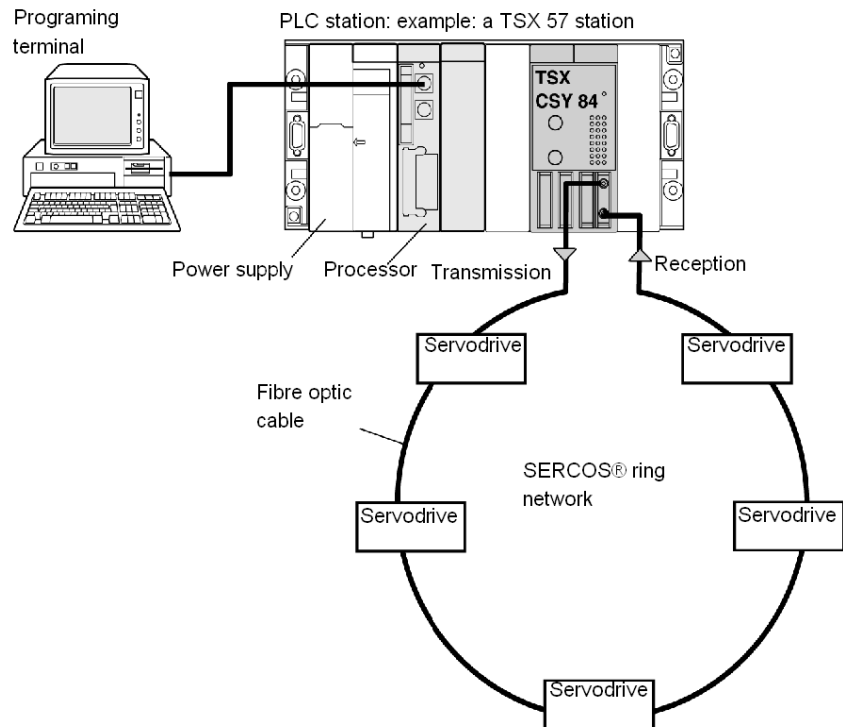
Architecture of a SERCOS® multi-axis control system

The architecture of a SERCOS® multi-axis control system on Premium PLCs includes:

- a TSX/PCI 57 PLC station,
- a TSX CSY 84 or TSX CSY 164 multi-axis control module
- servodrives that control the motors associated with the different axes,
- a fiber optic SERCOS® network.

Illustration

The diagram below gives an example of the architecture of a SERCOS® multi-axis control system.



Operating principle

The multi-axis control system consists of the TSX CSY 84 or TSX CSY 164 multi-axis control modules and the servodrives, interconnected in a network via fiber optic cables.

The drives, which are interconnected through the fiber optic network, act as individual axes.

The motion commands sent by the TSX CSY 84 or TSX CSY 164 modules are sent to each drive in the network and in return, the module receives the actual position values for each axis from the network.

Maximum number of real axes managed by a TSX CSY 84 module

A TSX CSY 84 module manages a maximum of 8 real axes associated with servodrives.

In addition to these real axes, the module can manage:

- 4 imaginary axes,
- 4 remote axes,
- 4 coordinated axis groups,
- 4 slave axis groups,
- 7 cam profiles.

Number of Application-Specific Modules for a TSX CSY 164 Module

- Channels 1 to 16 can each support either a real axis function, an imaginary axis function, or an external setpoint:
- 4 groups of coordinated axes (channels 17 to 20).
- 4 groups of slave axes (channels 21 to 24).
- 7 cam profiles (channels 25 to 31).

Project development

Project development can be implemented on a terminal (PC) with Unity Pro software, allowing:

- axes configuration,
- commissioning,
- project adjustment and diagnostics.

SERCOS® ring network

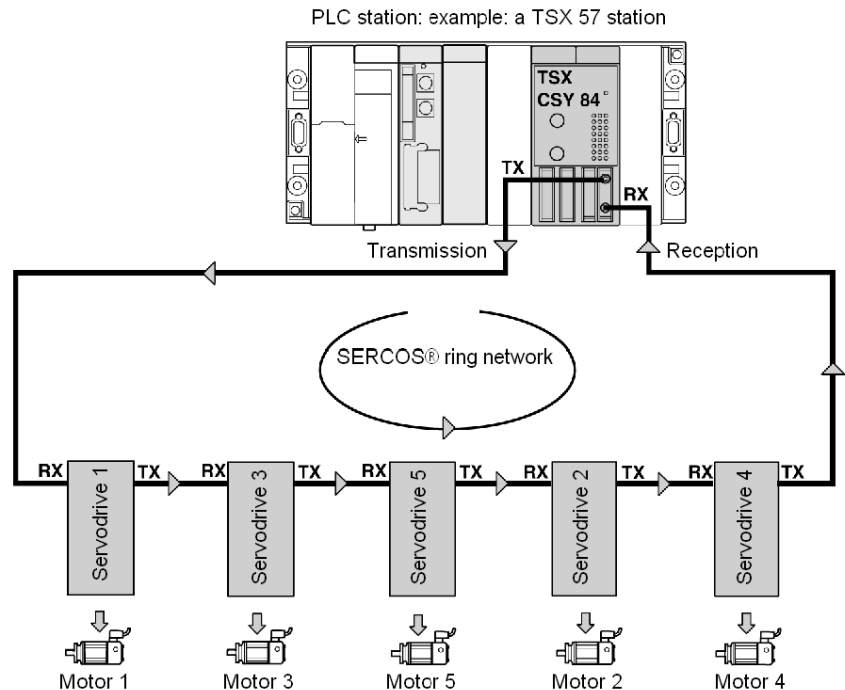
At a Glance

Via the SERCOS® fiber optic ring network, the TSX CSY 84/164 :

- transmits the motion control instructions defined by the project program to each axis,
- in return, receives actual data sent by the different axis position sensors from the servodrives.

SERCOS® ring network

The diagram below shows an example of a configuration of a SERCOS® network consisting of five LEXIUM servodrives interconnected by fiber optic cable to a TSX CSY 84 module.



Sending commands and receiving data

The module coordinates the motion activities of the different axes installed on the network :

- it sends motion control instructions for the different axes to the network and to the servodrives,
- it receives actual data for each axis from the network and executes the necessary processing of this data as required.

Via the (TX) connector and fiber optic cable, the module sends move instructions to the first drive, which interprets and executes them; the instructions are then sent to the next drive.

The last drive in the ring sends the actual data for all axes onto the connector (RX) via fiber optic cable.

NOTE: a servodrive that is powered down on the SERCOS® network causes the ring to open, thus creating a system fault.

Cycle time

Data is transmitted on the network in a single direction with typical cycle time of 4 ms. This time can be reduced to 2 ms in configuration mode if permitted by the volume of data exchanged.

Transmission speed

Transmission speed is defined by default at 4 Mbauds. If the drives cannot support this speed, it can be reduced to 2 Mbauds in configuration mode.

Maximum length of the different network segments

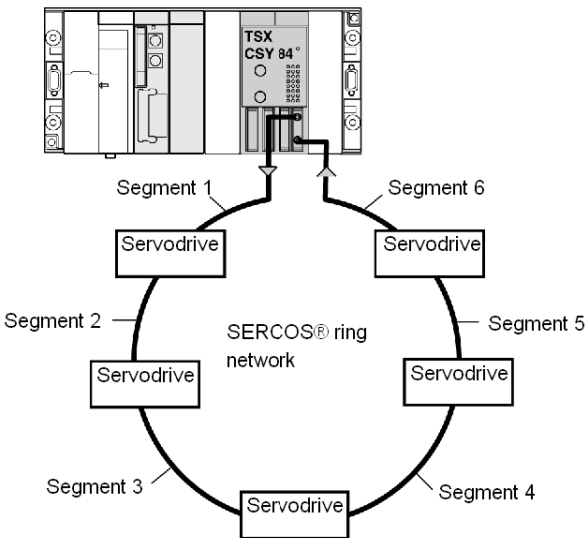
The maximum length of each segment of the SERCOS® network is limited to 40 meters, using plastic fiber optic cable recommended by Schneider Electric.

Illustration of different segments

The different segments of the SERCOS® network :

- TSX CSY 84 module / drive : segment 1
- Drive / drive: segment 2 to n
- Drive /module: segment n+1

The diagram below shows the different segments of a SERCOS® network with 5 servodrives connected to it.



Adjustment of optical power of transmitter depending on the length of the segment

Each element of the SERCOS® network (TSX CSY 84 module and servodrives) has an optical transmitter.

For each optical transmitter, the operator must adjust the optical power according to the segment length.

- Optical power of segment 1 (module / first drive): it is always provided by the TSX CSY 84 module optical transmitter. Optical power adjustment is carried out in configuration mode via Unity Pro by entering a percentage of optical power according to segment length.

Segment length (in meters)	Optical power (as a percentage of total power)
$0 < L < 15$	66%
$15 < L < 40$	100%

- Optical power of other segments (drive / drive and last drive / module) : it is always provided by the drive's optical transmitter. Optical power is adjusted via UniLink software by setting parameters for segment length only.

Fiber Optic Cables



Subject of this Chapter

This chapter presents the fiber optic cables for connecting the various components of the SERCOS network (TSX CSY 84 / 164 modules and variable speed controllers).

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Pre-Assembled Fiber Optic Cables	88
Kits for creating custom fiber optic cables	89

Pre-Assembled Fiber Optic Cables

At a Glance

Schneider Automation offers a range of 1 mm diameter plastic fiber optic cables for connecting the different components of the SERCOS network (TSX CSY 84/164 modules and variable speed controllers). Each cable is equipped at each end with an SMA type connector.

List of Cables

The table below provides the reference and length of each cable.

References	Lengths
990 MCO 00001	0.3 meters
990 MCO 00003	0.9 meters
990 MCO 00005	1.5 meters
990 MCO 00015	4.5 meters
990 MCO 00055	16.5 meters
990 MCO 00075	22.5 meters
990 MCO 000125	37.5 meters

Recommendations

When installing fiber optic cables, it is necessary to comply with the following recommendations:

NOTE: Recommendations concerning the curve radius of the cables.

For this type of cable, the minimum curve radius must be greater than 25 mm.

Failure to observe this may result in damage to the cables.

NOTE: Recommendation concerning the tension applied to the cables on installation.

The maximum tension applied to the cables on installation must not exceed 6 Kg.

Failure to observe this may result in damage to the cables.

Maximum permitted temperature: -40° C+80° C.

Kits for creating custom fiber optic cables

At a Glance

Schneider Automation offers two kits for creating custom cables:

- 1 tool kit,
- 1 equipment kit comprising a cable and connectors.

Tool kit

The table below provides the reference and the contents of the tool kit.

Reference	Contents	
	Quantity	Description
990 MCO KIT 00	1	Service instructions for creating a cable
	1	Cable stripping tool
	1	Connector clip pliers
	1	25 W / 110 V soldering iron

Equipment kit

The table below provides the reference and the contents of the equipment kit.

Reference	Contents	
	Quantity	Description
990 MCO KIT 01	12	SMA type connectors
	12	Insulation sleeves
	1	30-meter plastic fiber optic cable

Characteristics, Standards and service conditions

8

Subject of this Chapter

This chapter presents the different characteristics of the TSX CSY 84 module and the SERCOSnetwork.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Characteristics of the module	92
Characteristics of the SERCOS® network	93
Standards and Operating Conditions	94

Characteristics of the module

Electrical characteristics

The following table gives the electrical characteristics of the module.

Description of the parameters	Values	
	Typical	Maximum
Current consumed by the module from the 5V of the rack power supply	1.8 A	2 A
Power dissipated in the module	9 W	10 W
Fiber optic outputs	Complies with standard EN 61491	

Operating and storage temperature/hygrometry/altitude

The following table gives the electrical characteristics of the module.

Description of the parameters	Values
Operating temperature	0 to 60° C
Storage temperature	-25° C to 70° C
Hygrometry (without condensation)	5% to 95%
Operating altitude	0 to 2000 m

Characteristics of the SERCOS® network

Table of characteristics

The following table gives the main characteristics of the SERCOS® network.

Description of the parameters	Values
Addresses	1...254
Baud rate	2 or 4 Mbauds, configurable by software
Cycle time	4 ms

Standards and Operating Conditions

Standards

Standards identical to those applicable to Premium/Atrium PLCs.

Standard EN 61491:

Electrical equipment for industrial machines. Serial data links for real-time communication between control units and drives.

Operating Conditions and Environmental Requirements

These are identical to those applicable to Premium/Atrium PLCs.

List of variable speed controllers

List of variable controllers in the Schneider Electric range

Schneider Electric offers a range of variable speed controllers that are compatible with the SERCOS range on Premium PLCs. The table below provides the reference and description of each variable speed controller.

Reference	Description
MHD• 1004 •00	LEXIUM variable speed controller, continuous current of 1.5 A rms
MHD• 1008 •00	LEXIUM variable speed controller, continuous current of 3 A rms
MHD• 1017 •00	LEXIUM variable speed controller, continuous current of 6 A rms
MHD• 1028 •00	LEXIUM variable speed controller, continuous current of 10 A rms
MHD• 1056 •00	LEXIUM variable speed controller, continuous current of 20 A rms
MHD• 1112 •00	LEXIUM variable speed controller, continuous current of 40 A rms
MHD• 1198 •00	LEXIUM variable speed controller, continuous current of 70 A rms

Other variable speed controllers

All EN 61491-compliant variable speed controllers may be used with the TSX CSY 84/164 modules.

Motion control common to TSX CSY 84 and TSX CSY 164 modules



Subject of this section

This section introduces the TSX CSY 84 and TSX CSY 164 modules and describes the implementation of a motion control using these modules.

NOTE: In this section, all the data relating to TSX CSY 84 mounting is also applicable to the TSX CSY 164. Moreover, the TSX CSY 84 is taken as an example.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
10	Functional Description	99
11	Configuration	115
12	Programming the TSX CSY 84 and TSX CSY 164 modules	137
13	Adjusting the TSX CSY 84 and TSX CSY 164 Modules	231
14	Debugging the TSX CSY 84 / 164 modules	321
15	Diagnostics and Maintenance	341
16	SERCOS language objects	385
17	Appendices	453

Subject of this Chapter

This chapter describes the functionalities of module TSX CSY 84 and the functions it can perform.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Module Functions	100
"SERCOS®" Function	102
"Real Axis" Function	103
"Imaginary Axis" Function	104
"Remote Axis" Function	105
Concept of the Group	107
"Coordinated Axis Group" Function	108
"Slave Axis Group" Function	110
"Cam Profile" Function	113

Module Functions

At a Glance

The TSX CSY 84/164 modules have 5 types of application specific functions:

- SERCOS® function, installed by channel 0 (SERCOS® ring),
- Independent axis function (real, imaginary, remote axes), installed by channels 1 – 16,
- Co-ordinated axis group function, installed by channels 17 – 20,
- Slave axis group function, installed by channels 21 – 24,
- Cam profile function, installed by channels 25– 31.

NOTE: Unity accepts the MOD_PARAM function when used with the TSX CSY 84 but it is not operational. Hence, the MOD_PARAM function is not available for the TSX CSY 84 module but only for the TSX CSY 164 module.

Command Exchange

For each of the application-specific functions (SERCOS® ring, individual axis, co-ordinated group, slave group and Cam profile), exchanges are made at the rate of 1 command per channel for every 2 scans of the MAST PLC task.

The following 32 command bits enable commands to be authorized and confirmed:

- 16 bits CONTROL, active on the rising edge (%Qr.m.c.0 to %Qr.m.c.15),
- 16 bits ALLOW, active on the falling edge (%Qr.m.c.16 to %Qr.m.c.31).

Optimization of Performance

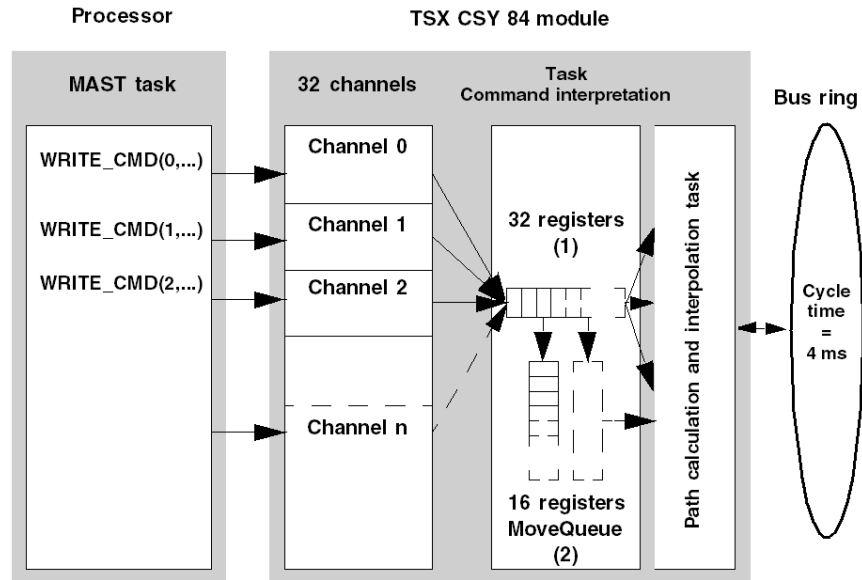
The implicit exchanges (%I, %Q) are installed for all channels, on each cycle of the MAST task. In order to optimize performance, it is recommended when possible to privilege the implicit exchanges (%Q), in relation to the explicit exchanges (WRITE_CMD).

Important

The TSX CSY 84 module must be configured in MAST task.

Graphical Representation of Exchanges

Exchanges of a WRITE_CMD command are made according to the following principle:



(1) 32 registers (1 per channel)

(2) 16 registers of 32 commands (8 real, 4 imaginary, 4 co-ordinated axis groups)

"SERCOS®" Function

At a Glance

The SERCOS® function is carried out by channel 0 of the TSX CSY 84 and TSX CSY 164 modules. It controls the bus ring. On power up or during bus closure, the axis control module performs auto learning of the bus.

Channel 0

Channel 0 is configured with default parameters. The parameters are modified either:

- by using the configuration or adjustment screens,
- or via the project using explicit exchanges.

Accessible Services

Channel 0 enables the following services to be used (see *Explicit Exchange Language Objects Associated with the Application-Specific Function*, page 389):

- READ_PARAM,
- WRITE_PARAM,
- WRITE_CMD,
- SAVE_PARAM,
- RESTORE_PARAM,
- READ_STS.

"Real Axis" Function

At a Glance

The "Real Axis" function is carried out by channels 1 – 8 of the TSX CSY 84 module and by channels 1 – 16 of the TSX CSY 164 module. It can control the servodrives via the bus ring to create physical axes (up to 8 possible physical axes).

NOTE: The parameters of an axis control module and servodrives are run separately via Unity Pro and UniLink. So, if the user changes the value of a parameter used by the module and the servodrive, using the UniLink configurator, this will not be updated in the axis control module. After this type of operation, the user must therefore adjust the parameter in the module using Unity Pro.

For more information on managing parameters, (see *Parameter Management*, page 398).

NOTE: The servodrive corresponding to the real axis 1 should be located at address 1 of the SERCOS®.ring.

Accessible Services

Channels 1 to 8 enable the following services to be used (seer *Explicit Exchange Language Objects Associated with the Application-Specific Function*, page 389):

- READ_PARAM,
- WRITE_PARAM,
- WRITE_CMD,
- SAVE_PARAM,
- RESTORE_PARAM,
- READ_STS,
- TRF_RECIPE.

"Imaginary Axis" Function

At a Glance

The "Imaginary Axis" function is carried out by channels 9 to 12 of the TSX CSY 84 module and by channels 1 to 16 of the TSX CSY 164 module.

An imaginary axis is not a physical axis but it can be used to co-ordinate the movement of several physical axes or it can also be the master axis of a slave axis group. It may also be used to test a system without moving the axis.

Accessible Services

Channels 9 to 12 (of the TSX CSY 84 module) and channels 1 to 16 of the TSX CSY 164 module give access to use of the following services: (see *Explicit Exchange Language Objects Associated with the Application-Specific Function*, page 389):

- READ_PARAM,
- WRITE_PARAM,
- WRITE_CMD,
- SAVE_PARAM,
- RESTORE_PARAM,
- READ_STS.

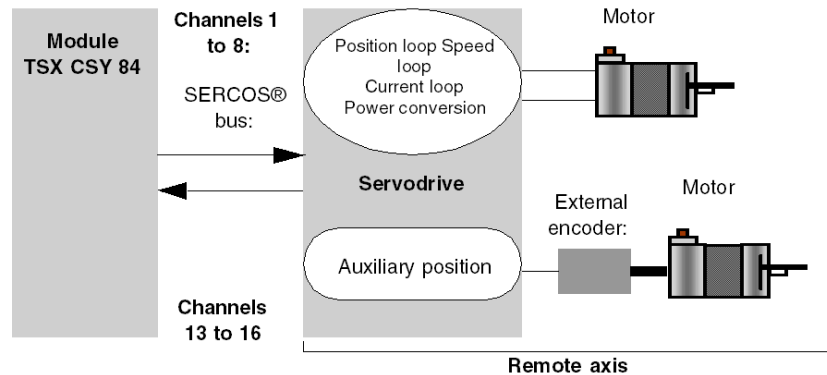
"Remote Axis" Function

At a Glance

The "Remote Axis" function is carried out by channels 13 to 16 of the TSX CSY 84 module and by channels 1 to 16 of the TSX CSY 164 module.

The remote axis can be used to integrate position data delivered by a remote system.

The following diagram illustrates the difference between a real axis and a remote axis, as seen by the servodrive with the TSX CSY 84 module:



Remote Axis

The remote axis function is accessible, according to its configuration, via:

- the `REMOTE_POSITION (%QDr.m.c.0)` language object (implicit exchange),
- a servodrive's auxiliary input.

NOTE: On the **Remote Axis** channels supported by a real axis, fault flagging is implemented on the corresponding **real axis** channel.

Accessible Services

Channels 13 to 16 (of the TSX CSY 84 module) and channels 1 to 16 of the TSX CSY 164 module give access to use of the following services: (see *Explicit Exchange Language Objects Associated with the Application-Specific Function*, page 389):

- READ_PARAM,
- WRITE_PARAM,
- WRITE_CMD,
- SAVE_PARAM,
- RESTORE_PARAM,
- READ_STS.

Concept of the Group

At a Glance

The TSX CSY 84/164 modules use the concept of the group to command axes. Any action on a group of axes entails an identical action on the member axes in the group.

Movement Group

All MovementGroups have their own MovementControl command interfaces.

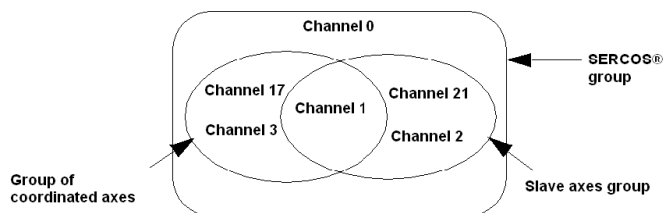
- the MovementControl interface for an individual axis always acts on that axis.
- the MovementControl interface for the SERCOS® ring acts on all of the MovementGroup axes on the string.
- the MovementControl interface for Coordinated Groups or Slave Groups are only in effect if the MovementGroup is controlled (Get).

The position control of control bits in these registers causes the corresponding action to occur for a set of axes, rather than a single axis.

The MovementGroup scans the MovementControl interface every SERCOS® clock cycle. Consequent changes to a control bit must be separated by at least one SERCOS® clock cycle. In general, the MovementStatus register for a MovementGroup should be examined to determine whether a control bit's action was successful.

Configuration Example on the TSX CSY 84 Module

Suppose that the following groups have been defined at configuration



- Channel 0: SERCOS® group
- Channel 21: group of slave axes 21, comprising real axes 1 and 2,
- Channel 17: group of coordinated axes 17, comprising real axes 1 and 3.

Any action on the Network channel (channel 0) affects groups 17 and 21, and real axes 1, 2 and 3.

Any action on the group of slave axes (channel 21) affects real axes 1 and 2.

Any action on the group of coordinated axes (channel 17) affects real axes 1 and 3.

NOTE: In order to use a real axis, this rule requires that the allow bits of the SERCOS® group (channel 0) are set to 1.

"Coordinated Axis Group" Function

At a Glance

The "Coordinated Axis Group" function is performed by channels 17 to 20 of the TSX CSY 84 and TSX CSY 164 modules.

A coordinated axis group is a set of physical axes (maximum of 8) moving in coordination. One of the axes in the group is defined as the coordination master.

The position of each of the axes is defined by the motion control. The reference speed for the movement of the coordinated set is the speed of the coordination master. The acceleration and speed of other axes are calculated so that all the axes start and finish moving at the same time.

NOTE: All the axes used by a coordinated group must be referenced before the group is confirmed and before implementing coordinated motions

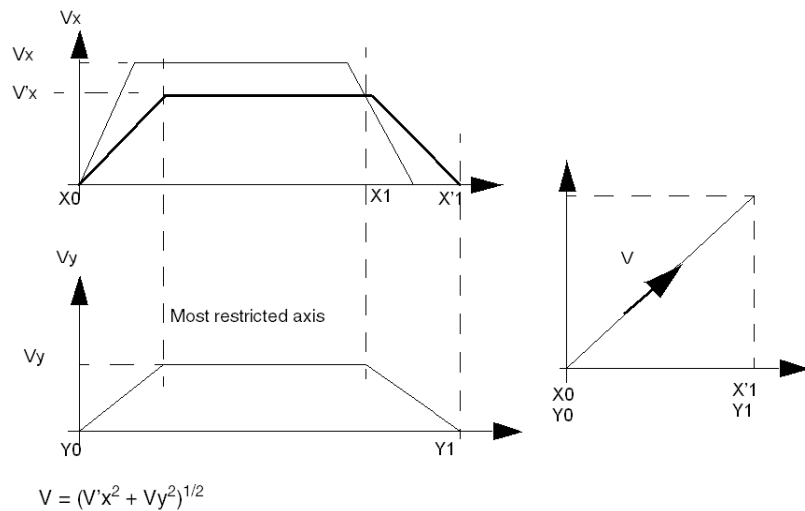
NOTE: When axes are configured in the slave groups, it is recommended to associate one group with a given axis.

NOTE: An axis member fault must be acknowledged at group level (CLEAR_FLT).

Reference Speed

The reference speed for a coordinated axis group motion must be calculated in relation to the most restricted axis. The algorithm reduces the speed of the most active axes.

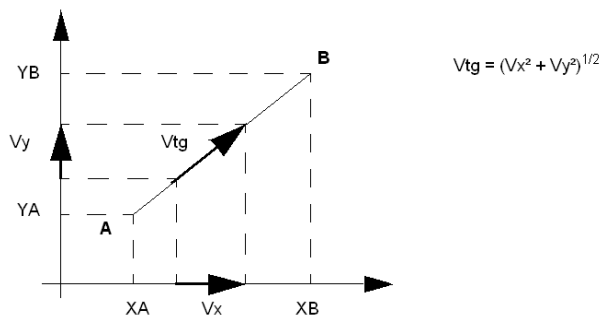
For example, in the case of a 2 axis system



Tangential Speed

Tangential speed is calculated in relation to the projection on the X axis of the initial speed as defined by the instruction `WRITE_CMD (MOVE)`. If there are no mechanical restraints, enter maximum speed for the other speeds.

Example in the case of a 2 axis system



The moving part moves from point A (XA, YA) to point B (XB, YB) at a tangential speed Vtg which projects on X and Y respectively as Vx and Vy.

The interpolator calculates speed Vy then Vtg using value Vx (the most restricted axis) in the `WRITE_CMD (MOVE)` instruction.

Accessible Services

Channels 17 to 20 enable the following services to be used (see *Explicit Exchange Language Objects Associated with the Application-Specific Function*, page 389):

- `WRITE_CMD`,
- `READ_STS`,
- `MOD_PARAM` (see page 482), only with the TSX CSY 164 module. This function is not available with the TSX CSY 84 module.

"Slave Axis Group" Function

At a Glance

The "slave axis group" function is performed by channels 21 to 24 of the TSX CSY 84/164 module.

A group of slave axes consists of a master axis and slave axes (max. 6), which follow the movement of the master axis.

There are 2 ways of following a master axis:

- In Ratio mode: the slave axes follow the master axis according to a relationship that has been defined in configuration and known as the Slave relationship (the position of the slave is the same as the master: setpoint or actual, multiplied by the ratio),
- In Cam mode: the slave axes follow the master axis according to a cam profile (a points table that gives the position of the slave as a function of the master's position: setpoint or current value).

The master axis may be a real or imaginary axis or an external setpoint. The slave axes are real or imaginary axes.

NOTE: When axes are configured in the slave groups, it is recommended to associate one group with a given axis.

NOTE: An axis member fault must be acknowledged at the level of the group (CLEAR_FLT).

CAUTION

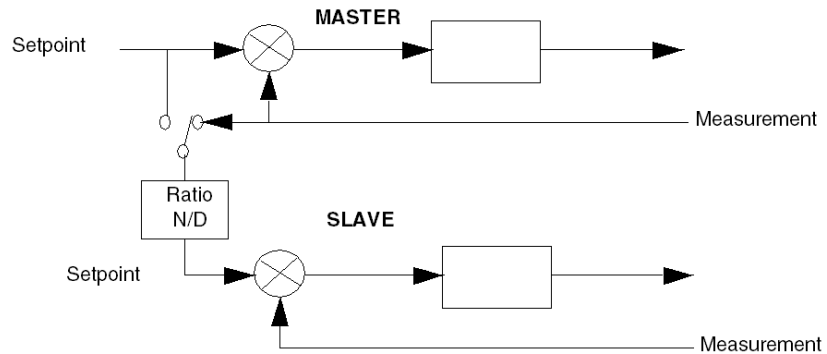
UNEXPECTED APPLICATION BEHAVIOR

When using master-slave configuration, ensure that the slaves capabilities are compatible with master speed and acceleration law characteristics.

Failure to follow these instructions can result in injury or equipment damage.

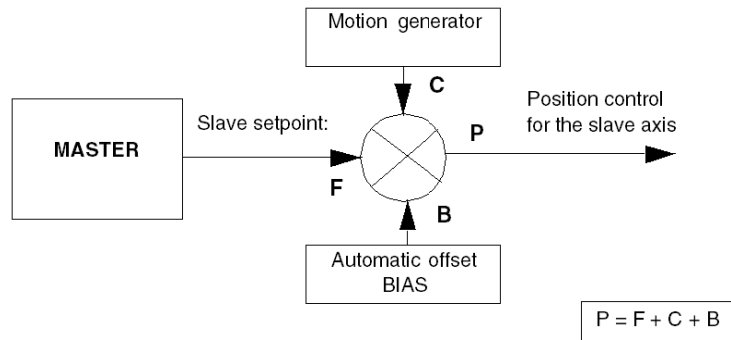
Ratio Mode

The ratio is defined by entering a numerator and a denominator. The ratio mode (gears) is able to manage switches (current value, setpoint), offsets (Bias), start conditions and follower stop



Following in Stop and Bias

These functionalities are able to add a movement or an automatic offset to the slave setpoint. This enables a movement with a subsequent activation to be anticipated or to continue with a movement after the subsequent deactivation of following.



Accessible Services

Channels 21 to 24 enable the following services to be used (see *Explicit Exchange Language Objects Associated with the Application-Specific Function*, page 389):

- READ_PARAM,
- WRITE_PARAM,
- WRITE_CMD (except Move),
- SAVE_PARAM,
- RESTORE_PARAM,
- READ_STS.
- MOD_PARAM (see page 389), only with the TSX CSY 164 module. This function is not available with the TSX CSY 84 module.

"Cam Profile" Function

At a Glance

The "Cam Profile" function is performed by channels 25 to 31 of the TSX CSY 84 module. The cam profiles are used by the slave axis groups to follow the master axis according to a points table known as the cam profile.

A cam profile is a 2 column points table:

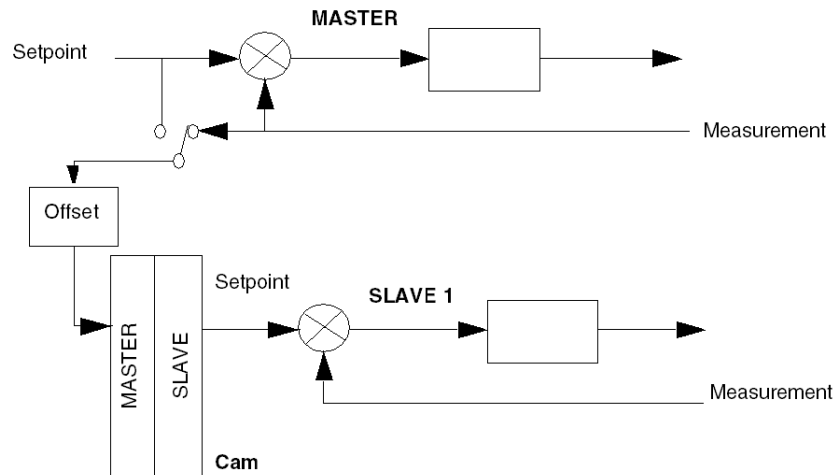
- a column that defines the position of the master in a slave axis group (usually an imaginary axis),
- a column that defines the position of the slave, associated with the master's position.

The interpolation between 2 consecutive points enables the positions not given by the table to be deduced. The interpolation may be linear or cubic.

The positions of the master must be increasing. For example, they must not go from 360 to 0 degrees. This would correspond to decreasing positions.

Cam Profile Diagram

The diagram representation of a slave axis controlled by a cam profile is as follows:

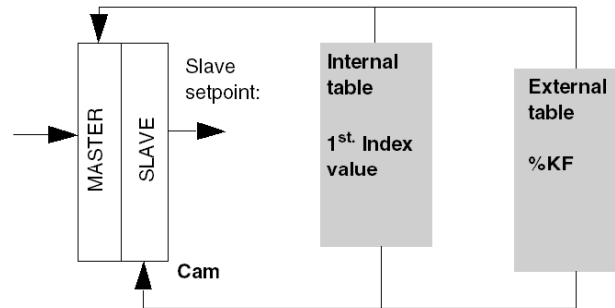


Cam

The cam can be implemented:

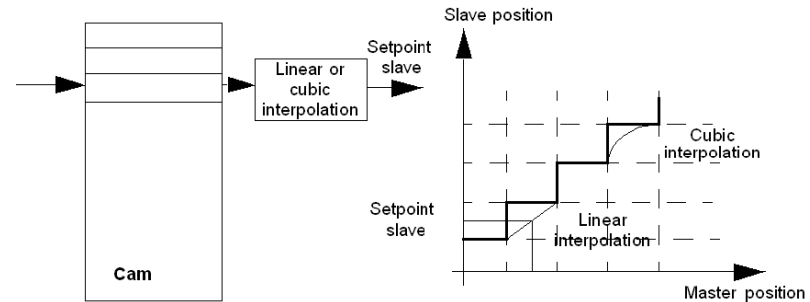
- either by an internal table in the axis control module, as defined by the initial table value and a fixed increment.
- or by an external table to the axis control module (contained in the PLC), defined as an address %KF.

Illustration



Interpolation

If the cam entry corresponds to a value between 2 consecutive points, the slave setpoint is calculated by linear or cubic interpolation (defined at configuration)



Accessible Services

Channels 25 to 31 enable the following services to be used (see *Interface Language*, page 396):

- WRITE_CMD,
- READ_STS,
- TRF_RECIPE.

Subject of this Chapter

This chapter describes the configuration screens of the TSX CSY 84 and SX CSY 85 modules and some of the functions they can perform: "SERCOS® Functions," "Real Axis", "Imaginary Axis", and more.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
IODDT Association	116
Module Configuration	117
Configuration of the Channels	121
Configuration of the SERCOS® Function (Channel 0)	122
Configuring an Individual Axis (Channels 1 to 12)	123
Configuration of an External Measurement Axis (Channels 13 to 16)	127
Configuration of a Coordinated Set (Channels 17 to 20)	129
Configuration of a Slave Axis Group (Channels 21 to 24)	130
Configuration of a Cam Profile (Channels 25 to 31)	133

IODDT Association

Reminder

The Variable editor is used to create and associate IODDTs for the TSX CSY 84 module. Please refer to the documentation describing IODDTs and their association (*see Unity Pro, Operating Modes*) with the various channels in application-specific modules.

Module Configuration

Introduction

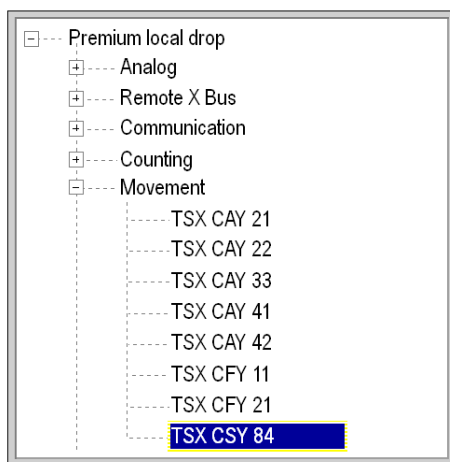
In Configuration mode, it is possible to define the operating characteristics for each TSX CSY 84 module path.

This mode is accessible in offline mode.

The TSX CSY 84 module occupies two positions in the rack. For this module to be proposed, it is therefore necessary to select the position the furthest to the right of the two slots allocated to receive the module.

Adding a Module

The following dialog box is used to add the TSX CSY 84 module to the configuration:

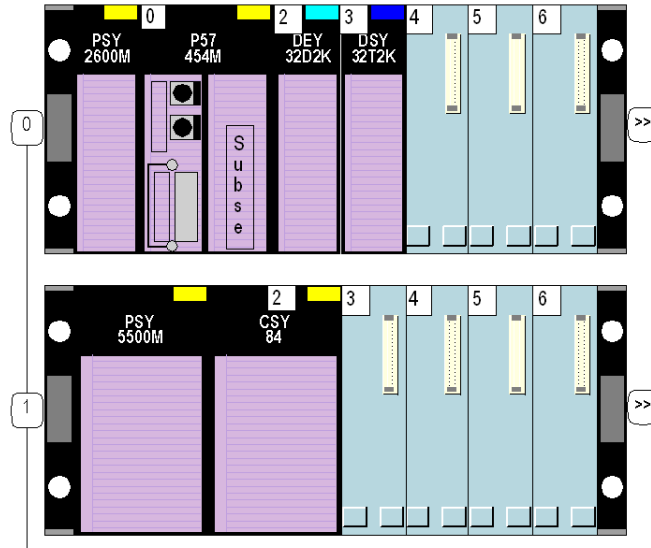


To do this:

- select **Movement**,
- select the reference of the module (TSX CSY 84),
- then drag and drop to the chosen slot.

Declaring the Module

To indicate that the module is declared in the configuration of the rack, the module appears in the rack together with its reference.



Deleting a Module

To delete a module from its position:

- click on the module to select it,
- press the <Del> key, which brings up a dialog box,
- confirm deletion of the module.

Application Specific Channels

The name "application specific channel" is given to all the channels of an intelligent module (counting module, axis command module, etc.). The 32 channels of the TSX CSY 84 module are application specific channels.

The number of application specific channels that a PLC has depends on its power and therefore its reference. To find out the properties of each processor, refer to the documentation for processors.

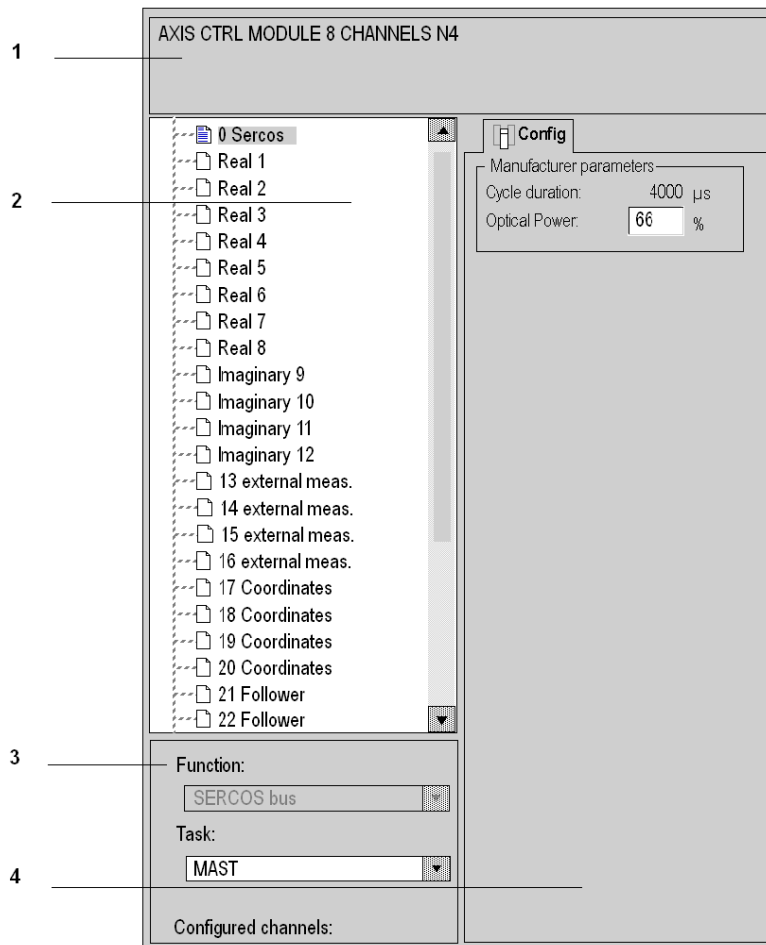
Accessing the Module Configuration Screen

To access the module configuration screen, double click on its graphical representation (CSY 84) or:

- select the module (by clicking on it),
- activate the **Open the module** command from the **Edit** drop-down menu.

Configuration Screen

The module configuration screen is as follows:



Description of the configuration screen

Field	Description
1	This band gives the catalog reference of the module and its geographical address in the PLC (rack number and position in the rack).
2	This field known as the Channel field is used to select the channel to be configured.
3	This field known as the General parameters field is used to configure the general parameters associated with the channel selected.
4	This field depends on the tab selected: in this case, it is the configuration zone for the channel selected in zone 2. Only the configuration tab is available in local mode.

Configuration of the Channels

Introduction

Configuring a channel involves defining the parameters for the application-specific function associated with this channel:

- Channel 0: SERCOS® function
- Channels 1 to 12: Independent axis (real or imaginary axis)
- Channels 13 to 16: Remote axis
- Channels 17 to 20: Coordinated axis group
- Channels 21 to 24: Slave axis group
- Channels 25 to 31: Cam profile

Configuration Check

While the parameter entries are being validated, a configuration check is performed. If there is non-coherency (e.g.: Max. Acceleration < Min. Acceleration), an error message displays the type of error. Furthermore, the parameters affected are displayed in red on the screen.

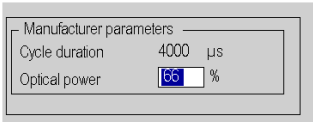
Configuration of the SERCOS® Function (Channel 0)

Introduction

The channel 0 that manages the SERCOS® bus is default-configured. The configuration screen allows you to input the optical power and specifies the bus parameter values.

Configuration Screen

The SERCOS® function configuration screen on TSX CSY 84 and TSX CSY 164 modules is as follows:



Parameter Description

The parameters for the SERCOS® ring are defined in the **Manufacturer parameters** field:

Parameters	Description
Cycle duration	Cycle time of the SERCOS® bus: 2 or 4 ms. By default 4 ms. The new value will be taken into account after switching to SERCOS® phase 0 (opening of the SERCOS® ring or SetCommandedPhase function) then switching to phase 4.
Optical power	Adjustment of optical power, necessary for the first segment (between the TSX CSY 84 or TSX CSY 164 module and the first servodrive). This power (in percentage terms) must be related to segment length (see adjustment chart). By default 66%.

Configuring an Individual Axis (Channels 1 to 12)

Introduction

An individual axis can be either a real axis (channels 1 to 8) or an imaginary axis (channels 9 to 12). The configuration of a real axis enables control of a physical axis (which uses a servodrive). In this case, a certain consistency must be ensured between the parameters entered in the TSX CSY 84 module configuration screen and those defined during servodrive configuration.

An imaginary axis can, for example, act as a master axis in a follower set. All the parameters of such an axis are defined in the configuration screens of the axis command module.

Configuration Screen

The Configuration screen for an individual axis appears below. There are 5 parameter entry zones: **Limits**, **Enabling position checking**, **Units**, **Scale factor**, and **Movement**.

The screenshot shows a configuration window with the following fields and options:

- Limits:**
 - ☒ Position checking
 - Max. position: 3.000000e+002
 - Min. position: -3.000000e+002
 - Max. speed: 1.000000e+005
 - Max. Acceleration: 1.000000e+005
 - Max. Deceleration: 1.000000e+004
 - Enable position band: ☐ Enable
 - Tolerance: 0.000000e+000
- Units:**
 - Type: Linear
 - Position: mm
 - Speed: mm/s
 - Acceleration: mm/s²
- Scale factor:**
 - Numerator: 1.000000e+000
 - Denominator: 1.000000e+000
- Movement:**
 - ☐ Module
 - Max. modulo: 2.000000e+001
 - Min. modulo: -2.000000e+001
 - In-position band: 7.000000e+000
 - Acceleration: 1.000000e+000
 - Deceleration: 1.000000e+000
 - Acceleration type: Trapezoid 125%

Limits Zone Parameters

Description

Parameters	Description
Position checking	For a limited machine, this checkbox can be used to enable the position limit check. The axis position is compared with the position limits defined during configuration. When the axis reaches one of its limits, its movement is stopped and an error is generated. For an infinite axis, this box should not be checked.
Max. position	Maximum position limit. This value is entered as a floating point.
Min. position	Minimum position limit. This value is entered as a floating point.
Max. speed	Maximum permissible speed. This value is independent of the value defined in the servodrive (real axis). This value is entered as a floating point. Setting the maximum speed to 0 disables speed monitoring.

Parameters	Description
Max. acceleration	Maximum permissible acceleration. This value is independent of the value defined in the servodrive (real axis). This value is entered as a floating point.
Max. deceleration	Maximum permissible deceleration. This value is independent of the value defined in the servodrive (real axis). This value is entered as a floating point.

Enabling Position Checking Zone Parameters

Description

Parameters	Description
Enable	This checkbox can be used to enable position checking. When the axis is disabled: ● If its movement is less than the tolerance, the axis returns to its last position when it is reactivated. ● If its movement is greater than the tolerance, the axis remains in its new position when it is reactivated.
Tolerance	Value of the monitoring window. This value is entered as a floating point.

Units Zone Parameters

Description

Parameters	Description
Type	Type of physical units used to express the position, speed and acceleration readings: Angular, Linear, Linear English or Encoder points.
Position	Position unit ● Angular: mrad, rad, deg, arcmin, revs ● Linear: μm, mm, cm, m ● Linear English: in, ft, yd, mile ● Encoder points: points.
Speed	Speed unit. ● Angular: mrad/s, rad/s, rad/min, deg/s, deg/min, arcmin/s, revs/s, revs/min ● Linear: $\mu\text{m/s}$, mm/s, mm/min, cm/s, cm/min, m/s, m/min ● Linear English: in/s, in/min, ft/s, ft/min, yd/min, mile/s ● Encoder points: points/ms, points/s, points/min
Acceleration	Acceleration unit. ● Angular: mrad/s^2, rad/s^2, deg/s^2, arcmin/s^2, revs/s^2, revs/min/s ● Linear: $\mu\text{m/s}^2$, mm/s^2, cm/s^2, m/s^2, m/min^2, g's ● Linear English: in/s^2, ft/s^2, yd/min^2, mil/s^2 ● Encoder points: points/ms^2, points/s^2

Scale Factor Zone Parameters

Description

Parameter	Description
Numerator	Scale factor numerator. This value is entered as a floating point.
Denominator	Scale factor denominator. This value is entered as a floating point.

Movement Zone Parameters

Description

Parameter	Description
Modulo	For an infinite axis, this checkbox can be used to activate the modulo function.
Max. modulo	Upper modulo limit. This value is entered as a floating point.
Min. modulo	Lower modulo limit. This value is entered as a floating point.
In-position band	Value of the in-position band. This value is entered as a floating point.
Acceleration	Acceleration value defined for a movement. This value is entered as a floating point.
Deceleration	Deceleration value defined for a movement. This value is entered as a floating point.
Acceleration type	Acceleration type: Rectangle 100%, Trapezoid 125%, Trapezoid 150%, Trapezoid 175% or Triangle 200%.

Rescaling

For a servodrive whose position is defined in angular units (degrees) the module performs a rescaling operation, according to its reference, which is measured in revolutions, and its speed, which is measured in revolutions/second.

For example, if the axis is configured as angular axis with a scale factor of 1/1:

- Position unit in revolutions and speed unit in rev/s: An incremental movement of position 1 and speed 1 will execute 1 revolution in 1 second.
- Position unit in degrees and speed unit in rev/min: An incremental movement of position 360 and speed 60 will execute 1 revolution in 1 second.

Changing the Unit Type

When the type of unit is changed in relation to the servodrive, which remains in revolutions, the reference unit of the module is mm (linear type) and inches (linear English type). The transformation achieved by the module is equal to:

- 1 revolution for the servodrive = 1 mm for the module (linear type)
- 1 revolution for the servodrive = 1 inch for the module (linear English type)

For example, if the axis is configured as linear with a scale factor of 1/1:

- Position unit in mm and speed unit in mm/s: An incremental movement of position 1 and speed 1 will execute 1 revolution in 1 second.
- Position unit in mm and speed unit in mm/s: An incremental movement of position 1000 and speed 1000 will execute 1 revolution in 1 second (or 1000 mm in 1 second).

Using the Scale Factor

Consider a project where the axis moves a belt; for instance 1 axis revolution moves the belt 100 mm. If you wish to express the position in mm:

- The position unit will be configured in mm, the speed in mm/s, the scale factor numerator will be equal to 100, and the denominator will remain at 1.
- An incremental movement of position 1000 and speed 1000 will move the belt 1 m (or 10 axis revolutions) in 1 second (speed 1000 mm/s).

Configuration of an External Measurement Axis (Channels 13 to 16)

Introduction

An external measurement channel is used to return external position information to the module. Configuring an external measurement involves configuring a real axis or an imaginary axis in which only the position information is valid.

Configuration Screen

The configuration screen of an external measurement is as follows. There are 3 parameter entry zones: **Scale factor**, **Modulo** and **Units** as well as a field for choosing the address.

Address

This field allows you to define the source of the position information:

- **Virtual:** the position is written by the project program in the output register %QDr.m.c.0,
- **1 to 8:** the position is read on the auxiliary position input of the real axis defined in the choice of the Address field (1 to 8).

Scale Factor Zone Parameters

Description

Parameter	Description
Numerator	Scale factor numerator. This value is entered as a floating point.
Denominator	Scale factor denominator. This value is entered as a floating point.

Modulo Zone Parameters

Description

Parameter	Description
Active	In the case of an infinite axis, this check box is used to activate the modulo crossing authorization.
Max modulo	Upper modulo limit. This value is entered as a floating point.
Min. modulo	Lower modulo limit. This value is entered as a floating point.

Units Zone Parameters

Description

Parameters	Description
Position	Position unit: mm, mm, cm, m, in, ft, yd, mil, mrad, rad, deg, arcmin, turns, pulses.

Internal Unit

The internal unit is mm for the linear (metric) type, inch for the English linear type and turn for the angular type.

Choosing a unit in degrees means that for a scale factor of 1/1, an increment of 1 pulse of the encoder changes the position by 360 degrees. For a unit in cm and a scale factor of 1/1, an increment of 1 pulse of the encoder changes the position by 1/10 mm.

Configuration of a Coordinated Set (Channels 17 to 20)

Introduction

A coordinated set is a set of real or imaginary axes whose movements are intercoordinated. To configure a coordinated set, it is necessary to have previously configured the independent axes belonging to the set.

Configuration Screen

The configuration screen of a coordinated set is as follows. This allows you to define the list of axes that will be intercoordinated (maximum of 8 axes).

Axis								Special function	
X	Y	Z	A	B	C	D	E	☐ Confirmation	Function number
1	2	5	N	N	N	N	N		0
								Address table %K	0

Axis X to Axis E

These 8 fields allow you to choose the axes that will be intercoordinated (axes 1 to 12). The value N indicates that the axis does not belong to the coordination. The main axis of the coordination is the axis defined in the Axis X field.

NOTE: The axes must be configured in order. For example, it is prohibited for Axis X to have the value N and Axis Y the value 1.

The acceleration profile of the axes of a coordinated set is, by default, rectangular. Although each axis taken separately can have a defined profile, it is nevertheless possible to change the profile of the set using the SetAccelType command.

Parameters of the Special Function Zone

Description

Parameter	Description
Confirmation	This check box is used to enable the activation of special functions.
Function number	This field is used to enter the number of the special function to be activated.
Address table %K	This field is used to enter the starting address of the table containing the parameters associated with the special function.

Configuration of a Slave Axis Group (Channels 21 to 24)

Introduction

A follower set is a set of axes comprising slave axes (maximum of 6) that track the movements of a master axis.

The master axis may be a real axis, an imaginary axis or an external setpoint. The slave axes are real or imaginary axes.

Configuration Screen

The configuration screen of a slave axis group appears below. It features 7 zones, which are used to configure the master axis and the 6 possible slave axes.

Master

This field is used to define the number of the master axis (1 to 16). The value N indicates that the master axis has not been chosen.

Slave Zones

The 6 slave zones (1 to 6) are identical. They are only active when the master number is defined.

Slave Zone Parameters

Description

Parameter	Description
Slave 1 (to 6)	Used to set the number of the slave axis (axes 1 to 12).
Measurement	When this button is checked, the slave axis follows the measured position of the master axis. This button operates alternately with the Setpoint button.
Setpoint	When this button is checked, the slave axis follows the position of the master axis setpoint. This button operates alternately with the Measurement button.
Gearing	When the button is checked, the slave axis follows the master axis in Ratio mode; that is to say following a ratio determined by the Ratio field. This button operates alternately with the Cam button.
Cam	When this button is checked, the slave axis follows the master axis in Cam mode; that is to say following the cam profile whose number is selected in the No. field. This button operates alternately with the Gearing button.
Ratio	In Ratio mode, these 2 fields are used to input the numerator and the denominator, which define the ratio between the master axis and the slave. These values are entered as floating points.
Start	Used to select the start condition: ● Immediate ● When the master position, increased by the offset value, reaches the threshold value defined in the Trigger field (CW travel) ● When the master position plus the offset value, reaches the threshold value defined in the Trigger field (CCW direction) ● When the master position plus the offset value is greater than or equal to the threshold value defined in the Trigger field ● When the master position plus the offset value is less than or equal to the threshold value defined in the Trigger field
No.	In Cam mode, this field is used to select the cam profile number (between 25 and 31).
Offset	In Cam mode, this field is used to input an offset value, which will be added to the master position, in order to define the position of the slave. The resulting position of the slave will be given by the index in the cam profile master table. This index is equal to the current position of the master + offset (for example, a cam profile defined from 0 to 1000 for the master coordinates). To start following a master position equal to 100000, the offset value should be equal to -100000.0. This offset is used, for example, to define a sine and cosine function from the same cam profile. This value is entered as a floating point.

Parameter	Description
Bias remains (residual)	When this box is checked: • A dynamic offset is added automatically to the master position in order to define the position of the slave. • The additional slave movements are not halted when the link with the master is removed.
Trigger	When the start condition is dependent on the position of the master axis in relation to a threshold, this field is used to input the threshold value. This value is entered as a floating point. Activation will take place when the current position of the master + offset > (or <, >, <) the value of the threshold (Trigger).
Stop on follow-on	When this box is checked, confirming the link between the master and the slave will stop any possible additional movement of the slave axis, following a deceleration profile determined automatically.
Stop master/flt	When this box is checked, the master stops when there is a contour discrepancy (fault) between the master and the slave.

Position of a Slave Axis

When an axis is a slave, its position depends solely on that of the master axis, which it is following. The configuration parameters of the axis (position limits, maximal speed, maximal acceleration, etc.) are ignored. To safeguard the project, configure these (security) parameters in the servodrive.

Scale Factor

In a group of slave axes, the scale factor is unrelated to the units used for the axes, when these are of the same type (Linear, Angular, etc.). For example, if the master axis is configured in m and the slave axis in cm (different units but the same type: Linear), and if a scale factor of 1/1 is used, this means that for every 1 mm traveled by the master, the slave will also move 1 mm.

If the master and slave units are of different types, the units must be converted into the reference unit of the type of unit (mm for the linear type, inch for the linear English type, rev for the angular type). For example, if the master is configured in m and the slave in revs (units of different types: Linear and Angular) the slave is to complete 1 revolution when the master moves 1 m, the scale factor will have to be defined as follows:

- 1 m = **1000** mm (in reference units of the linear type)
- 1 rev = **1** rev (in reference units of the angular type)

Therefore, the scale factor = **1000/1** (when the master travels 1000 mm, the slave completes 1 revolution).

Configuration of a Cam Profile (Channels 25 to 31)

Introduction

A cam profile is used to define, using a pulses table, the position of a slave axis in relation to the position of the master axis.

Module TSX CSY 84/164

On TSX CSY 84 and TSX CSY 164 modules, cam profile configuration is implemented on channels 25 to 31.

Configuration Screen

The configuration screen for a cam profile has 3 zones allowing you to define the master and slave points table.

The screenshot shows a configuration window with the following fields and controls:

- Linear/Cubic:** Radio buttons for selecting the interpolation method. 'Linear' is selected.
- No. of pulses:** A text input field containing the value '0'.
- Master increment:**
 - Unit:** A dropdown menu showing 'µm'.
 - Fixed/Variable:** Radio buttons. 'Fixed' is selected.
 - Start value:** A text input field containing '0.000000e+000'.
 - Increment:** A text input field containing '0.000000e+000'.
- Slave increment:**
 - Unit:** A dropdown menu showing 'µm'.
 - Fixed/Variable:** Radio buttons. 'Variable' is selected.
 - Address table of %KF:** A text input field containing '0'.

Table Parameters

Description

Parameters	Description
Linear	When this button is selected, interpolation between 2 consecutive cam profile points is linear. This button operates alternately with the Cubic button.
Cubic	When this button is selected, interpolation between 2 consecutive cam profile points is cubic. This button operates alternately with the Linear button.
No. of points	This field can be used to enter the number of points used to define the cam profile.

Parameters of the Master Increment Zone

Description

Parameters	Description
Unit	Used to define the unit used for the master increment. The unit selected can be a sub unit of the unit defined for the axes (for example, cm for the axes and mm for the increment).
Fixed	When this button is selected, the increment between 2 consecutive cam profile points is always the same. This button operates alternately with the Variable button.
Start value	For a fixed increment, this field can be used to enter the start value for the cam profile. This value is entered as a floating point.
Increment	For a fixed increment, this field can be used to define the value of the increment. This value is entered as a floating point.
Variable	When this button is selected, the increment between 2 consecutive cam profile points is variable. The value of the points is defined by a table of constant words %KF, whose length is equal to the number of points.
Address table of %KF	For a variable increment, this field can be used to enter the start address for the table of points for the master.

NOTE: A cam profile is always circular. It is necessary to ensure that the first and last values in the slave table are equal.

In the case of a circular movement, the table will fully describe the modulo and an additional pulse (modulo + 1) will be added, the slave value of which will be the first value given in the table.

For example, for a modulo of 360°, the values are 0 to 359, and the table is as follows: Table (master, slave): (0, **x0**); (1, x1); (2, x2); ...; (359, x359); (360, **x0**)

In the case of a linear movement, and if the last value in the table for the slave is not equal to the first value, additional points are added (for example, the last point will be repeated several times with the slave values gradually approaching the first value in the table).

Parameters of the Slave Increment Zone

Description

Parameters	Description
Unit	Used to define the unit, which is used for the slave increment. The unit selected can be a sub unit of the unit defined for the axes (for example, cm for the axes and mm for the increment).
Fixed	When this button is selected, the increment between 2 consecutive cam profile points is always the same. This button operates alternately with the Variable button.
Start value	For a fixed increment, this field can be used to enter the start value for the cam profile. This value is entered as a floating point.
Increment	For a fixed increment, this field can be used to define the value of the increment. This value is entered as a floating point.
Variable	When this button is selected, the increment between 2 consecutive cam profile points is variable. The value of the points is defined by a table of constant words %KF, whose length is equal to the number of points.
Address table of %KF	For a variable increment, this field can be used to enter the start address of the table of points for the slave.

Programming the TSX CSY 84 and TSX CSY 164 modules

12

Aim of this chapter

This chapter describes the different functions associated with axis motion.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
12.1	Motion Status Bits	138
12.2	Movement Control Functions	172
12.3	Movement Functions	188
12.4	Motion Function Following Deactivation of the Sequence	199
12.5	Functions of actual speed/position	209
12.6	Following Functions	216

12.1 Motion Status Bits

Subject of this Section

This section describes the motion status bits which can be accessed via implicit exchange belonging to the channel IODDT (bits %Ir.m.c.0 to %Ir.m.c.31) or by the GetStatus function.

What Is in This Section?

This section contains the following topics:

Topic	Page
RAMPING Bit (%Ir.m.c.0)	140
STEADY Bit (%Ir.m.c.1)	141
STOPPING Bit (%Ir.m.c.2)	142
PROFILE_END Bit (%Ir.m.c.3)	143
IN_POSITION Bit (%Ir.m.c.4)	144
AXIS_HOMING Bit (%Ir.m.c.5)	145
AXIS_HOMED Bit (%Ir.m.c.6)	146
AXIS_NOT_FOLLOWING bit (%Ir.m.c.7)	147
HOLDING Bit (%Ir.m.c.8)	148
RESUMING Bit (%Ir.m.c.9)	149
DRIVE_ENABLED Bit (%Ir.m.c.10)	150
DRIVE_DIAG Bit (%Ir.m.c.11)	151
DRIVE_WARNING Bit (%Ir.m.c.12)	152
DRIVE_FAULT Bit (%Ir.m.c.13)	153
DRIVE_DISABLED bit (%Ir.m.c.14)	154
AXIS_SUMMARY_FLT Bit (%Ir.m.c.15)	155
AXIS_COMM_OK Bit (%Ir.m.c.16)	156
AXIS_IS_LINKED Bit (%Ir.m.c.17)	157
AXIS_IN_CMD Bit (%Ir.m.c.18)	158
AXIS_AT_TARGET Bit (%Ir.m.c.20)	159
AXIS_POS_LIMIT Bit (%Ir.m.c.21)	160
AXIS_NEG_LIMIT Bit (%Ir.m.c.22)	161
AXIS_WARNING Bit (%Ir.m.c.23)	162
BIAS_REMAIN Bit (%Ir.m.c.24)	163

Topic	Page
AXIS_MANUAL_MODE Bit (%Ir.m.c.25)	164
DRIVE_REALTIME_BIT1 Bit (%Ir.m.c.26)	165
DRIVE_REALTIME_BIT2 Bit (%Ir.m.c.27)	166
AXIS_HOLD Bit (%Ir.m.c.28)	167
AXIS_HALT Bit (%Ir.m.c.29)	168
AXIS_FASTSTOP Bit (%Ir.m.c.30)	169
AXIS_READY Bit (%Ir.m.c.31)	170
CONF_OK Bit (%Ir.m.c.32)	171

RAMPING Bit (%Ir.m.c.0)

Description

This bit (status 1) indicates that the ordered movement profile is either acceleration or deceleration.

Applied to Real or Imaginary Axis Functions

Acceleration or deceleration profile type.

Applied to a Group of Co-ordinated Axes

Acceleration or deceleration profile type.

The AXIS_IS_LINKED member axes also have their RAMPING bits set to 1.

Applied to a Group of Slave Axes

This bit is set to 1 if one or more RAMPING bits in the member axes are set to 1.

In ratio mode, a member axis RAMPING bit is set to 1 while the ratio of the slave axis is changing to a new value.

Applied to SERCOS® Function

This bit is set to 1 if the RAMPING bit in an axis is set to 1.

STEADY Bit (%Ir.m.c.1)

Description

This bit (status 1) indicates that the movement profile demands a uniform speed.

Applied to Real or Imaginary Axis Functions

Uniform speed profile type.

Applied to a Group of Co-ordinated Axes

Uniform speed profile type.

The AXIS_IS_LINKED member axes also have their STEADY bits set to 1.

Applied to a Group of Slave Axes

This bit is set to 1 if all the STEADY bits in the member axes are set to 1.

The STEADY bit remains clear if the move of a slave is only a copy of that of the master (no additional move).

Applied to SERCOS® Function

This bit is set to 1 if all the STEADY bits in the member axes are set to 1.

STOPPING Bit (%Ir.m.c.2)

Description

This bit (status 1) indicates that the ordered movement profile is deceleration type with movement stop.

It remains set to 1 until the IN_POSITION (%Ir.m.c.4) bit is set to 1.

The STOPPING bit is set to 0 when the PROFILE_END (%Ir.m.c.3) bit goes to status 1.

NOTE: The STOPPING bit is not set to status 1 in the case of deceleration due to the resetting of the ALLOW_MOVE or ALLOW_RESUME bits.

Applied to Real or Imaginary Axis Functions

Deceleration type movement profile with movement stop.

Applied to a Group of Co-ordinated Axes

Deceleration type movement profile with movement stop.

The AXIS_IS_LINKED member axes also have their STOPPING bits set to 1.

Applied to a Group of Slave Axes

This bit is set to 1 if one or more STOPPING bits in the member axes are set to 1.

The STOPPING bits do not check if the axis is stabilized, as an axis may be moving in response to a movement by its master.

Applied to SERCOS® Function

This bit is set to 1 if the STOPPING bit in an axis is set to 1.

PROFILE_END Bit (%Ir.m.c.3)

Description

This bit (status 1) indicates that the movement demanded has ended.

The axis may still be settling until IN_POSITION (%Ir.m.c.4) is set to 1 or STOPPING (%Ir.m.c.2) is cleared.

PROFILE_END is set to 1 when the axis is disabled or stop is effected.

Applied to Real or Imaginary Axis Functions

The movement profile has issued its last command to the drive. The axis may still be settling.

Applied to a Group of Co-ordinated Axes

The Co-ordinatedGroup's movement profile has issued its last command to all the member axis drives.

The AXIS_IS_LINKED member axes also have their PROFILE_END bits set to 1. The member axes may still be settling.

Applied to a Group of Slave Axes

This bit is set to 1 if all the member axis PROFILE_END bits are set to 1. This does not mean that the member axes are not moving. They may be following the movement of their master axis.

Applied to SERCOS® Function

This bit is set to 1 if all the PROFILE_END bits in the axes are set to 1.

IN_POSITION Bit (%Ir.m.c.4)

Description

This bit (status 1) indicates that axis position is in the in-position band and that the STOPPING (%Ir.m.c.2) bit is at 1.

IN_POSITION is set to 0 when the axis is disabled or stop is effected.

Applied to Real Axis Function

The axis is within the in-position band (STOPPING bit at 1).

Applied to Imaginary Axis Function

IN_POSITION is set as soon as PROFILE_END (%Ir.m.c.3) is set to 1.

Applied to a Group of Co-ordinated Axes

All the axes in the in-position band (STOPPING bits at 1) and the different member axes IN_POSITION bits (AXIS_IS_LINKED) are set to 1.

Applied to a Group of Slave Axes

This bit is set to 1:

- if all the member axis IN_POSITION bits are set to 1,
- the following is disabled.

NOTE: The member axis IN_POSITION bits are set to 0 when they follow the master, whether it moves or not.

Applied to SERCOS® Function

This bit is set to 1 if all the member axes IN_POSITION bits are set to 1.

AXIS_HOMING Bit (%Ir.m.c.5)

Description

This bit (status 1) indicates that the axis is performing a Reference Point
(*see page 182*) type function.

Applied to Real Axis Function

Reference Point type function.

Applied to Imaginary Axis Function

Not significant

Applied to the Group of Co-ordinated Axes, Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if one or more member axis AXIS_HOMING bits
(AXIS_IS_LINKED) are set to 1.

AXIS_HOMED Bit (%Ir.m.c.6)

Description

This bit (status 1) indicates that the Reference point function has been correctly executed.

Applied to Real Axis Function

The axis position reading is referenced from the source.

This bit is not set to 1 in the following cases:

- the reference point function has never been executed or failed to complete,
- the SERCOS® network breaks down or the drive detects a failure in the position feedback device,
- following the Unhome function.

NOTE: This bit is set to 1 following the ForcedHomed function.

Applied to Imaginary Axis Function

Not significant.

Applied to the Group of Co-ordinated Axes, Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if one or more member axis AXIS_HOMED bits (AXIS_IS_LINKED) are set to 1.

AXIS_NOT_FOLLOWING bit (%Ir.m.c.7)**Description**

This bit (status 1) indicates that the drive is ignoring the controller's motion profile while it performs a special operation (for example, reference point, FastStop, or Stop).

NOTE: This bit is not available on all SERCOS® drives.

Applied to Real Axis Function

Controller inhibited while drive performs a special operation.

Applied to Imaginary Axis Function

The bit is set to 1 if the axis is disabled and it is set to 0 if the axis is enabled.

Applied to the Group of Co-ordinated Axes, Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if one or more member axis AXIS_NOT_FOLLOWING bits are set to 1.

HOLDING Bit (%Ir.m.c.8)

Description

This bit (status 1) indicates that the axis is either:

- stopped at rest,
- or decelerating with movement stop.

Applied to Real or Imaginary Axis Functions

Axis is stopped or coming to rest.

Applied to a Group of Co-ordinated Axes

The axes in the group of co-ordinated axes are stopped or in the process of stopping.

The AXIS_IS_LINKED member axes also have their HOLDING bits set to 1.

Applied to the Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if one or more member axis HOLDING bits are set to 1.

RESUMING Bit (%Ir.m.c.9)

Description

This bit (status 1) indicates that the axis resumes after a pause. The axis is accelerating again to achieve the movement profile.

Applied to Real or Imaginary Axis Functions

Axis resuming after a pause.

Applied to a Group of Co-ordinated Axes

Axis resuming for a group of co-ordinated axes after a pause.

The AXIS_IS_LINKED member axes also have their RESUMING bits set to 1.

Applied to the Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if one or more member axis RESUMING bits are set to 1.

DRIVE_ENABLED Bit (%Ir.m.c.10)

Description

This bit (status 1) indicates that the drive is activated and the associated motor is powered.

NOTE: If the AXIS_READY (%Ir.m.c.31) bit is set to 1, it may be that a movement command is in progress on the axis.

Applied to Real Axis Function

Drive activated, motor powered.

Applied to Imaginary Axis Function

The axis is activated following a CONTROL_ENABLE bit rising edge.

Applied to the Group of Co-ordinated Axes, Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if all the member axis DRIVE_ENABLED bits (AXIS_IS_LINKED) are set to 1.

DRIVE_DIAG Bit (%Ir.m.c.11)

Description

This bit (status 1) indicates a modification to a class 3 SERCOS® diagnostic bit.

The IDN 0013 variable contains current information.

This bit is set to 0 when reading the IDN 0013 variable.

NOTE: The class 3 SERCOS® bits are not affected by the SERCOS® standard IDN 0013.

Applied to Real Axis Function

A class 3 diagnostic bit has changed (information).

Applied to Imaginary Axis Function

Not significant

Applied to the Group of Co-ordinated Axes, Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if one or more member axis DRIVE_DIAG bits (AXIS_IS_LINKED) are set to 1.

DRIVE_WARNING Bit (%Ir.m.c.12)

Description

This bit (status 1) indicates a modification to a class 2 SERCOS® diagnostic bit.
The IDN 0012 variable contains current information.
This bit is set to 0 when reading the IDN 0012 variable.

Applied to Real Axis Function

A class 2 diagnostic bit has changed (Warning).

Applied to Imaginary Axis Function

Not significant

Applied to the Group of Co-ordinated Axes, Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if one or more member axis DRIVE_WARNING bits (AXIS_IS_LINKED) are set to 1.

DRIVE_FAULT Bit (%Ir.m.c.13)

Description

This bit (status 1) indicates a modification to a class 1 SERCOS® diagnostic bit.

The IDN 0011 variable contains current information.

This bit is set to 0:

- when reading the IDN 0011 variable,
- from a CONTROL_CLEAR_FLT rising edge.

Applied to Real Axis Function

A class 1 diagnostic bit has changed (default).

Applied to Imaginary Axis Function

Not significant

Applied to the Group of Co-ordinated Axes, Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if one or more member axis DRIVE_FLT bits (AXIS_IS_LINKED) are set to 1.

DRIVE_DISABLED bit (%Ir.m.c.14)

Description

This bit (status 1) indicates that the drive is deactivated and the associated motor is cut off.

Applied to Real Axis Function

Drive deactivated, motor cut off.

Applied to Imaginary Axis Function

Axis is deactivated following the reset of the ALLOW_ENABLE bit.

Applied to the Group of Co-ordinated Axes, Group of Slave Axes or SERCOS® Ring Functions

This bit is set to 1 if all the member axis DRIVE_DISABLED bits (AXIS_IS_LINKED) are set to 1.

AXIS_SUMMARY_FLT Bit (%Ir.m.c.15)

Description

This bit (status 1) indicates either:

- a faulty servodrive (DRIVE_FLT),
- a SERCOS® communication fault (AXIS_COM_OK to 0),
- a faulty motion profile controller (AXIS_POS_LIMIT or AXIS_NEG_LIMIT).

It may also indicate slave group or coordinated group faults (a new function common to all TSX CSY 84 and TSX CSY 164 modules):

- confirmed groups with one faulty axis will be flagged as faulty (%Ir.m.c.15 to 1), with data supplied by the GET_MOTION_FAULT function, i.e.: group member fault (bit 02 (MF_MEMBER_FAULT) of the MOTION_FAULT word).

as well as fault and warning propagation (a new function common to all TSX CSY 84 and TSX CSY 164 modules):

- a real axis fault.

Information concerning the fault can be accessed with the help of the GetMotionFault (*see page 355*) function.

NOTE: The ALLOW_NOT_FLT bit with status = 0 causes a user fault.

Applied to Real or Imaginary Axis Functions

Motion Fault.

Applied to the Coordinated Axes Group, Slave Axes Group or SERCOS® Ring Functions

This bit is set to 1 if one or more member axis AXIS_SUMMARY_FLT bits (AXIS_IS_LINKED) are set to 1.

AXIS_COMM_OK Bit (%Ir.m.c.16)

Description

This bit (status 1) indicates that the drive has successfully powered up and the communications between the movement controller and the drive are active.

Applied to Real Axis Function

Cyclic communication between the movement controller and the SERCOS® drive has been established.

Applied to Imaginary Axis Function

Not significant

Applied to Group of Co-ordinated Axes or Group of Slave Axes Functions

This bit is set to 1 if all the member axis AXIS_COM_OK bits are set to 1.

Applied to SERCOS® Function

Meaning of bit at 1 in the different possible cases.

No axis configured	At least one axis is configured
SERCOS® cyclic communication established and possibility of configuring axes.	all the AXIS_IS_LINKED member axes also have their AXIS_COMM_OK bits set to 1;

AXIS_IS_LINKED Bit (%Ir.m.c.17)

Description

This bit (status 1) indicates that the axis is an active part of an axis group. The axis is responding to commands from a Co-ordinatedGroup or SlaveGroup.

NOTE: In the set of slave axes, the master is not a member axis.

Applied to Real and Imaginary Axis Functions

Active part of a Co-ordinatedGroup or SlaveGroup.

Applied to a Group of Co-ordinated Axes

Non significant bit

When the Co-ordinatedGroup is activated and active, the member axis AXIS_IS_LINKED bits are set to 1.

The member axes follow the movement profile of the Co-ordinatedGroup.

NOTE: it is not permitted to act directly on the member axes of the Co-ordinatedGroup.

Applied to a Group of Slave Axes

This bit is set to 1 if all the slave member axis bits (AXIS_IS_LINKED) are set to 1.

The member axes follow the movement profile of the SlaveGroup master axis as a FollowerRatio or CamProfile.

NOTE: Movement commands to the member axes are allowed and are added to the movement resulting from following the master axis.

Applied to SERCOS® Function

This bit is set to 1 if one or more axis AXIS_IS_LINKED bits are set to 1.

AXIS_IN_CMD Bit (%Ir.m.c.18)

Description

This bit (status 1) indicates that the axis is active and can receive orders.

Applied to Real and Imaginary Axis Functions

Active axis.

Applied to a Group of Co-ordinated Axes

This bit is set to 1 when:

- the Co-ordinatedGroup is activated,
- the member axes (AXIS_IS_LINKED) have not been released from its control.

Member axes do not respond to orders, which are sent to them.

The Co-ordinatedGroup AXIS_IN_CMD bit is set to 0 after the ALLOW_AQUIRE or ALLOW_ENABLE bits have been reset.

NOTE: Movement commands are given to the Co-ordinatedGroup, not to the member axes.

Applied to a Group of Slave Axes

This bit is set to 1 when:

- the Co-ordinatedGroup is activated,
- the member axes have not been released from its control.

The member axes also have their AXIS_IN_CMD bit set to 1, since the member axes will respond to movement commands directed to them, as well as follow the master axis if following is on. The two movement profiles are summed.

Applied to SERCOS® Function

Not significant.

AXIS_AT_TARGET Bit (%Ir.m.c.20)

Description

This bit (status 1) indicates that the position of the axis is:

- in the in-position band,
- has arrived at the end of the movement profile (PROFILE_END at 1).

AXIS_AT_TARGET is cleared when a new movement command is issued.

NOTE: Unlike IN_POSITION, AXIS_AT_TARGET is not changed when the axis is disabled, nor is it set after a halt is completed unless the halted position reaches the original movement profile's target position.

Applied to Real or Imaginary Axis Functions

Axis is in the in-position band with the PROFILE_END at 1.

Applied to the Group of Co-ordinated Axes or SERCOS® Ring Functions

This bit is set to 1 when all the member axis AXIS_AT_TARGET bits (AXIS_IS_LINKED) are set to 1.

Applied to a Group of Slave Axes

This bit is set to 1 if all the member axis AXIS_AT_TARGET bits are set to 1.

NOTE: The member axis AXIS_AT_TARGET bits are set to 0 when following the master.

AXIS_POS_LIMIT Bit (%Ir.m.c.21)

Description

This bit (status 1) indicates that the axis has reached or exceeded the upper limit (max. position).

The validity of this bit requires the DRIVE_ENABLED and AXIS_HOMED bits to be set to 1.

This bit is reset using the CONTROL_CLEAR_FLT bit rising edge in the current value if the fault has disappeared.

Applied to Real and Imaginary Axis Functions

Position of the commanded axis on or over the upper limit.

Applied to the Functions of a Group of Co-ordinated Axes, Group of Slave Axes and SERCOS® Ring

This bit is set to 1 if one member axis AXIS_POS_LIMIT bit (AXIS_IS_LINKED) is set to 1.

AXIS_NEG_LIMIT Bit (%Ir.m.c.22)

Description

This bit (status 1) indicates that the axis has reached or exceeded the lower limit (min. position).

The validity of this bit requires the DRIVE_ENABLED and AXIS_HOMED bits to be set to 1.

This bit is reset using the CONTROL_CLEAR_FLT bit rising edge in the current value if the fault has disappeared.

Applied to Real and Imaginary Axis Functions

Position of the commanded axis on or under the lower limit.

Applied to the Functions of a Group of Co-ordinated Axes, Group of Slaves or SERCOS® Ring

This bit is set to 1 if one member axis AXIS_NEG_LIMIT bit (AXIS_IS_LINKED) is set to 1.

AXIS_WARNING Bit (%Ir.m.c.23)

Description

This bit (status =1) indicates either:

- detection of an axis motion warning (*see page 353*).

or slave group or coordinated group faults (a new function common to all TSX CSY 84 and TSX CSY 164 modules):

- unconfirmed groups with one faulty axis will be flagged as Warning (%Ir.m.c.23 to 1), with data supplied by the GET_MOTION_WARNING function, i.e.: **group member fault** (bit 05 (MW_MEMBER_FAULT) of the MOTION_WARNING word).

The text displayed on the Unity Pro screen will be as follows: "Group member fault".

As well as fault and warning propagation (a new function common to all TSX CSY 84 and TSX CSY 164 modules):

- axes which have been stopped due to fault propagation (by the groups or by the warning function) will be flagged as Warning (%Ir.m.c.23 to 1), with data supplied by the GET_MOTION_WARNING function, i.e.: **linked axis fault** (bit 02 (MW_STOP_BY_SET) of the MOTION_WARNING word).

The text displayed on the Unity Pro screen will be as follows: "Group propagation stop".

- confirmed or unconfirmed groups with one axis on Warning will be flagged as Warning (%Ir.m.c.23 to 1), with data supplied by the **group member warning** (bit 04 (MW_MEMBER_WARNING) function of the MOTION_WARNING word).

The text displayed on the Unity Pro screen will be as follows: "Group member Warning".

The GetMotionWarning function displays the current motion warnings.

The AXIS_WARNING bit is set to 0 using a CONTROL_CLEAR_FLT rising edge.

Applied to Real and Imaginary Axis Functions

Detection of an axis motion warning.

Applied to a Group of Coordinated Axes

This bit is set to 1 if one member axis AXIS_WARNING bit (AXIS_IS_LINKED) is set to 1.

BIAS_REMAIN Bit (%Ir.m.c.24)

Description

This bit (status 1) indicates that an offset has been added to the command position.

NOTE: This bit is only concerned with the Real and Imaginary axis functions.

AXIS_MANUAL_MODE BIT (%Ir.m.c.25)

Description

This bit is used to check which mode is currently active by reading the value of the following discrete input:

- %Ir.m.c.25 = 0: auto mode (by default),
- %Ir.m.c.25 = 1: manual mode.

DRIVE_REALTIME_BIT1 Bit (%Ir.m.c.26)

Applied to Real Axis Function

This bit reflects the status of the drive's "SERCOS® real time" status bit 1.

Applied to Imaginary Axis Function

Not significant.

Applied to the Functions of a Group of Coordinated Axes, Group of SERCOS® Axes

This bit is set to 1 if one or more member axis DRIVE_REALTIME_BIT1 bits (AXIS_IS_LINKED) are set to 1.

DRIVE_REALTIME_BIT2 Bit (%Ir.m.c.27)

Applied to Real Axis Function

This bit reflects the status of the drive's "SERCOS® real time" status bit 2.

Applied to Imaginary Axis Function

Not significant.

Applied to the Functions of a Group of Co-ordinated Axes, Group of Slave Axes and SERCOS® Ring

This bit is set to 1 if one or more member axis DRIVE_REALTIME_BIT2 bits (AXIS_IS_LINKED) are set to 1.

AXIS_HOLD Bit (%Ir.m.c.28)

Description

This bit (status 1) indicates that the axis has come to rest following the Hold function.

This bit is set to 0 in the following cases:

- one of the bits quoted below is set to 1:
 - AXIS_HALT,
 - AXIS_FAST_STOP,
 - AXIS_HOMING,
 - DRIVE_DISABLED,
 - AXIS_SUMMARY_FLT.
- the axis is inactive with the AXIS_IN_COMMAND bit set to 0.

Applied to Real and Imaginary Axis Functions

Axis stopped at rest.

Applied to the Functions of a Group of Co-ordinated Axes, Group of Slave Axes and SERCOS® Ring

This bit is set to 1 if all the member axis AXIS_HOLD bits (AXIS_IS_LINKED) are set to 1.

AXIS_HALT Bit (%Ir.m.c.29)

Description

This bit (status 1) indicates that the axis has stopped and cannot receive a movement command.

It is set to 1 when the ALLOW_MOVE bit is set to 0.

This bit is set to 0 in the following cases:

- one of the bits quoted below is set to 1:
 - AXIS_HALT,
 - AXIS_FAST_STOP,
 - AXIS_HOMING,
 - DRIVE_DISABLED,
 - AXIS_SUMMARY_FLT.
- the axis is inactive with the AXIS_IN_COMMAND bit set to 0.

NOTE: The ALLOW_MOVE bit is set to 0 in the MovementControl register.

Applied to Real and Imaginary Axis Functions

Axis stopped with no possibility of movement commands.

Applied to the Functions of a Group of Co-ordinated Axes, Group of Slave Axes and SERCOS® Ring

This bit is set to 1 if all the member axis AXIS_HALT bits (AXIS_IS_LINKED) are set to 1.

AXIS_FASTSTOP Bit (%Ir.m.c.30)

Description

This bit (status 1) indicates that an axis has Fast Stop status.

The bit is set to 0:

- when the axis is disabled (ALLOW_ENABLE),
- following a fault on the axis,
- when the axis is re-activated (CONTROL_ENABLE),
- if one of the DRIVE_DISABLED or AXIS_SUMMARY_FLT bits is set to 1,
- if the axis is inactive with the AXIS_IN_COMMAND bit set to 0.

Applied to Real and Imaginary Axis Functions

Axis in FastStop status.

Applied to the Functions of a Group of Co-ordinated Axes, Group of Slave Axes and SERCOS® Ring

This bit is set to 1 if all the member axis AXIS_FASTSTOP bits (AXIS_IS_LINKED) are set to 1.

AXIS_READY Bit (%Ir.m.c.31)

Description

This bit (status 1) indicates that the axis is ready to receive a movement command.

The axis is active with the following bits:

- AXIS_IN_COMMAND set to 1,
- DRIVE_DISABLED set to 1,
- AXIS_HOMING set to 0,
- AXIS_HOLD set to 0,
- AXIS_HALT set to 0,
- AXIS_FAST_STOP set to 0,
- AXIS_NOT_FOLLOWING set to 0,
- AXIS_SUMMARY_FLT set to 0.

Applied to Real Axis, Imaginary Axis and Group of Co-ordinated Axes Functions

The axis is ready to receive a movement command.

Applied to a Group of Slave Axes

The set is active and ready to be commanded to follow the master axis.

Applied to SERCOS® Function

This bit is set to 1 if all the individual axis (Real Axis, Imaginary Axis and External Setpoint Axis) AXIS_READY bits are set.

CONF_OK Bit (%Ir.m.c.32)

Description

This bit (status 1) indicates that the axis is configured.

NOTE: This bit is only concerned with the Real and Imaginary axis functions.

12.2 Movement Control Functions

Subject of this Section

This section describes the movement control functions.

What Is in This Section?

This section contains the following topics:

Topic	Page
Motion Control Functions: General	173
Management of ALLOW and CONTROL Bits	174
MovementControl Bits Modifiable on Edge	176
Control Bits Modifiable on Change of State	177
Status Bits and Words	178
Manual Mode	179
Set_Functional_Mode Function	181
Home Function	182
Reference Point with the Home Function	184
ForcedHomed Function	185
Unhome Function	186
SetPosition Function	187

Motion Control Functions: General

Motion Control Functions

The motion control functions are used to carry out the following functions, for a given MotionSet (Real axis, Imaginary axis, Remote Axis, Group of slave axes, Network or Group of coordinated axes):

- Reset the fault statuses for a MotionSet (ClearFault),
- Enable or disable a MotionSet (EnableDrive and DisableDrive),
- Stop a MotionSet (FastStop and Halt), immediately or not.
- Hold or resume a MotionSet (Hold and resume),
- Reset a movement axis to Home (Home),
- Move the motion axis away from homing point (see Forcedhome). Unhome
- Release and acquire isolated axes which are part of a Slave axis group or a Coordinated axis group (Release and Acquire),
- Enable or disable a slave axis group (FollowOn and FollowOff),
- Acquire and enable or release and disable slave axes in a FollowOn group (Link and UnLink),

The functions that carry out these operations send a motion control then return to the application server, sometimes before the command has been completed.

The application server has to monitor the MotionGroup status to determine whether command execution is complete.

The application server monitors the status bit (%I) for the axis to check the motion status.

Motion control

Many commands are also sent via motion control bits.

Some motion commands require an ALLOW bit in their configuration before they can be correctly executed.

The ALLOW bits are set to 1 by default on resetting the module, (with the exception of the ALLOW_ENABLE bit for the Network channel), so that all the corresponding motion commands can be authorized.

Management of ALLOW and CONTROL Bits

Principle

Movement control bits (see *Language Objects and IODDTs Associated Specifically with the SERCOS Module, page 408*) are managed in the following way:

- set the channel 0 ALLOW_ENABLE bit (%Qr.m.c.0.26) to 1 at the start of the program,
- use the CONTROL / ALLOW bits to activate / deactivate a function:
 - a CONTROL bit rising edge with the ALLOW bit at 1 activates the function,
 - the ALLOW bit at 0 deactivates the function.

ALLOW Bits

ALLOW bits activate a command on the falling edge of the bit (changing from state 1 to state 0). It has a disallowing action while the bit is at state 0. When the ALLOW bit changes from state 0 to state 1, this removes the inhibiting action and allows the associated commands.

Presymbolization of the ALLOW bits begins with ALLOW_ (see *Language Objects and IODDTs Associated Specifically with the SERCOS Module, page 408*).

NOTE: ALLOW bits are initialized at state 1.

CONTROL Bits

CONTROL bits activate a command on the rising edge of the bit. The command is issued; it will have effect if the MovementSet is in a state, which allows the command to be carried out. You must therefore check the MovementStatus of the MovementSet before activating a command. CONTROL bits have a corresponding ALLOW bit. One common reason for the inability of a CONTROL to activate a command stems from the fact that the ALLOW bit associated to this command or the SERCOS® channel has the value 0.

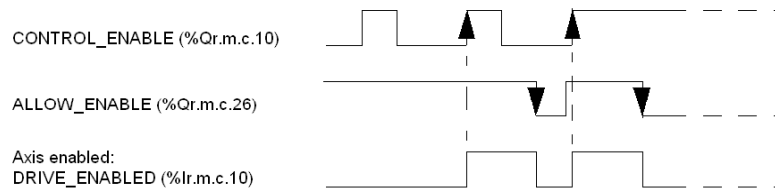
CONTROL bit names begin with CONTROL_ (see *Language Objects and IODDTs Associated Specifically with the SERCOS Module, page 408*).

Special Cases

The pulse period must be greater than the module cycle time (default Cycle_time = 4 ms), so that the command is "seen" by the module.

Example

ENABLE function



MovementControl Bits Modifiable on Edge

ALLOW Bits

The table below lists the ALLOW bits:

Language object	Name	Description
%Qr.m.c.18	ALLOW_ACQUIRE	Falling edge: frees the controlled axes. The AXIS_IN_COMMAND MovementStatus bit is set to 0 when the axes are freed. Inhibit action (status 0): prevents acquisition of controlled axes by this MovementGroup. Associated CONTROL bit: CONTROL_ACQUIRE
%Qr.m.c.26	ALLOW_ENABLE	Falling edge: deactivates the controlled axes. The DRIVE_DISABLED (MovementStatus) bit is set to 1 when the axes are deactivated. Inhibit action (status 0): prevents activation of the MovementGroup. Associated CONTROL bit: CONTROL_ENABLE
%Qr.m.c.27	ALLOW_FOLLOW	Falling edge: inhibits following for a SlaveGroup or a member of such a group. The AXIS_IS_LINKED (MovementStatus) bit is set to 0 when following is inactive. Inhibit action (status 0): inhibits following activation. Associated CONTROL bit: CONTROL_FOLLOW
%Qr.m.c.28	ALLOW_RESUME	Falling edge: sends a hold command to the controlled axes. The AXIS_HOLD MovementStatus bit is set to 1 when the movement profile is paused at zero speed. Inhibit action (status 0): disallows resuming. Hold state remains whenever axes are activated. Associated CONTROL bit: CONTROL_RESUME
%Qr.m.c.29	ALLOW_MOVE	Falling edge: issues a halt command. The AXIS_HALT MovementStatus bit is set when the halt process starts. Inhibit action (0): inhibits movement commands. Halt state remains whenever axes are activated.
%Qr.m.c.30	ALLOW_NOT_FASTSTOP	Falling edge: sends a FastStop command to the controlled axes. The AXIS_FASTSTOP MovementStatus bit is set to 1. Inhibit action (status 0): Inhibits movements and remains in FastStop state when the axes are activated. Rising edge: cancels FastStop if the axes are supplied. The AXIS_FASTSTOP MovementStatus bit is set to 0.
%Qr.m.c.31	ALLOW_NOT_FLT	Falling edge: triggers a user error. The AXIS_SUMMARY_FLT MovementStatus bit is set to 1. Inhibit action (status 0): imposes user error.

Control Bits Modifiable on Change of State

CONTROL Bits

The table below lists the CONTROL bits:

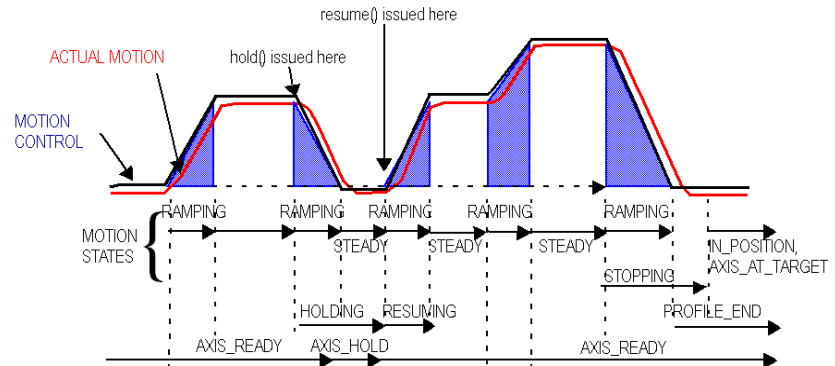
Language object	Name	Description
%Qr.m.c.2	CONTROL_ACQUIRE	Rising edge: acquires command of controlled axes and links them to the MovementGroup. The AXIS_IN_COMMAND bit is set if the operation is performed correctly. Associated ALLOW bit: ALLOW_ACQUIRE
%Qr.m.c.10	CONTROL_ENABLE	Rising edge: activates the controlled axes. The DRIVE_ENABLED bit is set to 1 if the drive is valid. Associated ALLOW bit: ALLOW_ENABLE
%Qr.m.c.11	CONTROL_FOLLOW	Rising edge: activates following for the SlaveGroup or a member of such a group. The AXIS_IS_LINKED bit is set to 1 when following is active. Associated ALLOW bit: ALLOW_FOLLOW
%Qr.m.c.12	CONTROL_RESUME	Rising edge: resumes from a hold (ALLOW_RESUME). The AXIS_HOLD (MovementStatus) bit is set to 0 when the resume starts. Associated ALLOW bit: ALLOW_RESUME
%Qr.m.c.15	CONTROL_CLEAR_FLT	Rising edge: clears movement faults (MovementFault). The AXIS_SUMMARY_FLT bit is set to 0 if the operation is performed correctly.
%Qr.m.c.4	CONTROL_JOG_POS	Rising edge: performs continuous movements in the positive direction. Falling edge: stops the current movement
%Qr.m.c.5	CONTROL_JOG_NEG	Rising edge: performs continuous movements in the negative direction. Falling edge: stops the current movement
%Qr.m.c.13	CONTROL_INC_POS	Rising edge: performs incremental movements in the positive direction.
%Qr.m.c.14	CONTROL_INC_NEG	Rising edge: performs incremental movements in the negative direction.

Status Bits and Words

Introduction

Status bits (%I) give the status of the axis control module (controller) and of the variable speed controller.

The following figure shows a movement profile with various movement conditions indicated.



Manual Mode

Introduction

This mode allows you to use specific commands to perform continuous or incremental movements along real or imaginary axes.

NOTE: This mode is available:

- via the program with a TSX CSY 84 module with SV version ≥ 1.2 or TSX CSY 164 module (any release),
- via the Unity Pro debug screen (version 2.1 or later).

Manual movements use the speed configured by default (1/2 of VMax). However, it is possible to modify:

- the default speed configuration via the debug screen or via the program (*see page 247*),
- the movement speed using the Set_Speed_Override function (*see page 197*),
- the increment value in the chosen position unit via the debug screen.

It is also possible to reference the axis to the origin or to a specific point via the debug screen or via the program (*see page 182*) and (*see page 185*) and (*see page 187*).

The following commands are not available in manual mode:

- MoveImmed (*see page 192*) and MoveQueue (*see page 194*),
- Pause (%Qr.m.c.28) and Resume (%Qr.m.c.12),
- Following (%Qr.m.c.11) and Enable (%Qr.m.c.2).

Accessing Manual Mode

Manual mode can be accessed from the debug screen or via the program (*see page 181*).

NOTE: By default, the axis is in Auto mode after configuration.

The Different Commands

Four commands can be used to perform different movements.

Name	Description
Jog+	used on a rising edge to perform continuous movements in a positive direction (1). used on a falling edge to stop the movement in progress.
Jog-	used on a rising edge to perform continuous movements in a negative direction (1). used on a falling edge to stop the movement in progress.
Inc+	used on a rising edge to perform incremental movements on a referenced axis in a positive direction (2).
Inc-	used on a rising edge to perform incremental movements on a referenced axis in a negative direction (2).
Legend	
(1)	The movement limit values are: - the software limits for a referenced axis, - the maximum for the theoretical position of the axis for a non-referenced axis.
(2)	Increment given in thousandths of the chosen unit. The value of the chosen increment must be within the movement limit values for the axis less the tolerance around target position.

Activation Conditions

The different commands are only activated under the following conditions:

- the servodrive must be enabled,
- manual mode is activated for the specified axis,
- the axis is stopped,
- two commands cannot be sent simultaneously,
- if the axis is referenced and is not within the software limits, the direction of movement must be such that the axis can return within the software limits.

NOTE: If any one of these conditions is not met, a message relating to the missing condition is displayed.

Exception: If, during the same cycle and for the same axis, the module receives a falling edge on the Jog command in progress and a rising edge on another manual command, only the falling edge is taken into account.

Set_Functional_Mode Function

Description

The Set_Functional_Mode allows you to access manual mode.

NOTE: This mode is available for real and imaginary axes.

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD*, page 402).

Language Objects

The table below presents the different language objects linked to the Set_Functional_Mode function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	2572	Command
%MDr.m.c.20	PARAM_CMD_1	-	Value to write (1)
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.
Legend			
(1) To write the value: 0 to change to manual mode 1 to abort manual mode			

Associated Discrete Input

It is possible to check the mode that is currently active by reading the value of following discrete input:

Object	Name	Description
%IXY.i.25	AXIS_MANUAL_MODE	0: Auto Mode (by default) 1: Manual mode

NOTE: The value of the input changes after the motor stops if the mode is changed while a movement is being performed.

Home Function

Description

This function allows a reference point request to be sent to the speed drive.

NOTE: The type of reference point should be defined beforehand in the drive.

Note: The programmable position limits are not active until the reference point of an axis has been completed correctly, or the ForcedHome function is called.

Note: The end of travel switches must be connected before the reference point, since the drive may use these inputs to determine when it should reverse its movement.

During a reference point operation, any attempt to interrupt this operation before the end will cause the axis to stop and fault signalling. It is therefore necessary to acknowledge this fault before the axis can be moved again.

Action of the Function on the Real Axis

The axis must be enabled and in `AXIS_IN_COMMAND`. It must not be moving, holding or faulted, and it must have the `ALLOW_MOVE` MovementControl bit set to 1 in order for the command to be successful.

How to Use this Function

This function is implemented either:

- directly by the bit position control (implicit exchanges)
- using the `WRITE_CMD` (see *WRITE_CMD*, page 402).

Associated Language Objects

Implicit Exchanges: The table below presents the different language objects linked to this function.

Object	Name	Effective	Description
%Qr.m.c.29	ALLOW_MOVE	Status 1	Allow bit
%Qr.m.c.228	ALLOW_RESUME	Status 1	Allow bit
%Qr.m.c.30	ALLOW_NOT_FASTSTOP	Status 1	Allow bit
%Ir.m.c.5	AXIS_HOMED	-	Status bit

Exchange via WRITE_CMD: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MW _r .m.c.26	ACTION_CMD	6034	Command
%MFr.m.c.31	PARAM_CMD_3	-	Direction of the reference point course: 1.0 (positive) or -1.0 (negative)
%MFr.m.c.33	PARAM_CMD_4	-	Speed of the reference point course in SpeedUnits
%MW _r .m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

Reference Point with the Home Function

How to Home SERCOS® Drives

The home switch should be connected to the drive input. The ideal position for the source switch is 180 degrees away from the zero pulse of the resolver or the top of the encoder when the movement is in the direction back towards the specified source. Placement of this transition within a band of ± 45 degrees from the resolver null or encoder marker pulse results in a homing fault (unless the home fault is disabled by the drive's configuration software). The placement position of the switch can be determined by reading the home switch placement error parameter, SERCOS® IDN S-0298 or IDN US-0298, if supported by the drive.

The homing sequence is divided into two steps:

- finding the source switch,
- moving to the source position.

During either step, a Halt, FastStop or DisableDrive function sent to the motion controller will cause homing to abort with an `AXIS_SUMMARY_FLT` and leave the axis in an un-homed condition. If the Hold function is issued during the first step, homing is aborted with an `AXIS_SUMMARY_FLT`, leaving the axis in an un-homed condition. Hold and Resume functions are allowed during the second step.

NOTE: During this step, the drive searches for a reference point switch when moving in the specified source direction. The drive, depending on the drive manufacturer, may reverse direction if it encounters an EOT switch or if it encounters the home switch but not in the specified homing direction. (See the homing description in your SERCOS® drive manual).

In general, this operation involves the following events:

- The drive determines which direction to begin moving the axis:
 - if the Source switch indicates NOT PRESENT, the axis moves in the specified reference point direction.
 - if the Source switch is detected, the axis moves in the opposite direction from the specified reference point direction in order to find the source switch.

ForcedHomed Function

Description

This function is used to force the reference point of an axis. This function is used as part of the manual homing procedure. The application may move the axis to a known position, send a SetPosition function to attribute a known value to the position reading, and then send the ForcedHome function. The movement controller then makes the software position limits effective, and the AXIS_HOMED bit is set to 1.

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via WRITE_CMD: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MW _r .m.c.26	ACTION_CMD	6039	Command
%MW _r .m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

Unhome Function

Description

This function cancels the reference point of an axis.

The movement controller's software position limits are not effective if an axis is not referenced. Executing the Unhome function causes the software position limits to be ignored.

NOTICE

UNEXPECTED AXIS MOVEMENT

Use the end of travel (EOT) switches connected directly to the drive when the axis is not referenced.

Failure to follow these instructions can result in equipment damage.

How to Use this Function

This function is implemented with the help of the instruction `WRITE_CMD` (see page 402).

Associated Language Objects

Implicit Exchanges: The table below presents the different language objects linked to this function.

Object	Name	Effective	Description
%Ir.m.c.6	AXIS_HOMED	-	Status bit

Exchange via `WRITE_CMD`: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	6038	Command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

SetPosition Function

Description

The SetPosition function changes the position reading of the axis to the given value. The position can only be changed if the axis is not moving. The position reading is modified in the axis control module. It does not change the position value on the drive itself. This means that if axis commissioning software tool is monitoring the drive by a direct connection to the drive, the position display will not show the changed position value. However, the positions monitored or commanded by the axis control module will reflect the new parameter.

The Home function clears all previous position settings generated by the SetPosition function.

CAUTION

UNEXPECTED APPLICATION BEHAVIOR - BIAS_REMAIN MODE

Do not send a SET_POSITION command as long as the %Ir.m.c.24 bit value is at 1.

Failure to follow these instructions can result in injury or equipment damage.

NOTE: In the case of a sequence with BIAS_REMAIN, the recognition on the slave of the resulting offset is signalled on the %Ir.m.c.24 bit

How to Use this Function

This function is implemented with the help of the WRITE_CMD (see page 402) instruction.

Associated Language Objects

Exchange via WRITE_CMD: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	2053	Command
%MFr.m.c.31	PARAM_CMD_3	-	Value to write.
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

12.3 Movement Functions

Subject of this Section

 This section describes movement functions.

What Is in This Section?

 This section contains the following topics:

Topic	Page
About Movement Functions	189
MoveType Data Type	191
MoveImmed Function	192
MoveQueue Function	194
GetMoveQueueLength Function	196
GetSpeedOverride and SetSpeedOverride Function	197
EnableRealTimeCtrlBit Function	198

About Movement Functions

Structure of a MOVE Function

The basic structure of a movement function is as follows:

<Move function>(<movement type>, <target position>, <speed to target>)

where:

- Move function is one of:
 - MoveImmed**: aborts any movement profile in progress and moves to the new target position.
 - MoveQueue**: blends a movement profile onto the end of a profile in progress to reach the new target position.
- movement type is one of:
 - ABS_MOVE**: Absolute movement (requires the axis to be referenced). .
The target position is taken as an absolute position value referenced from the source position. The movement takes the shortest route (take care if modulus are activated).
 - ABS_MOVE_POS**: Absolute movement in the positive direction for axes with modulus activated or co-ordinated axes.
 - ABS_MOVE_NEG**: Absolute movement in the negative direction for axes with modulus activated or co-ordinated axes.
 - ABS_MOVE_NO_ROLLOVER**: Absolute movement for independant axes, configured with modulo. A value superior to the configured modulo can be reached.
 - INCR_MOVE**: Incremental move. The target position is taken as an incremental value referenced from the current position (for a MoveImmed) or from the last blended target position (MoveQueue).
 - CONT_MOVE**: Continuous move. The target position indicates the direction of a continuous motion profile that moves to the software limits.
- target position is a real number which specifies the target position in the default position units of the axis.
- speed to target is a real number which specifies the approach speed in the default velocity units of the axis.

Functioning

The MOVE function initiates movement. The movement function will return to the application before the movement profile is complete. The application program uses the axis status bits (%I) to determine the movement profile status. (See the *SERCOS language objects, page 385*).

ABS_MOVE functions that specify a target position that is beyond the maximum or minimum position for the movement axis generate a MovementError since the movement axis cannot reach the target. INCR_MOVE move functions that generate a target beyond the limits do not generate a MovementError since the calculation of the target point may be delayed due to the movement queuing mechanism. Such INCR_MOVE movements are converted into a target position to reach the limit and will generate an AXIS_SUMMARY_FAULT when the limit is reached. Approach speeds that are greater than the maximum speed limit for a movement axis are converted into approach speeds at the speed limit.

Movement Status

The IN_POSITION and AXIS_AT_TARGET bits are set to 0 at the start of a movement. The RAMPING bit is set when the movement axis is commanded to accelerate or decelerate. The STEADY bit is set to 1 when the movement axis is commanded to move at a constant speed. STOPPING is set to 1 at the start of the deceleration to the target position. IN_POSITION is set to 1 when the actual position and the target position are within the in-position band after STOPPING has been set to 1. PROFILE_END is set to 1 when the controller commands the servo to remain at the target position. AXIS_AT_TARGET is set when the actual position and the target position are within the in-position band after PROFILE_END is set to 1. STOPPING will be set to 0 when the axis has reached the PROFILE_END and is IN_POSITION.

MoveType Data Type

Description

The movement of an axis towards a position (movement) can be incremental, continuous or absolute:

- Incremental: The move occurs relative to the current position.
- Continuous: The axis moves towards the maximum or minimum position which is used as a target position for the movement.
- Absolute: The axis moves towards an absolute position.

Value of the Data Type

The following table lists the various values of the data type:

Type of movement	Value
ABS_MOVE	0
INCR_MOVE	1
CONT_MOVE	2
ABS_MOVE_POS	3
ABS_MOVE_NEG	4
ABS_MOVE_UNLINK *	10
CONT_MOVE_UNLINK *	12
ABS_MOVE_POS_UNLINK *	13
ABS_MOVE_NEG_UNLINK *	14
ABS_MOVE_NO_ROLLOVER *	20
ABS_MOVE_NO_ACC_CORRECTION **	30
INCR_MOVE_NO_ACC_CORRECTION **	31

*Function not dealt with in this chapter; see the following chapter.

**Function not dealt with in this chapter; see the following chapter (*see page 226*).

MovImmmed Function

Description

This function causes the specified axis to move towards a given position at the specified speed. The movement is immediately applied and overrides any other move currently in progress. The movement in progress and any queued movements are cancelled.

An immediate movement can be:

- Incremental: The movement occurs relative to the current position.
- Continuous: the axis moves toward the maximum or minimum position. The minimum or maximum position is used as the target position of the Move function.
- Absolute: the absolute movement causes the axis to move to an absolute position.

How to Use this Function

This function is implemented with the help of the `WRITE_CMD` instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via `WRITE_CMD`: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	513	Command
%MDr.m.c.27	PARAM_CMD_1	-	Movement Type (1)
%MDr.m.c.29	PARAM_CMD_2	-	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 10000$). see opcode 2172, real axis (see page 456) or imaginary axis (see page 460).
Legend			
(1): INCR_MOVE, ABS_MOVE, ABS_MOVE_POS, ABS_MOVE_NEG, ABS_MOVE_NO_ROLLOVER or CONT_MOVE			
(2): N= 1 for an independent axis, N includes between 2 and 8 for a group of coordinated axes.			
(3): A = $[31+2(N-1)]$, B = $[3+(N-1)]$			
(4): C = $[31+2(N+N'-1)]$, D = $[3+(N+N'-1)]$			
(5): N' = Position of the axis in the group			

Object	Name	Function code	Description
%MFr.m.c.A (3)	PARAM_CMD_B (3)	-	Target position of axis N (2)
%MFr.m.c.C (4)	PARAM_CMD_D (4)	-	Approach speed to target of N' axis (5)
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.
Legend			
(1): INCR_MOVE, ABS_MOVE, ABS_MOVE_POS, ABS_MOVE_NEG, ABS_MOVE_NO_ROLLOVER or CONT_MOVE			
(2): N= 1 for an independent axis, N includes between 2 and 8 for a group of coordinated axes.			
(3): $A = [31+2(N-1)]$, $B = [3+(N-1)]$			
(4): $C = [31+2(N+N'-1)]$, $D = [3+(N+N'-1)]$			
(5): N' = Position of the axis in the group			

Example of Parameters for a Group of Coordinated Axes

For a movement with 2 member axes:

- PARAM_CMD_1 (%MDr.m.c.27) = Movement type
- PARAM_CMD_3 (%MFr.m.c.31) = Position of axis 1
- PARAM_CMD_4 (%MFr.m.c.33) = Position of axis 2
- PARAM_CMD_5 (%MFr.m.c.35) = Approach speed of axis 1
- PARAM_CMD_6 (%MFr.m.c.37) = Approach speed of axis 2

MoveQueue Function

Description

This function causes the specified axis to move towards a given position at the specified speed.

This function is appended to any moves currently executing or waiting to execute. The MoveQueue is blended into the current trajectory without stopping at the end of the previously executed motion control. A maximum of 32 movements can be queued.

All entries in the queue are deleted when a MoveImmed, Halt or FastStop function is executed, or when an AXIS_SUMMARY_FAULT or DRIVE_FAULT occurs.

A MoveQueue function can be:

- **Incremental:** The movement occurs relative to the last queued target position.
- **Absolute:** the absolute movement causes the axis to move to an absolute position.
- **Continuous:** the axis moves toward the maximum or minimum position. The minimum or maximum position is used as the target position of the movement function.

WARNING

UNEXPECTED APPLICATION BEHAVIOR - MOVEQUEUE FUNCTION BEHAVIOR

A MoveQueue issued after a continuous move will never be executed. The continuous movement will fault when a position limit is reached, and the queued moves will be erased.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Associated Language Objects

Exchange via `WRITE_CMD`: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	520	Command
%MDr.m.c.27	PARAM_CMD_1	-	Movement Type (1)
%MDr.m.c.29	PARAM_CMD_2	-	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 0000$). see opcode 2172, real axis (see page 456) or imaginary axis (see page 460).
%MFr.m.c.A (3)	PARAM_CMD_B (3)	-	Target position of axis N (2)
%MFr.m.c.C (4)	PARAM_CMD_D (4)	-	Approach speed to target of N' axis (5)
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.
Legend			
(1): INCR_MOVE, ABS_MOVE, ABS_MOVE_POS, ABS_MOVE_NEG, ABS_MOVE_NO_ROLLOVER or CONT_MOVE			
(2): N= 1 for an independent axis, N includes between 2 and 8 for a group of coordinated axes.			
(3): $A = [31+2(N-1)]$, $B = [3+(N-1)]$			
(4): $C = [31+2(N+N'-1)]$, $D = [3+(N+N'-1)]$			
(5): N' = Position of the axis in the group			

Example of Parameters for a Group of Coordinated Axes

For a movement with 2 member axes:

- PARAM_CMD_1 (%MDr.m.c.27) = Movement type
- PARAM_CMD_3 (%MFr.m.c.31) = Position of axis 1
- PARAM_CMD_4 (%MFr.m.c.33) = Position of axis 2
- PARAM_CMD_5 (%MFr.m.c.35) = Approach speed of axis 1
- PARAM_CMD_6 (%MFr.m.c.37) = Approach speed of axis 2

GetMoveQueueLength Function

Description

The GetMoveQueueLength function returns the number of entries in the queue for the specified axis.

When the number of entries in the queue is greater than 1, the axis moves bumplessly along a continuous path. Entries are placed in the queue using the MoveQueue function. Entries are removed from the queue as the movement for an axis is planned (current movement and next movement).

If a GetMoveQueueLength is executed immediately after the first entry is placed in the queue by a MoveQueue function, the returned value will be 0. This is because the movement for the axis is planned as soon as the entry is placed in the queue. The trajectory generator removes it before the GetMoveQueueLength function can be executed. Up to 34 movements can be managed: 32 movements in the queue, the current and next movements present in the trajectory generator.

NOTE: The GetMoveQueue only returns the number of entries waiting without taking into account either the current or next movements.

How to Use this Function

This function is implemented with the help of the `WRITE_CMD` instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via `WRITE_CMD`: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	9510	Command
%MDr.m.c.20	RETURN_CMD_1	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

GetSpeedOverride and SetSpeedOverride Function

Description

The GetSpeedOverride function sends the relevant speed modulation coefficient to an axis.

The SetSpeedOverride function sets the relevant speed modulation coefficient for an axis.

The value is explicitly specified as a percentage.

The coefficient is between 0 and 100.

A modulation coefficient of 0% would render the axis immobile.

A modulation coefficient of 100% would cause the axis to move at the commanded speed.

NOTE: The coefficient affects the acceleration and deceleration values.

NOTE: The axis follows the same trajectory whatever the speed modulation coefficient value.

How to Use These Functions

These functions are implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via WRITE_CMD: The table below presents the different language objects linked to these functions.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	1513	Read command
		2513	Write command
%MFr.m.c.22	RETURN_CMD_2	-	Result of the read command
%MFr.m.c.31	PARAM_CMD_3	-	Value to write
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

EnableRealTimeCtrlBit Function

Description

This function is used to enable or disable the RealTime control bits 1 and 2 (%Qr.m.c.6 and %Qr.m.c.7 respectively).

Enabling the control bits causes them to be sent to the drive via the SERCOS network.

NOTE: By default, these bits are inactive.

Refer to the IDN documentation for details of the commands associated with these control bits.

How to Use this Function

This function is implemented with the help of the instruction WRITE_CMD (see page 402).

Associated Language Objects

Exchange via WRITE_CMD: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	2564	Command
%MDr.m.c.27	PARAM_CMD_1	-	1 = Enable 0 = Disable.
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

12.4 Motion Function Following Deactivation of the Sequence

Aim of this Section

This section describes the functions enabling a motion to be sequenced on deactivation of the sequence. These commands are based on "MoveImmed" and "MoveQueue".

What Is in This Section?

This section contains the following topics:

Topic	Page
Overview of the Function	200
"MoveImmed" Function	201
"MoveImmed" Function with ABS_MOVE_NO_ROLLOVER	203
"MoveQueue" Function	205
"MoveQueue" Function with ABS_MOVE_NO_ROLLOVER	207

Overview of the Function

Objective

The objective is to sequence a motion on deactivation of the sequence.

These functions are executed using the "MoveImmed (*see page 201*)" and "MoveQueue (*see page 205*)" commands, with different types of motions.

Type of movement

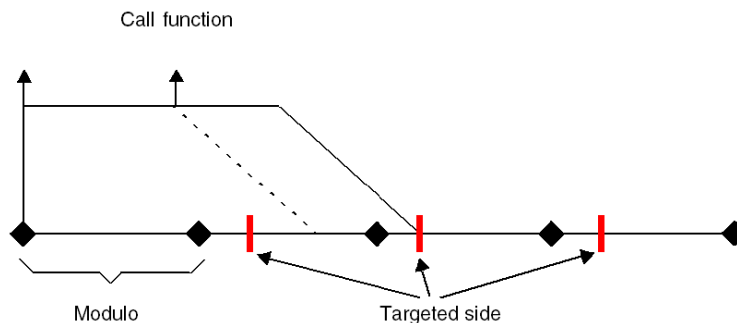
The following table lists the various values of the data type:

Type of movement	Value	Description
ABS_MOVE_UNLINK	10	Absolute motion (as close as possible if the axis is configured with modulo).
CONT_MOVE_UNLINK	12	Continuous motion.
ABS_MOVE_POS_UNLINK	13	Absolute motion, positive direction imposed (axis with active modulo).
ABS_MOVE_NEG_UNLINK	14	Absolute motion, negative direction imposed (axis with active modulo).
ABS_MOVE_NO_ROLLOVER	20	Absolute motion for independant axes configured with modulo. A value superior to the configured modulo can be reached.

"MoveImmed" Function

Objective

This command is used for a sequence deactivation of a member axis of a set of slave axes, then execution (without jolts or halts) of a motion.



Features

This command is based on the current "MoveImmed" (*see page 192*) command (opcode 513).

This command concerns **real axes** or **imaginary axes** that form part of a **slave group** configured in **ratio**, **setpoint following** and **BIAS_REMAIN**. The axes must be **linked**.

Initial condition:

- the master is a real or imaginary axis,
- the speed of the master is taken to be constant between deactivation of the sequence and the following 4 SERCOS cycles,

When a motion of this type is in progress, the status bit "**RESUMING**" is set to 1. The bit is set to 0 by the system after motion completion.

Object	Type	Symbol	Explanation	Comment
%MW.r.m.c.19	Word	ERROR_CMD		Error Code*
%MD.r.m.c.20	Double word	RETURN_CMD_1		not used
%MFr.m.c.22	Floating point	RETURN_CMD_2		not used
%MFr.m.c.24	Floating point	RETURN_CMD_3		not used
%MW.r.m.c.26	Word	ACTION_CMD	513	MoveImmed
%MD.r.m.c.27	Double word	PARAM_CMD_1	10 or 12 or 13 or 14	Type

Object	Type	Symbol	Explanation	Comment
%MDr.m.c.29	Double word	PARAM_CMD_2	Acceleration / deceleration coefficient	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 10000$). see opcode 2172, real axis (<i>see page 456</i>) or imaginary axis (<i>see page 460</i>).
%MFr.m.c.31	Floating point	PARAM_CMD_3	Target position	
%MFr.m.c.33	Floating point	PARAM_CMD_4	Target speed	

***Command rejection:**

- all cases of "MoveImmed" command rejection,
- the axis must be linked (CMD_NOT_ALLOWED: command rejection = 50),
- no motion sequencing function allowed when sequence deactivation already in progress (CMD_NOT_ALLOWED: command rejection = 50).

Case of command abort for bit7 MW_MERGE_UNLINK (*see page 356*):

certain conditions are vital and must be respected for the command to be executed.

If one of these conditions is false, the command is abandoned:

- slave set-up in ratio, setpoint following, BIAS_REMAIN mode,
- the status of the axis must be: AXIS_READY = 1,
- the group's master axis must not be remote,
- there must be no additional motion on the axis and no pause in progress (AXIS_HOLD = 0 and HOLDING = 0).

Notes

Case of a slave with active modulo and direction motion type imposed (13 or 14):

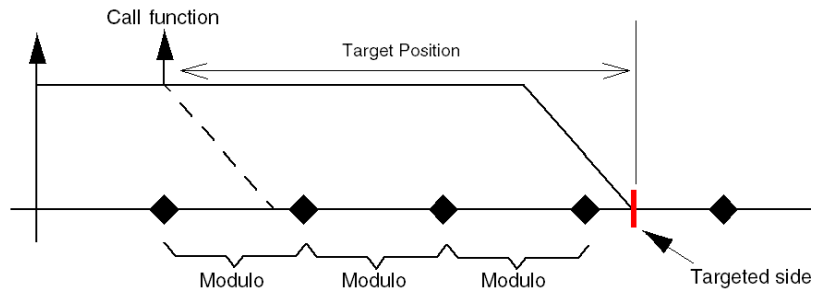
- if the axis is stopped, the motion will be "at most" one rotation,
- if the axis is previously in motion:
 - distance to be traveled and deceleration incline permitting, the axis will be able to switch to the requested target speed before stopping,
 - otherwise, the module will calculate the minimum number of revolutions to perform in order to stop at the absolute requested position (respecting the deceleration incline).

"MoveImmed" Function with ABS_MOVE_NO_ROLLOVER

Objective

This command is used for a sequence deactivation of a member axis of a set of slave axes, then execution (without jolts or halts) of a motion.

It allows an independent axis configured with modulo to reach an absolute position superior than the positive limit of the modulo.



Example: Case of an axis configured with modulus of 360 degrees

- Target position: 1200 degrees

Result: The final position at the end of the movement will be:

(Target Position - Number of Modulos)

$$1200 - 3 \times 360 = 120 \text{ degrees}$$

Features

This command is based on the current "MoveImmed" (*see page 192*) command (opcode 513).

This command concerns **real axes** or **imaginary axes** that form part of a **slave group** configured in **ratio**, **setpoint following** and **BIAS_REMAIN**. The axes must be **linked**.

Initial condition:

- the master is a real or imaginary axis,
- the speed of the master is taken to be constant between deactivation of the sequence and the following 4 SERCOS cycles,

When a motion of this type is in progress, the status bit "**RESUMING**" is set to 1. The bit is set to 0 by the system after motion completion.

Object	Type	Symbol	Values	Comment
%MW.r.m.c.19	Word	ERROR_CMD		Error Code*
%MDr.m.c.20	Double word	RETURN_CMD_1		not used
%MFr.m.c.22	Floating point	RETURN_CMD_2		not used
%MFr.m.c.24	Floating point	RETURN_CMD_3		not used
%MW.r.m.c.26	Word	ACTION_CMD	513	Movelmmed
%MDr.m.c.27	Double word	PARAM_CMD_1	20	Type
%MDr.m.c.29	Double word	PARAM_CMD_2	Acceleration / deceleration coefficient	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 10000$). see opcode 2172, real axis (see page 456) or imaginary axis (see page 460).
%MFr.m.c.31	Floating point	PARAM_CMD_3	Target position	
%MFr.m.c.33	Floating point	PARAM_CMD_4	Target speed	

***Command rejection:**

- all cases of "Movelmmed" command rejection,
- the axis must be linked (CMD_NOT_ALLOWED: command rejection = 50),
- no motion sequencing function allowed when sequence deactivation already in progress (CMD_NOT_ALLOWED: command rejection = 50).

Case of command abort for bit7 MW_MERGE_UNLINK (see page 356):

certain conditions are vital and must be respected for the command to be executed.

If one of these conditions is false, the command is abandoned:

- slave set-up in ratio, setpoint following, BIAS_REMAIN mode,
- the status of the axis must be: AXIS_READY = 1,
- the group's master axis must not be remote,
- there must be no additional motion on the axis and no pause in progress (AXIS_HOLD = 0 and HOLDING = 0).

Notes

Case of a slave with active modulo and direction motion type imposed (20):

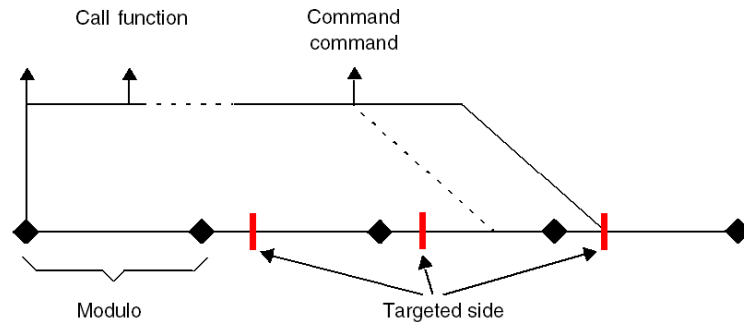
- if the axis is stopped, the motion will be "at most" one rotation,
- if the axis is previously in motion:
 - distance to be traveled and deceleration incline permitting, the axis will be able to switch to the requested target speed before stopping,
 - otherwise, the module will calculate the minimum number of revolutions to perform in order to stop at the absolute requested position (respecting the deceleration incline).

"MoveQueue" Function

Objective

This command is used to prepare the motion, which will be executed (without jolts or halts) on deactivation of the sequence for a member axis of a set of slave axes.

This command does not deactivate the sequence of the axis.



Features

This command is based on the current "MoveQueue" (*see page 194*) command (opcode 520).

This command concerns **real axes** or **imaginary axes** that form part of a **slave group** configured in **ratio**, **setpoint following** and **BIAS_REMAIN**. The axes must be **linked**.

Initial condition:

- the master is a real or imaginary axis,
- the speed of the master is taken to be constant between deactivation of the sequence and the following 4 SERCOS cycles,

The fact that an additional motion is executed on the slave axis after deactivation of the sequence ("MoveQueue" pending) is sent to the axis by status bit **RESUMING**. This bit is set to 0 by the system after the motion is executed.

Object	Type	Symbol	Explanation	Comment
%MWr.m.c.19	Word	ERROR_CMD		Error code
%MDr.m.c.20	Double word	RETURN_CMD_1		not used
%MFr.m.c.22	Floating point	RETURN_CMD_2		not used
%MFr.m.c.24	Floating point	RETURN_CMD_3		not used
%MWr.m.c.26	Word	ACTION_CMD	520	MoveQueue

Object	Type	Symbol	Explanation	Comment
%MDr.m.c.27	Double word	PARAM_CMD_1	Type	10 or 12 or 13 or 14
%MDr.m.c.29	Double word	PARAM_CMD_2	Acceleration / deceleration coefficient	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 10000$). see opcode 2172, real axis (<i>see page 456</i>) or imaginary axis (<i>see page 460</i>).
%MFr.m.c.31	Floating point	PARAM_CMD_3	Target position	
%MFr.m.c.33	Floating point	PARAM_CMD_4	Target speed	

Command rejection:

- all existing cases of "MoveQueue" command rejection (*see page 370*),
- the axis must be linked (CMD_NOT_ALLOWED: command rejection = 50),
- no motion sequencing function allowed when sequence deactivation already in progress (CMD_NOT_ALLOWED: command rejection = 50).

Case of command abort for bit7 MW_MERGE_UNLINK (*see page 356*):

certain conditions between transmission of the command and activation of Unlink are vital and must be respected for the command to be executed.

If one of these conditions is absent, the motion to sequence is abandoned:

- slave set-up in ratio, setpoint following, BIAS_REMAIN mode,
- the status of the axis must be: AXIS_READY = 1 (*see page 170*),
- the group's master axis must not be remote,
- At the time of sequence deactivation, there must be no additional motion on the axis and no pause in progress (AXIS_HOLD = 0 and HOLDING = 0).

Notes

Case of a slave with active modulo and direction motion type imposed (13 or 14):

- if the axis is stopped, the motion will be "at most" one rotation,
- if the axis is previously in motion:
 - distance to be traveled and deceleration incline permitting, the axis will be able to switch to the requested target speed before stopping,
 - otherwise, the module will calculate the minimum number of revolutions to perform in order to stop at the absolute requested position (respecting the deceleration incline).

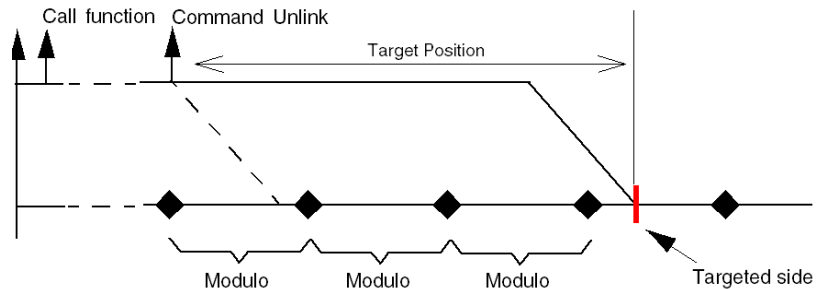
"MoveQueue" Function with ABS_MOVE_NO_ROLLOVER

Objective

This command is used to prepare the motion, which will be executed (without jolts or halts) on deactivation of the sequence for a member axis of a set of slave axes.

This command does not deactivate the sequence of the axis.

It allows an independent axis configured with modulo to reach an absolute position superior than the positive limit of the modulo.



Example: Case of an axis configured with modulus of 360 degrees

- Target position: 1200 degrees

Result: The final position at the end of the movement will be:

(Target Position - Number of Modulos)

$$1200 - 3 \times 360 = 120 \text{ degrees}$$

Features

This command is based on the current "MoveQueue" (see page 194) command (opcode 520).

This command concerns **real axes** or **imaginary axes** that form part of a **slave group** configured in **ratio**, **setpoint following** and **BIAS_REMAIN**. The axes must be **linked**.

Initial condition:

- the master is a real or imaginary axis,
- the speed of the master is taken to be constant between deactivation of the sequence and the following 4 SERCOS cycles,

The fact that an additional motion is executed on the slave axis after deactivation of the sequence ("MoveQueue" pending) is sent to the axis by status bit **RESUMING**. This bit is set to 0 by the system after the motion is executed.

Object	Type	Symbol	Values	Comment
%MW.r.m.c.19	Word	ERROR_CMD		Error code
%MDr.m.c.20	Double word	RETURN_CMD_1		not used
%MFr.m.c.22	Floating point	RETURN_CMD_2		not used
%MFr.m.c.24	Floating point	RETURN_CMD_3		not used
%MW.r.m.c.26	Word	ACTION_CMD	520	MoveQueue
%MDr.m.c.27	Double word	PARAM_CMD_1	20	Type
%MDr.m.c.29	Double word	PARAM_CMD_2	Acceleration / deceleration coefficient	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 0000$). see opcode 2172, real axis (<i>see page 456</i>) or imaginary axis (<i>see page 460</i>).
%MFr.m.c.31	Floating point	PARAM_CMD_3	Target position	
%MFr.m.c.33	Floating point	PARAM_CMD_4	Target speed	

Command rejection:

- all existing cases of "MoveQueue" command rejection (*see page 370*),
- the axis must be linked (CMD_NOT_ALLOWED: command rejection = 50),
- no motion sequencing function allowed when sequence deactivation already in progress (CMD_NOT_ALLOWED: command rejection = 50).

Case of command abort for bit7 MW_MERGE_UNLINK (*see page 356*):

certain conditions between transmission of the command and activation of Unlink are vital and must be respected for the command to be executed.

If one of these conditions is absent, the motion to sequence is abandoned:

- slave set-up in ratio, setpoint following, BIAS_REMAIN mode,
- the status of the axis must be: AXIS_READY = 1 (*see page 170*),
- the group's master axis must not be remote,
- At the time of sequence deactivation, there must be no additional motion on the axis and no pause in progress (AXIS_HOLD = 0 and HOLDING = 0).

Notes

Case of a slave with active modulo and direction motion type imposed (20):

- if the axis is stopped, the motion will be "at most" one rotation,
- if the axis is previously in motion:
 - distance to be traveled and deceleration incline permitting, the axis will be able to switch to the requested target speed before stopping,
 - otherwise, the module will calculate the minimum number of revolutions to perform in order to stop at the absolute requested position (respecting the deceleration incline).

12.5 Functions of actual speed/position

Subject of this Section

This sub-chapter describes the functions of actual speed and position.

What Is in This Section?

This section contains the following topics:

Topic	Page
GetActualSpeedFunction	210
GetCommandSpeed Function	211
Modulo counter service	212
GetUnrolledPosition Function	213
GetUnrolledCommandedPosition Function	214
GetCommandedPosition Function	215

GetActualSpeedFunction

Description

This is a movement parametering function. The GetActualSpeed function returns the current movement speed value for the specified axis.

Command via READ_PARAM

No associated internal word.

Command via WRITE_CMD

ACTION_CMD (%MWr.m.c.26) = 5065
RETURN_CMD_1 (%MDr.m.c.20) = 0
RETURN_CMD_2 (%MFr.m.c.22) = Speed
RETURN_CMD_3 (%MFr.m.c.24) = 0
ERROR_CMD (%MWr.m.c.19):

Code	Name	Description
2014	PROXY_NOT_CONNECTED	The motion axis was never successfully configured.

GetCommandSpeed Function

Description

The GetCommandSpeed function returns the current value of the speed of the specified axis.

How to Use this Function

This function is implemented with the help of the `WRITE_CMD` instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	5066	Command
%MFr.m.c.22	RETURN_CMD_2	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

Modulo counter service

Objective

Count the number of modulo.

At a Glance

This command concerns:

- **real** axes configured using Modulo,
- **imaginary** axes configured using Modulo.

Object	Type	Symbol	Explanation	Comment
%MW.r.m.c.19	Word	ERROR_CMD		
%MDr.m.c.20	Double word	RETURN_CMD_1	NB_MODULO	Modulo number (positive or negative direction)
%MFr.m.c.22	Floating point	RETURN_CMD_2		
%MFr.m.c.24	Floating point	RETURN_CMD_3		
%MW.r.m.c.26	Word	ACTION_CMD	573	ReadNBModulo
%MDr.m.c.27	Double word	PARAM_CMD_1	0	
%MDr.m.c.29	Double word	PARAM_CMD_2	0	
%MFr.m.c.31	Floating point	PARAM_CMD_3	0	
%MFr.m.c.33	Floating point	PARAM_CMD_4	0	

Internally, in TSX CSY, the current position (value of NB_MODULO) is reset to 0 on:

- a homing 6034 (*see page 182*),
- a position forcing 2053 (*see page 187*) inside the revolution.

GetUnrolledPosition Function

Description

The GetUnrolledPosition function returns the axis position without modulo.

NOTE: The axis does not require the axis modulo to be enabled to use this function.

How to Use this Function

This function is implemented with the help of the `WRITE_CMD` instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via `WRITE_CMD`: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MW _r .m.c.26	ACTION_CMD	546	Command
%MF _r .m.c.22	RETURN_CMD_2	-	Result of the command
%MW _r .m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

GetUnrolledCommandedPosition Function

Description

The GetUnrolledCommandedPosition function returns the instantaneous unrolled position setpoint without modulo, which is currently being sent to the drive.

The instantaneous position command changes continually while the axis is moving. It equals the target position of a movement profile only when the axis has stopped moving and AXIS_AT_TARGET bit is set to 1.

NOTE: The axis modulo does not need to be activated to use this function.

How to Use this Function

This function is implemented with the help of the `WRITE_CMD` instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via `WRITE_CMD`: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	547	Command
%MFr.m.c.22	RETURN_CMD_2	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

GetCommandedPosition Function

Description

The GetCommandedPosition function returns the instantaneous position setpoint presently being sent to the drive.

The instantaneous position command changes continually while the axis is moving. It equals the target position of a movement profile only when the axis has stopped moving and AXIS_AT_TARGET bit is set to 1.

How to Use this Function

This function is implemented with the help of the `WRITE_CMD` instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via `WRITE_CMD`: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	1053	Command
%MFr.m.c.22	RETURN_CMD_2	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

12.6 Following Functions

Subject of this Section

 This section describes the following functions.

What Is in This Section?

 This section contains the following topics:

Topic	Page
About Following Functions	217
"Target sequence activation" function	219
"Estimation of Remaining Time Before Activation of Sequence" Function	221
Sequence stop (Unlink) of the slave group set off by a trigger	223
Movement Function for coordinated group compatible with the MMS Quantum function	226
FollowOn 415 Function Description	228

About Following Functions

Activating the Following Function

You can activate the following function by calling the Link (WRITE_CMD) or FollowOn (implicit exchanges) functions. You can deactivate the following function by calling the UnLink (WRITE_CMD) or FollowOff (implicit exchanges) functions.

Following does not start as soon as it is activated: issuing a command to activate following starts a sequence of events, and following is not considered until all the events are complete. For the slave axis, the AXIS_IS_LINKED bit indicates when following actually starts.

Once following actually begins, the slave axes follow the master. If the master is under the control of the multi-axis movement controller, the application program commands the master axis to move to achieve the desired following motion.

Movement with Active Following

The master may be executing a move when following is activated.

- In Ratio mode, the slave axis accelerates to the desired following speed.
- In Cam mode, the slave axis rapidly accelerates to achieve the desired speed specified by the CamProfile.

The slave may be executing a move command when following is activated, under the following conditions:

- If the FOLL_ON_HALT SlaveMode bit is set, then this movement command in progress is issued a halt command when following is triggered. This cancels the movement, and the resulting slave axis movement is based only on following the master axis.
- If the FOLL_ON_HALT SlaveMode bit not set to 1, this movement command in progress is not stopped. The resulting slave axis movement is the sum of the movement from following the master plus the movement from the movement command in progress.

Movement with Inactive Following

The master may be executing a move when following is deactivated.

- In Ratio mode, the slave ratio is decelerated to 0 using the configured deceleration. This stops the movement of the slave axis based on following the master axis.
- In Cam mode, the slave axis will immediately stop following the master without any deceleration.

The slave may be executing a move (in addition to following the master) when following is deactivated, under the following conditions:

- If the FOLL_BIAS_REMAINS SlaveMode bit is set to 0, the movement profile in progress is issued with a stop command. The slave axis reaches zero speed at the end of any deceleration of the movement from following the master and when the movement command in progress is stopped. Following is then deactivated and the AXIS_IS_LINKED MovementStatus bit is set to 0.
- If the FOLL_BIAS_REMAINS SlaveMode bit is set, this move command in progress is not stopped. The movement of following the master is therefore zero, and the AXIS_IS_LINKED bit is set to 0. **The slave axis continues to move according to the movement command in progress.** The AbsoluteSlaveBias remains in effect until the slave axis receives a stop or disable command (%I.r.m.c.24 = 1). This means that any ABS_MOVE command will be offset by the value of the AbsoluteSlaveBias.

In order to command an absolute movement to a slave axis while the AbsoluteSlaveBias remains, you must subtract the AbsoluteSlaveBias from the commanded position:

`commanded position = desired position - AbsoluteSlaveBias`

Once the slave axis completes a halt or disable command, the

AbsoluteSlaveBias is automatically reset, and normal position commands may be sent.

Activating Following in Ratio and Cam Modes

In each of these modes, activating following causes the movement controller to calculate a new value for the SlaveBias. This new value is chosen to prevent the slave axis from moving, given the current master position and the SlaveRatio (for Ratio mode) or CamProfile (for Cam mode). You may adjust SlaveBias to any desired value after following is activated or by issuing an incremental movement command to the slave axis.

Note

In Ratio mode, the SlaveBias value changes with the SlaveRatio. Monitor the SlaveBias after the RAMPING bit of the slave axis is reset.

NOTE: SlaveBias: Position delta between master and slave.

"Target sequence activation" function

Objective

This function is used to synchronize a slave axis and master axis to a preset position.

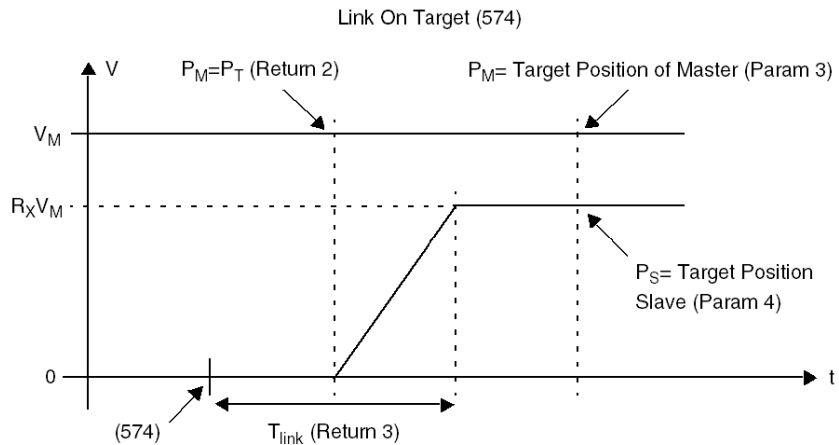
This synchronization is performed using the **"SetMasterTriggerPosition"** and **"FollowOn"** functions.

Operation

The slave axis is stopped while the master axis is in motion.

The user activates the function using OPCODE 574 **"LinkOnTarget"**. Once it receives the request, the module calculates the threshold position of the master.

When the position of the master axis crosses the threshold calculated by the system, the slave axis starts its motion according to the acceleration ramp set up.



The slave axis takes into account:

- "Target position of the slave/master axes" torque,
- the current master position,
- the current master speed,
- the current slave position,
- the Master/Slave ratio,
- the acceleration configured for the slave.

This function applies where the master remains at a constant speed between the function's time of launch and sequence activation "bit **AXIS_IS_LINKED** = 1 (see page 157)". Also, the master cannot be a remote axis.

"LinkOnTarget" function

- **Both rotation directions** (positive and negative) are authorized,
- the master axis can be **real** or **imaginary**,
- the master axis can be configured in modulo,
- the slave axis can be configured in modulo.

The command is given by a WRITE_CMD sent to the slave group channel:

Object	Type	Symbol	Explanation	Comment
%MWr.m.c.19	Word	ERROR_CMD		Command rejected *
%MDr.m.c.20	Double word	RETURN_CMD_1		
%MFr.m.c.22	Floating point	RETURN_CMD_2		Activation position of the master (equivalent to the "GetMasterTriggerPosition (see page 224)" function)
%MFr.m.c.24	Floating point	RETURN_CMD_3		Time remaining before sequence activation of the slave (in seconds)
%MWr.m.c.26	Word	ACTION_CMD	574	LinkOnTarget
%MDr.m.c.27	Double word	PARAM_CMD_1	SlaveNumber	SERCOS address of the slave to link
%MDr.m.c.29	Double word	PARAM_CMD_2	0	not used
%MFr.m.c.31	Floating point	PARAM_CMD_3	MasterPosition	Target position of master axis
%MFr.m.c.33	Floating point	PARAM_CMD_4	SlavePosition	Target position of slave axis

* Command refused:

- slave already linked: 69
- slave in motion: 70
- master stopped: 72
- master in wrong rotation direction: 71
- master not configured: 37
- master is remote axis: 37
- slave not configured in this group: 37
- slave not configured: 7003
- group not configured: 7002

Notes:

The speed of the master must remain constant between the time of the sequence activation request and the sequence activation ("bit **AXIS_IS_LINKED** = 1 (see page 157)").

"Estimation of Remaining Time Before Activation of Sequence" Function

Objective

To estimate the remaining time before activation of a slave axis sequence in a slave group.

Operation

This function can be used for a slave group. The slave axis is taken to be stopped. The master axis is in motion. A sequence activation request is in progress on the slave axis. This function returns an estimation of the remaining time before effective activation of the slave sequencing, "bit **AXIS_IS_LINKED** =1 (*see page 157*)". i.e. the time the master axis needs to reach the position threshold the slave is waiting for, and the time the slave axis needs to reach its setpoint speed (master speed * sequencing ratio). This function supposes the master speed will remain constant between the instant this function is launched and the instant the slave sequencing is activated.

"GetTimeToLink" Function

- **Both rotation directions** are authorized,
- the master axis can be **real** or **imaginary**,
- the master axis can be configured in modulo,
- the slave axis can be configured in modulo.

The command is given by a WRITE_CMD sent to the slave group channel:

Object	Type	Symbol	Explanation	Comment
%MWr.m.c.19	Word	ERROR_CMD		Command rejected *
%MDr.m.c.20	Double word	RETURN_CMD_1		
%MFr.m.c.22	Floating point	RETURN_CMD_2		Estimation of time remaining before sequence activation of the slave
%MFr.m.c.24	Floating point	RETURN_CMD_3		
%MWr.m.c.26	Word	ACTION_CMD	575	GetTimeToLink
%MDr.m.c.27	Double word	PARAM_CMD_1	SlaveNumber	SERCOS address of the slave
%MDr.m.c.29	Double word	PARAM_CMD_2		
%MFr.m.c.31	Floating point	PARAM_CMD_3		
%MFr.m.c.33	Floating point	PARAM_CMD_4		

*** Command refused:**

- master not configured: 37
- master is remote axis: 37
- slave not configured: 37
- slave already linked: 69
- slave in motion: 70
- master stopped: 72
- slave acceleration zero: 1
- master in wrong rotation direction: 71

Notes:

The speed of the master must remain constant between the time this command is used and the sequence activation ("bit **AXIS_IS_LINKED = 1** (*see page 157*)").

Sequence stop (Unlink) of the slave group set off by a trigger

Objective

Sequence stop of an axis forming part of a slave group on position activation.

"SetTriggerUnlink" Function

This command concerns:

Real or imaginary axes configured **with or without Modulo** and linked.

This command is performed by the "SetTriggerUnlink" command based on the "Write_cmd":

Object	Type	Symbol	Explanation	Comment
%MWr.m.c.19	Word	ERROR_CMD		Error code*
%MDr.m.c.20	Double word	RETURN_CMD_1		
%MFr.m.c.22	Floating point	RETURN_CMD_2		
%MFr.m.c.24	Floating point	RETURN_CMD_3		
%MWr.m.c.26	Word	ACTION_CMD	2180	SetTriggerUnlink
%MDr.m.c.27	Double word	PARAM_CMD_1	Slave number	
%MDr.m.c.29	Double word	PARAM_CMD_2	Trigger type	Zero (0), ** > (1), < (2), ≥ (3), ≤ (4).
%MFr.m.c.31	Floating point	PARAM_CMD_3	Trigger position	
%MFr.m.c.33	Floating point	PARAM_CMD_4	0	

On reception of the command, the axis (part of a linked group) automatically makes the Unlink as soon as the trigger condition becomes true.

** Trigger type:

- 0 = immediate sequence deactivation,
- 1 = sequence deactivation when the master position exceeds the threshold in the positive direction,
- 2 = sequence deactivation when the master position exceeds the threshold in the negative direction,
- 3 = sequence deactivation when the master position is greater than or equal to the threshold,
- 4 = sequence deactivation when the master position is less than or equal to the threshold.

* Error code: the command is rejected if:

- the axis is not linked (CMD_NOT_ALLOW (50)),
- the trigger position is outside the software stops (RANGE_ERROR (1)),
- the trigger type is unknown (RANGE_ERROR (1)),
- the slave number is invalid.

Reading the Last Position Trigger Assigned by the SetTriggerUnlink Command

The "**GetTriggerUnlinkPosition**" command is used to read the last position trigger assigned by a previous "**SetTriggerUnlink**":

Object	Type	Symbol	Explanation	Comment
%MWr.m.c.19	Word	ERROR_CMD		Error code*
%MDr.m.c.20	Double word	RETURN_CMD_1		
%MFr.m.c.22	Floating point	RETURN_CMD_2	Trigger position	
%MFr.m.c.24	Floating point	RETURN_CMD_3		
%MWr.m.c.26	Word	ACTION_CMD	1180	GetTriggerUnlink Position
%MDr.m.c.27	Double word	PARAM_CMD_1	Slave number	
%MDr.m.c.29	Double word	PARAM_CMD_2		
%MFr.m.c.31	Floating point	PARAM_CMD_3		
%MFr.m.c.33	Floating point	PARAM_CMD_4		

* Error code: command rejected if the slave number is invalid.

Reading the Last Trigger Type Assigned by the SetTriggerUnlink Command

The "**GetTriggerUnlinkType**" command is used to read the last trigger type assigned by a previous "**SetTriggerUnlink**":

Object	Type	Symbol	Explanation	Comment
%MWr.m.c.19	Word	ERROR_CMD		Error code*
%MDr.m.c.20	Double word	RETURN_CMD_1	Trigger type	Zero (0), > (1), < (2), ≥ (3), ≤(4).
%MFr.m.c.22	Floating point	RETURN_CMD_2		
%MFr.m.c.24	Floating point	RETURN_CMD_3		
%MWr.m.c.26	Word	ACTION_CMD	1181	GetTriggerUnlink Type
%MDr.m.c.27	Double word	PARAM_CMD_1	Slave number	

Object	Type	Symbol	Explanation	Comment
%MDr.m.c.29	Double word	PARAM_CMD_2		
%MFr.m.c.31	Floating point	PARAM_CMD_3		
%MFr.m.c.33	Floating point	PARAM_CMD_4		

* Error code: command rejected if the slave number is invalid.

Movement Function for coordinated group compatible with the MMS Quantum function

Description

This function causes the specified axis to move towards a given position at the specified speed. The movement is immediately applied and overrides any other move currently in progress. The movement in progress and any queued movements are cancelled. The movements are identical to the trajectories generated by the MMS Quantum module.

An immediate movement can be:

- Incremental: the movement occurs relative to the current position (31).
- Absolute: the absolute movement causes the axis to move to an absolute position (30).

How to Use this Function

This function is implemented with the help of the `WRITE_CMD` instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via `WRITE_CMD`:The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	513 or 520	Command
%MDr.m.c.27	PARAM_CMD_1	-	Movement Type (1)
%MDr.m.c.29	PARAM_CMD_2	-	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 10000$). see opcode 2172, real axis (see page 456) or imaginary axis (see page 460).
Legend			
(1): ABS_MOVE_NO_ACC_CORRECTION, INCR_MOVE_NO_ACC_CORRECTION			
(2): N= 1 for an independent axis, N includes between 2 and 8 for a group of coordinated axes.			
(3): A = [31+2(N-1)], B = [3+(N-1)]			
(4): C = [31+2(N+N'-1)], D = [3+(N+N'-1)]			
(5): N' = Position of the axis in the group			

Object	Name	Function code	Description
%MFr.m.c.A (3)	PARAM_CMD_B (3)	-	Target position of axis N (2)
%MFr.m.c.C (4)	PARAM_CMD_D (4)	-	Approach speed to target of N' axis (5)
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.
Legend			
(1): ABS_MOVE_NO_ACC_CORRECTION, INCR_MOVE_NO_ACC_CORRECTION			
(2): N= 1 for an independent axis, N includes between 2 and 8 for a group of coordinated axes.			
(3): A = [31+2(N-1)], B = [3+(N-1)]			
(4): C = [31+2(N+N'-1)], D = [3+(N+N'-1)]			
(5): N' = Position of the axis in the group			

Example

For a movement with 2 member axes:

- PARAM_CMD_1 (%MDr.m.c.27) = Movement type
- PARAM_CMD_3 (%MFr.m.c.31) = Position of axis 1
- PARAM_CMD_4 (%MFr.m.c.33) = Position of axis 2
- PARAM_CMD_5 (%MFr.m.c.35) = Approach speed of axis 1
- PARAM_CMD_6 (%MFr.m.c.37) = Approach speed of axis 2

FollowOn 415 Function Description

Introduction

The FollowOn 415 function specifies to which FollowerSet an axis (that is already a member of several FollowerSet) will be linked.

By adding the identifier of an axis as a parameter of the FollowOn 415 function, each axis member of the FollowerSet can be individually linked to its master axis.

Operating Mode

When an axis belongs to several FollowerSet, the FollowOn 415 function allows to choose the FollowerSet on which its monitoring will be activated. The motions are linked to the master axis of the FollowerSet on which its monitoring has been activated.

Description

The FollowOn 415 function activates the monitoring of:

- Every axis member of the FollowerSet when used without any parameters.
- An axis, when its identifier is used as a parameter.

The function is only available for one FollowerSet.

NOTE:

The operation is not allowed for axes:

- That do not belong to the FollowerSet.
- Where the monitoring is already activated.

Command

Syntax of the instruction:

```
ACTION_CMD (%MWxy.i.26) = 415
```

Command Parameters

The table below lists the command parameters available for the FollowOn 415 function:

Standard Symbol	Value	Address
PARAM_CMD_1	Address of the slave axis member of the group to be linked.	%MDxy.i.27
PARAM_CMD_2	0	%MDxy.i.29
PARAM_CMD_3	0.0	%MDxy.i.31
PARAM_CMD_4	0.0	%MDxy.i.33

Return Command Values

The table below lists the command return values for the FollowOn 415 function:

Standard Symbol	Value	Address
RETURN_CMD_1	0	%MDxy.i.20
RETURN_CMD_2	0.0	%MDxy.i.22
RETURN_CMD_3	0.0	%MDxy.i.24

Error Codes

The table below lists the errors codes sent by the FollowOn 415 function:

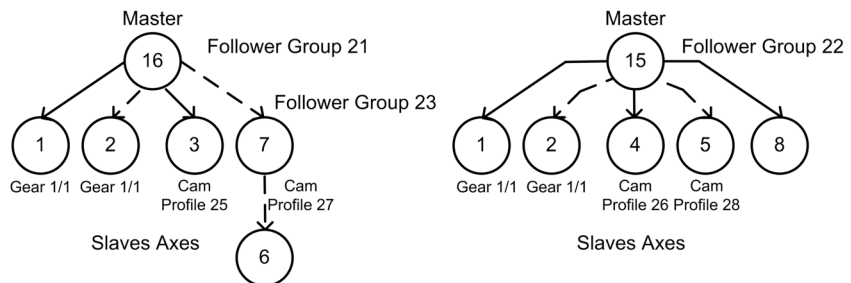
Error Code	Standard Symbol	Meaning	Address
38	NO_SUCH_AXIS	The axe does not belong to the FollowerSet.	-
42	AXIS_NOT_ALLOWED	The axe is not validated.	-
50	CMD_NOT_ALLOWED	Monitoring not allowed on the axe.	-
69	AXIS_LINKED	This axis is already linked.	-
7002	BAD_ID	FollowerSet not configured.	-
7003	WARN_BAD_DATA	Slave axis not configured.	-
-	ERROR_CMD	Possible error while explicitly writing a WRITE_CMD command.	%MWxy.i.19

NOTE: Before linking an axis to a master axis that belongs to another FollowerSet, it is necessary to stop the axis monitoring with the output:

```
%Qxy.i.27 : ALLOW_FOLLOW = 0
```

Example

In the example below, axes 1 and 2 (Gear 1/1) are common to Follower Groups 21 and 22. Axes 1 and 2 can be switched between Follower Group 21 or 22 at any time, using the FollowOn 415 function, even if groups 21 and 22 are both active.



Adjusting the TSX CSY 84 and TSX CSY 164 Modules

13

Aim of This Chapter

This chapter describes the principle for adjusting the SERCOS® application-specific parameters.

All procedures for the TSX CSY 84 are the same for the TSX CSY 164 module.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
13.1	SERCOS® Module Adjustment: General	232
13.2	Adjustment of the Parameters of a Real, Imaginary or Remote Axis	237
13.3	Parameters of a Group of Slave Axes	271
13.4	Cam Profile Adjustment	294
13.5	Sercos® ring parameters	306
13.6	SERCOS® IDN Read/Write Functions	312

13.1 **SERCOS® Module Adjustment: General**

Subject of this Section

This sub-chapter introduces the adjustment screens of the SERCOS® module and gives a reminder of some general principles of explicit exchange.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description of the SERCOS® Module Adjustment Screens: General	233
Adjustment Principles	236

Description of the SERCOS® Module Adjustment Screens: General

Introduction

The adjustment screen displays the adjustment parameters of a SERCOS® channel.

These parameters can be modified in offline or online mode.

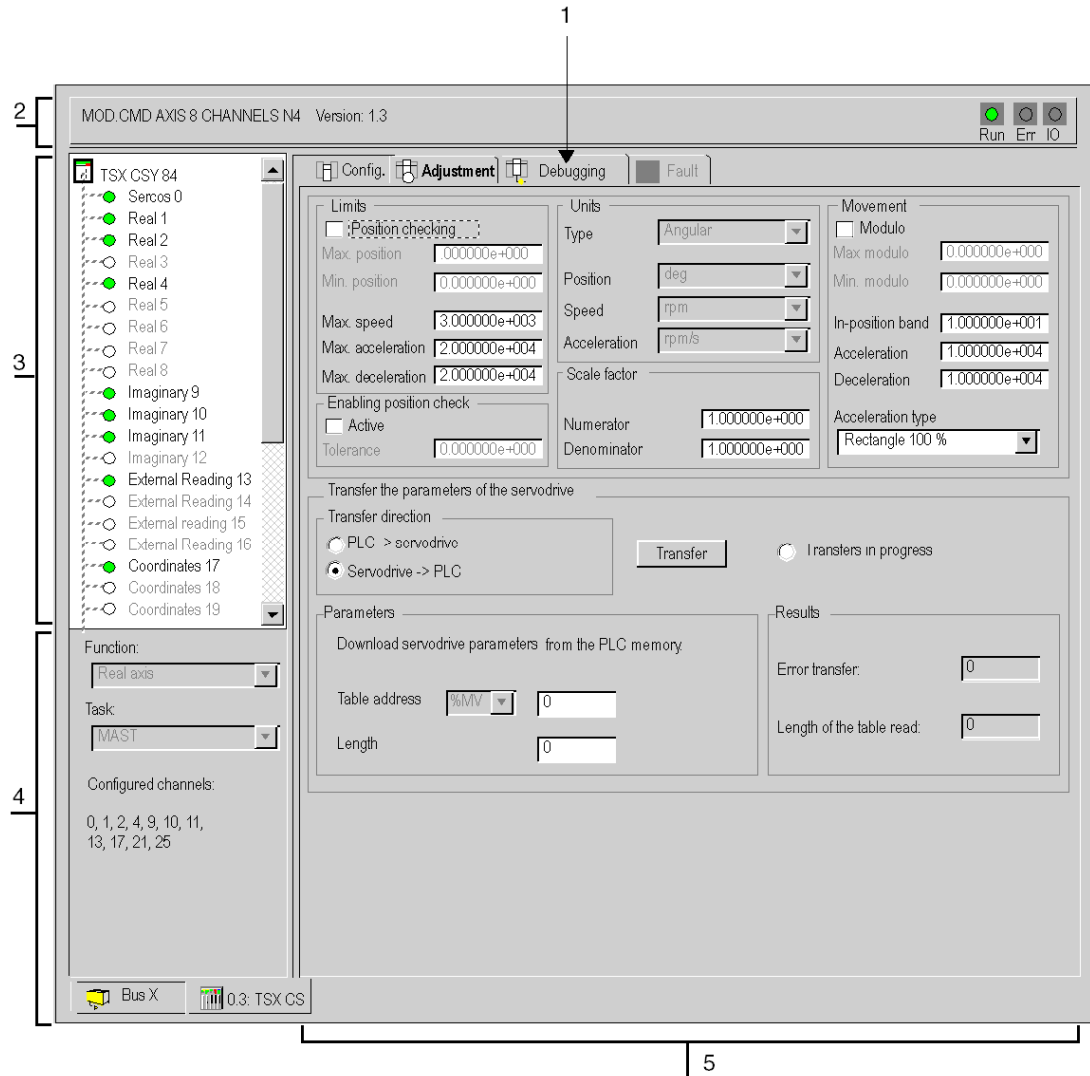
It also gives access to the configuration and debug screen.

The structure of the adjustment screen is very similar to that of the configuration screen.

NOTE: For an introduction to the processes of configuring, adjusting and debugging a project, see the section on Common application specific functions.

Illustration

The following figure shows an example of an adjustment screen for the TSX CSY 84 module:



Description

The table below shows the different elements of the debug screen.

Address	Element	Function
1	Tabs	The tab in the foreground indicates the current mode (Debug in this example). Each mode can be selected using the corresponding tab. Available modes are: ● Debug, accessible only in online mode, ● Diagnostic (Fault), accessible only in online mode, ● Adjustment, ● Configuration.
2	Module zone	Displays the short title of the module. In the same zone there are 3 LEDs showing the status of the module in online mode: ● RUN shows the operating mode of the module. ● ERR indicates an internal module error. ● I/O indicates a fault outside the module or an application fault.
3	Channel zone	Allows you to: ● display the tabs, by clicking on the device reference: ● Description: provides the device characteristics. ● I/O objects (see <i>Unity Pro, Operating Modes</i>): provides a preview symbol of the input/output objects. ● Fault: gives access to the device faults (in online mode). ● choose the channel, ● display the Symbol, channel name defined by the user (via the variables editor).
4	General parameters zone	Used to unforce bits and view the counting function: ● Unforce: button used to unforce forced bits. ● Function: recalls the configured counting function. This information cannot be modified. ● Task: displays the MAST or FAST task configured. This information cannot be modified.
5	Current parameters zone	This zone displays the I/O status and the different parameters for the count in progress. If the contents of the counting register are unusable following an input fault, the message or LED Invalid measure displays red. Note: With CTY 2A/4A modules only, the functions to be displayed in the extended screen may be selected in a window located in the parameters zone on the right. With the CTY 2C module, all functions are automatically displayed.

Adjustment Principles

Introduction

The adjustment function is used to view and modify the operating parameters of the module.

the adjustment of the TSX CSY 84/164 module parameters is done either:

- directly using the adjustment screens with Unity Pro software of version 1.0 or later (offline or online mode),
- or using explicit exchanges (*see page 389*) between the project and the axis control module (online).

NOTE: When the parameters are changed using the Set... adjustment function (WRITE_CMD instruction), the module does not guarantee consistency with the values contained in the internal words %MW.

Channels Enabling Explicit Exchanges

Explicit commands are used to exchange parameters with:

- channel 0 (Network),
- channels 1 to 16 (independent axes),
- channels 21 to 24 (groups of slave axes).

13.2 Adjustment of the Parameters of a Real, Imaginary or Remote Axis

Subject of this Section

This sub-chapter introduces the adjustment screens of a real, imaginary or remote axis as well as the different accessible parameters.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description of the Adjustment Screen of a SERCOS® Module: Real or Imaginary Axes	238
Description of the Adjustment Screen of a SERCOS® Module: Remote Axes	240
Description of the TRF_RECIPE Screen for Real Axes	242
TRF_RECIPE	244
TRF_RECIPE Examples	246
Speed Parameter: DefaultSpeed	247
Speed Parameter: SpeedLimit	248
About Acceleration/Deceleration Parameters	249
Acceleration/Deceleration Parameter: Accel	250
Acceleration/Deceleration Parameter: Decel	251
Acceleration/Deceleration Parameter: AccelMax	252
Acceleration/Deceleration Parameter: DecelMax	253
Acceleration/Deceleration Parameter: AccelType	254
Data type AccelerationType	256
Resolution Parameters: GearRatio	257
Position Limit and Modulo Parameter: PositionLimit	259
About the Modulo	261
Position Limit and Modulo Parameter: RolloverLimit	263
Position Limit and Modulo Parameter: EnableRollover	265
About the In-Position Band	266
Position Limit Parameter: InPositionBand	267
About Enable Mode	268
Position Limit Parameter: EnableMode	269
Behavior Slave Axis Disable	270

Description of the Adjustment Screen of a SERCOS® Module: Real or Imaginary Axes

Introduction

This screen is used to directly modify the configuration parameters of a real or imaginary axis in online mode.

NOTE: For more information about the common parts of the different adjustment screens, see *Description of the SERCOS® Module Adjustment Screens: General*, page 233.

Illustration

The diagram below shows the adjustment zone of a SERCOS® channel configured in a real or imaginary axis.

The screenshot displays a configuration window for a SERCOS module. It is divided into three main sections: Limits, Units, and Movement.

- Limits:** Contains a checked 'Position checking' checkbox. Below it are input fields for 'Max. position' (5.000000e+006), 'Min. position' (5.000000e+002), 'Max. speed' (5.000000e+005), 'Max. acceleration' (5.000000e+004), and 'Max. Deceleration' (5.000000e+002). At the bottom, there is an 'Enabling position checking' section with an 'Active' checkbox and a 'Tolerance' field (0.000000e+000).
- Units:** Features a 'Type' dropdown menu set to 'Linear'. Below are dropdowns for 'Position' (µm), 'Speed' (µm/s), and 'Acceleration' (µm/s²). A 'Scale factor' section includes 'Numerator' (1.000000e+000) and 'Denominator' (1.000000e+000) fields.
- Movement:** Includes an unchecked 'Modulo' checkbox. Below are input fields for 'Max. modulo' (0.000000e+000), 'Min. modulo' (0.000000e+000), 'In-position band' (7.000000e+000), 'Acceleration' (1.000000e+002), and 'Deceleration' (1.000000e+001). At the bottom, the 'Acceleration type' is set to 'Rectangle 100 %'.

Description

The table below shows the **Limit** of the adjustment zone.

Element	Associated parameter	Bit/ word language associated	Symbol/Description
Position checking	-	%MWr.m.c.35:X4	This check box is used to activate the control of position limits.
Max. position	PositionLimits	%MWr.m.c.55	POSITION_MAX
Min. position		%MWr.m.c.57	POSITION_MIN
Max. speed	SpeedLimit	%MWr.m.c.53	SPEED_MAX
Max. acceleration	AccelMax	%MWr.m.c.49	ACCEL_MAX
Min. Acceleration	DecelMax	%MWr.m.c..	DECEL_MAX

The table below shows the **Ctrl of the validation position** of the adjustment zone.

Element	Associated parameter	Bit/ word language associated	Description
Active	-	%MWr.m.c.35:X1	This checkbox can be used to enable position checking.
Tolerance	-	%MWr.m.c.43	Value of the monitoring window (cannot be modified)

The table below shows the **Units** of the adjustment zone. This information cannot be modified.

Element	Associated parameter	Language word associated	Symbol
Type	-	%KWr.m.c.34	-
Position	PositionUnits	%MWr.m.c.65	-
Speed	VelocityUnits	%MWr.m.c.64	-
Acceleration	AccelUnits	%MWr.m.c.63	-

The table below shows the **Scale factor** of the adjustment zone.

Element	Associated parameter	Language word associated	Symbol
Numerator	GearRatio	%MWr.m.c.59	SCALE_NUMERATOR
Denominator		%MWr.m.c.61	SCALE_DENOMINATOR

The table below shows the **Movement** of the adjustment zone.

Element	Associated parameter	Bit/ word language associated	Symbol
Modulo	EnableRollover	%MWr.m.c.35:X0	ENABLE_ROLLOVER
Max. modulo	RolloverLimit	%MWr.m.c.45	ROLLOVER_MAX
Min. modulo		%MWr.m.c.47	ROLLOVER_MIN
In-position band	InPositionBand	%MFr.m.c.41	IN_POSITION_BAND
Acceleration	Accel	%MWr.m.c.36	ACCEL
Deceleration	Decel	%MWr.m.c.38	DECEL
Acceleration type	AccelType	%MWr.m.c.40	ACCEL_TYPE

Description of the Adjustment Screen of a SERCOS® Module: Remote Axes

Introduction

This screen is used to directly modify the configuration parameters of a remote axis in online mode.

NOTE: for more information about the common parts of the different adjustment screens, see *Description of the SERCOS® Module Adjustment Screens: General*, page 233.

Illustration

The diagram below shows the adjustment zone of a SERCOS® channel configured in a remote axis.

Description

The table below shows the **Scale factor** of the adjustment zone.

Element	Associated parameter	Bit/ word language associated	Symbol/Description
Numerator	FollowerRatio	%MFr.m.c.Y (2)	NUMERATOR_X (1)
Denominator			DENOMINATOR_X (1)
Legend			
(1): X included between 1 and 8			
(2): Value of Y, See <i>Slave Group Parameter: FollowerRatio</i> , page 280.			

The table below shows the **Modulo** of the adjustment zone.

Element	Associated parameter	Bit/ word language associated	Symbol/Description
Active	EnableRollover	%MWr.m.c.35:X0	ENABLE_ROLLOVER
Maximum	RolloverLimit	%MWr.m.c.45	ROLLOVER_MAX
Minimum		%MWr.m.c.47	ROLLOVER_MIN

The table below shows the **Units** of the adjustment zone. This information cannot be modified.

Element	Associated parameter	Language word associated	Symbol
Position	PositionUnits	%MW r.m.c. 65	-

Description of the TRF_RECIPE Screen for Real Axes

Introduction

This screen, located at the bottom of the adjustment screen of a real axis, is used to transfer adjustment parameters between the PLC and the servodrive

This function has to be in online mode.

NOTE: For more information about the common parts of the different adjustment screens, see *Description of the SERCOS® Module Adjustment Screens: General*, page 233.

Illustration

The diagram below shows the screen associated with the TRF_RECIPE function.

The screenshot shows a graphical user interface for the TRF_RECIPE function. It is titled "Transfer the parameters of the servodrive". The interface is divided into several sections:

- Transfer direction:** Contains two radio buttons: "Read" (unselected) and "Write" (selected). To the right are two checked checkboxes: "Master" table and "Slave" table. A "Transfer" button is located to the right of these checkboxes.
- Transfer in progress:** A radio button that is currently unselected.
- Parameters:** A section titled "Transfer of the servodrive parameters to the PLC memory." It contains two input fields: "Address table" with a dropdown menu showing "%MW" and a value of "100", and "Length" with a value of "200".
- Results:** A section containing two output fields: "Transfer error:" with a value of "22", and "Length of the table read:" with a value of "0".

Description

The table below shows the **Direction of transfer** of the TRF_RECIPE zone.

Element	Language word associated	Symbol
PLC -> servodrive	%MW _{r.m.c.10}	ACTION_TRF
Servodrive -> PLC		

The table below shows the **Parameters** for the TRF_RECIPE zone.

Element	Language word associated	Description
Address table	%MDr.m.c.11	Two parameters may be modified: the type of object memory for storing data: ● %MW in read write ● %KW in read only
	Immediate value	Address of the beginning of the table
Length	Immediate value	Table length

The table below shows the **Result** of the TRF_RECIPE zone.

Element	Language word associated	Symbol
Error transfer	%MWr.m.c.3	This field provides the error code (<i>see page 376</i>) in the case of an invalid transfer.
Length of the table read	%MDr.m.c.4	Real length of the table

TRF_RECIPE

Reminder

This service is used to read or write variable speed controller parameters with the "real axis" function.

NOTE: This service is used both to read and to write cam profiles and to begin the execution of special functions (see information about these functions).

Syntax of TRF_RECIPE instruction

TRF_RECIPE (IODDT_Var, length, address %MW): transfers from the variable speed controller parameters, from the cam profile, or from the parameters of a special function into/from the table which begins with the address %MW. The length of this table to be transferred is defined by the length parameter. The action to be carried out is defined by the word %MW.r.m.c.10 (ACTION_TRF).

Example: TRF_RECIPE (%CH1.4.3,10,100) (write variable speed controller parameters of real axis 3, of the module situated in position 4 of rack 1, in length table 10, which begins with the address %MW100).

TRF_RECIPE Interface

The command to be carried out is defined in the word %MW.r.m.c.10 and the result of the command is available in the word %MW.r.m.c.3 to %MW.r.m.c.8.

Address	Type	Symbol	Meaning
%MW.r.m.c.3	Word	ERROR_TRF	Write error of the TRF_RECIPE command
%MDr.m.c.4	Double word	RETURN_TRF_1	Return 1 of the function
%MFr.m.c.6	Floating point	RETURN_TRF_2	Return 2 of the function
%MFr.m.c.8	Floating point	RETURN_TRF_3	Return 3 of the function
%MW.r.m.c.10	Word	ACTION_TRF	Action to be carried out
%MDr.m.c.11	Double word	PARAM_TRF_1	Parameter 1
%MDr.m.c.13	Double word	PARAM_TRF_2	Parameter 2
%MFr.m.c.15	Floating point	PARAM_TRF_3	Parameter 3
%MFr.m.c.17	Floating point	PARAM_TRF_4	Parameter 4

Actions Carried Out by TRF_RECIPE

The actions that can be carried out using the TRF_RECIPE service are:

Function	ACTION_TRF (%MWr.m.c.10)	Meaning
Real axis (1)	16001	Write variable speed controller parameters to the PLC memory.
Real axis (1)	26001	Unwrite variable speed controller parameters from the PLC memory.

Legend

(1)	PARAM_TRF_1 to PARAM_TRF_4 = 0

Exchanges

Description

1	Write parameters in the saved memory (Real axis, WRITE_CMD)
2	Unwrite parameters from the saved memory (real Axis, WRITE_CMD)
3	Write parameters into the PLC memory (real Axis, ACTION_TRF = 16001)
4	Unwrite parameters from the PLC memory (real Axis, ACTION_TRF = 26001)

TRF_RECIPE Examples

Example 1: Saving the variable controller parameters

Transferring the variable speed controller parameters from real axis 3 of the module situated at position 4 of rack 1, into the internal words %MW100 to %MW1100, in order to save them. The length of the table being 1000 words. The IODDT-type variable T_CSY_TRF used is called VAR_TRF_1 and is associated with channel 3 of the module.

```
! (*If no exchange of parameters is in progress, then transfer the
  parameters table      into the PLC*)
VAR_TRF_1.ACTION_TRF := 16001;
TRF_RECIPE (VAR_TRF_1, 1000, 100);
END_IF;
```

Speed Parameter: DefaultSpeed

Description

This parameter is used to define a default value for the movement speed of the axis given in SpeedUnits. Move with a speed equal to 0.

NOTE: By default, the system applies a default value equal to one tenth of the value of the SpeedLimit value.

Associated Functions

The associated functions of this parameter are:

- GetDefaultSpeed which sends a read request for the actual default value,
- SetDefaultSpeed which sends a write request for the new default value.

These functions use the `WRITE_CMD` instruction.

Associated Language Objects

The table below presents the different language objects linked to the DefaultSpeed parameter.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	1065	Read command.
		2065	Write command.
%MFr.m.c.22	RETURN_CMD_2	-	Result of the read.
%MFr.m.c.31	PARAM_CMD_3	-	Value to write.
%MWr.m.c.19	ERROR_CMD	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Speed Parameter: SpeedLimit

Description

The parameter is used to define the maximum speed of an axis given in SpeedUnits.

This limit applies to the speed sent to the variable speed controller by the axis control module.

The maximum speed must be greater than or equal to the actual speed.

NOTE: The variable speed controller has its own speed limit.

Associated Functions

The associated functions of this parameter are:

- GetSpeedLimit which sends a read request for the actual maximum speed,
- SetSpeedLimit which sends a write request for the new maximum speed.

These functions use, according to the language objects, the `WRITE_CMD`, `READ_PARAM` or `WRITE_PARAM` instructions.

Language Objects

The table below presents the different language objects linked to the SpeedLimit parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.53	SPEED_MAX	-	READ_PARAM	Read command.
			WRITE_PARAM	Write command.
%MWr.m.c.26	ACTION_CMD	1066	WRITE_CMD	Read command.
		2066		Write command.
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write.
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>).

About Acceleration/Deceleration Parameters

Definition

Acceleration is used to increase the speed of an axis along a movement profile. Acceleration values are limited to the value of the maximum acceleration.

Deceleration is used to reduce the speed of an axis along a movement profile. Deceleration values are limited to the maximum deceleration value.

The acceleration/deceleration ramp is applied to movement commands for that axis.

Movement Planning

The trajectory generator uses this acceleration/deceleration to calculate the commands sent to the controlled drive. The controlled drive has independent acceleration/deceleration parameters. For homing, the drive uses the SERCOS® standard homing acceleration/deceleration parameter (identification number - IDN S-0-0042). For emergency stops, the drive may use a special deceleration value. (See your drive's Proprietary (P) IDN list.)

Maximum Acceleration

The servo drive's maximum acceleration/deceleration limit (SERCOS® Standard IDN S-0-0138 Bipolar Acceleration Limit or IDN S-0-0136 Positive Acceleration Limit) must be greater than the movement controller's AccelMax and DecelMax parameters.

Acceleration/Deceleration Parameter: Accel

Description

This parameter is used to set the acceleration ramp to be applied to movement commands on an axis.

The value of this acceleration is specified using AccelerationUnits.

Associated Functions

The associated functions of this parameter are:

- GetAccel which sends a read request for the actual acceleration ramp value of the designated axis,
- SetAccel which sends a write request for the new acceleration ramp value of the designated axis.

These functions use, according to the language objects, the `WRITE_CMD`, `READ_PARAM` or `WRITE_PARAM` instructions.

Language Objects

The table below presents the different language objects linked to the Accel parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.36	ACCEL	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWFr.m.c.26	ACTION_CMD	1041	WRITE_CMD	Read command
		2041		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write.
%MWFr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Acceleration/Deceleration Parameter: Decel

Description

This parameter is used to set the deceleration ramp to be applied to movement commands on an axis.

The value of this deceleration is specified using AccelerationUnits.

Associated Functions

The associated functions of this parameter are:

- GetDecel which sends a read request for the actual deceleration ramp value of the designated axis,
- SetDecel which sends a write request for the new deceleration ramp value of the designated axis.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM, or WRITE_PARAM instructions.

Language objects

The table below presents the different language objects linked to the AccMax parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.38	DECEL	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWFr.m.c.26	ACTION_CMD	1042	WRITE_CMD	Read command
		2042		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write.
%MWFr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Acceleration/Deceleration Parameter: AccelMax

Description

This parameter is used to define the maximum acceleration applied to an axis.

The value of this acceleration is defined with the units UnitsAcceleration.

NOTE: The maximum acceleration must be greater than or equal to the current acceleration parameter.

NOTE: The maximum acceleration of the controlled speed drive (IDN standardized SERCOSS-0-0138 - Bipolar acceleration limit or IDN S-0-0136 - Positive acceleration limit) must be greater than the AccelerationMax parameter of the movement controller.

Related functions

The functions related to this parameter are:

- GetAccelMax which sends a request to read the value of the current maximum acceleration,
- SetAccelMax which sends a request to write the new maximum acceleration value.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Language objects

The table below shows the various language objects linked to the AccelMax parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.49	ACCEL_MAX	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWFr.m.c.26	ACTION_CMD	1116	WRITE_CMD	Read command
		2116		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the reading.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to be written.
%MWFr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Acceleration/Deceleration Parameter: DecelMax

Description

This parameter is used to define the maximum deceleration applied to an axis.

The value of this deceleration is specified using AccelerationUnits.

NOTE: The maximum deceleration must be greater than or equal to the actual deceleration parameter.

Associated Functions

The associated functions of this parameter are:

- GetDecelMax which sends a read request for the actual maximum acceleration value,
- SetDecelMax which sends a write request for the new maximum acceleration value.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Language objects

The table below presents the different language objects linked to the DecelMax parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.51	DECEL_MAX	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWFr.m.c.26	ACTION_CMD	1117	WRITE_CMD	Read command
		2117		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write.
%MWFr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Acceleration/Deceleration Parameter: AccelType

Description

This function is used to define the acceleration type for the axis.

The acceleration type specifies the shape of the acceleration profile used when changing velocities on an axis. The movement controller provides three major types of acceleration profile:

- **Rectangular:** this acceleration profile changes the velocity in the shortest amount of time. However, it also causes the largest amount of jerks to the axis.
- **Triangular:** this acceleration profile takes the longest period of time to change the velocity (twice as slow as the rectangular acceleration profile). However, it causes the smallest amount of jerk to the axis.
- **Trapezoidal:** The trapezoidal profiles change velocity faster than the triangular but slower than the rectangular acceleration profile. The jerk on an axis is more significant than the triangular but less than the rectangular. There are three choices of trapezoidal acceleration profile, specified by how much longer than the rectangular profile each takes to finish changing the velocity:
 - 1.25 times longer than the rectangular profile,
 - 1.5 times longer,
 - 1.75 times longer.

The acceleration type can be modified at any moment. It will be applied to movements in the queue.

In general, the movement controller processes the commands in the queue ahead (two points along) of where the axis is currently moving to.

NOTE: This is the only way to change the acceleration type of a group of co-ordinated axes. The acceleration type is rectangular by default.

Acceleration codes

The possible acceleration types are as follows:

Code	Type	Description
0	RECTANGULAR_ACCEL	$t1 = 0$, $Tacc = Tacc_{rec}$
1	TRIANGULAR_ACCEL	$Tacc = 2Tacc_{rec}$
2	ACCEL_TRAP_125	$t2 = 3t1$, $Tacc = 1.25Tacc_{rec}$
3	ACCEL_TRAP_150	$t1 = t2$, $Tacc = 1.5Tacc_{rec}$
4	ACCEL_TRAP_175	$t1 = 3t2$, $Tacc = 1.75Tacc_{rec}$

Associated Functions

The associated functions of this parameter are:

- GetAccelType which sends a read request for the actual acceleration type,
- SetAccelType which sends a write request for the new acceleration type.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Associated Language Objects

The table below presents the different language objects linked to the AccelType parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.40	ACCEL_TYPE	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWFr.m.c.26	ACTION_CMD	1540	WRITE_CMD	Read command
		2540		Write command
%MFr.m.c.20	RETURN_CMD_1	-	-	Result of the read.
%MFr.m.c.27	PARAM_CMD_1	-	-	Value to write.
%MWFr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Data type AccelerationType

Description

The acceleration type specifies the shape of the acceleration profile used when changing velocities on an axis. The motion controller provides three major types of acceleration profiles:

- Rectangular: This acceleration profile changes the velocity in the shortest amount of time. However, it also causes the largest amount of jerks to the axis.
- Triangular: This acceleration profile takes the longest period of time to change the velocity (twice as long as the rectangular acceleration profile). However, it causes the smallest amount of jerk to the axis.
- Trapezoidal: The trapezoidal profiles change velocity faster than the triangular but slower than the rectangular acceleration profile. The jerk on an axis is more significant than the triangular but less than the rectangular. There are three choices of trapezoidal acceleration profile, specified by how much longer than the rectangular profile each takes to finish changing the velocity:
 - 1.25 times longer than the rectangular profile
 - 1.5 times longer
 - 1.75 times longer

Value of the Data Type

The following table lists the various data values:

Acceleration type	Value
RECTANGULAR_ACCEL	0
TRIANGULAR_ACCEL	1
ACCEL_TRAP_125	2
ACCEL_TRAP_150	3
ACCEL_TRAP_175	4

Resolution Parameters: GearRatio

Description

This parameter is used to set the ratio of axis position to drive position for an axis.

Ratio

The ratio represents the physical transmission between a motor and the axis movement point (a slide, table, spindle, etc.). It should only need to be changed if the transmission changes. It must be set to a valid value for correct axis positioning to occur. The ratio functions are GetGearRatio and SetGearRatio.

The definition of ratio is:

`axis position units / drive position units`

This ratio is independent of the resolution of the feedback device attached to the motor, and of the default unit system for an axis. It is dependent on the axis units: being linear or angular.

Here is an example, where the drive position units on the SERCOS® drives are set as revolutions:

- a lead screw is attached to the motor and moves the axis 2 inches/turn of the motor, so the ratio is set to `inch(2) / rev(1)`.
- A transmission moves a rotary table 1 turn for 10 turns of the motor, so the ratio is set to `rev(1) / rev(10)`.

Imaginary Axis and Ratio

A servo drive does not control an Imaginary Axis. The drive position units must be encoder feedback counts. The axis position units can be any angular or linear units. The ratio should represent typical resolutions of position feedback devices. For example:

A ratio of: `rev(1) / feedback count(65536)`

A ratio of: `mm(1) / feedback count(1000)`

Remote Axis and GearRatio

The position feedback for a RemoteSetpoint is the feedback register for that axis. The 32-bit integer placed in the register is assumed to have feedback count units. The ratio is chosen to convert to the desired axis units. For example:

A ratio of: `rev(1) / feedback count(10000)`

A ratio of: `mm(1) / feedback count(1000)`

Associated Functions

The associated functions of this parameter are:

- GetGearRatio which sends a read request for the actual ratio,
- SetGearRatio which sends a write request for the new ratio.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Associated Language Objects

The table below presents the different language objects linked to the Accel parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.59	SCALE_NUMERATOR	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MFr.m.c.61	SCALE_DENOMINATOR	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWFr.m.c.26	ACTION_CMD	1500	WRITE_CMD	Read command
		2500		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read (Numerator).
%MFr.m.c.24	RETURN_CMD_3	-	-	Result of the read (Denominator).
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write (Numerator).
%MFr.m.c.33	PARAM_CMD_4	-	-	Value to write (Denominator).
%MWFr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Position Limit and Modulo Parameter: PositionLimit

Description

This function is used to define the positive and negative limits for the axis.

The value of these limits is specified as PositionUnits.

The positive or negative limit cannot be set to a value that causes the current position to be outside of the new limits. Use the SetPosition function or a movement function to change the current position to a value that is within the new limits before changing a limit.

NOTE: Position limits are not active until after the Home reference point function.

Associated Functions

The associated functions of this parameter are:

- GetPositionLimit which sends a read request for the actual positive and negative limit values (parameter function PARAM_CMD_1) of the designated axis,
- SetPositionLimit which sends a write request for the new positive and negative limit values (parameter function PARAM_CMD_1) of the designated axis,

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

NOTE: When an incremental movement is required and the target position exceeds a **position limit** (%MFr.m.c.55 and %MFr.m.c.57 of the real or imaginary axis), the target will take as its objective the **position limit**, and the axis will switch to **limit fault** at the end of the movement. This remains true in the case of a co-ordinated type movement.

Language objects

The table below presents the different language objects linked to the PositionLimit parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.55	POSITION_MAX	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MFr.m.c.57	POSITION_MIN	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MW.r.m.c.26	ACTION_CMD	1505	WRITE_CMD	Read command
		2505		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: 1 : positive limit -1 : negative limit
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write: Position.
%MW.r.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

About the Modulo

Modulo Limits

An individual axis can be configured to roll over the position reading when the axis crosses either the positive or negative modulo limit. Rolling over the position reading constrains the position reading to stay within the range of values between the negative and positive modulo limits.

For example, when controlling a rotary table and you want the axis to report position readings constrained between 0 and 360 degrees. If the modulus are activated when the axis reaches the positive modulo limit of 360 degrees, the position reading is rolled over to read the negative limit value of 0 degrees.

Position readings are constrained to be greater than or equal to the negative modulo limit, or less than the positive modulo limit. The position reading never equals the positive modulo limit. In this rotary table example, with modulo limits between 0 and 360 degrees, the axis never returns a position reading of 360 degrees. Instead, it returns 0 degrees.

The difference between the positive modulo limit and the negative modulo limit is known as the modulo range. This rotary table example has a modulo range of 360 degrees, or 1 revolution.

The positive and negative modulo limits can be any value, as long as the negative limit is less than and not equal to the positive limit. The negative limit does not have to be a negative number, only less than the positive limit. For the rotary table example, common negative and positive limit settings might be 0 and 360 degrees, or -180 and +180 degrees. Less common settings might be 360 and 720 degrees. All these settings have a modulo range of 360 degrees.

Modulo may be activated or deactivated for an axis.

In this case, the position readings are not rolled over, and the distance the axis has traveled is reported based on the 0 position. That is, distance is specified as a result of the home command and the SetPosition function. This position measurement is referred to as the position without modulo.

Modulo Limits and Position Limits

The positive and negative modulo limits are independent of the positive and negative position limits. The position limits are compared with the position without modulo of an axis to determine if the End Of Travel for an axis has been reached. The limits should be deactivated

If the position limits are set to values that are inside the modulo range of an axis that has activated modulos, the multi-axis motion controller prevents the axes from moving beyond the position limits. The axis does not have unlimited movement in this configuration.

In certain projects, you may want to set the position limits to values greater than the modulo limits. For example, suppose your project controls a rotary table that may have some attached cabling. The table may only be able to rotate a few turns before the cabling twists too much and becomes damaged. In such a project, you will certainly want to set position limits in order to limit the movement to prevent cabling damage. But you may also wish to set the modulo limits to one revolution of the table. For example, if the cable becomes tangled after five turns of the table, the project would set the positive position limit to the value (5 turns) and the negative position limit to the value(-5 turns). The modulo limits might be set at 0 degrees and 360 degrees. The axis is then prevented from moving more than 5 revolutions away from its source.

Position Limit and Modulo Parameter: RolloverLimit

Description

This parameter is used to define the positive and negative limits of the modulo of an axis.

The axis does not require the axis modulo to be enabled to use this function. This function does not activate the rollover. (Use the EnableRollover function). Rollover limits can be changed while an axis is enabled, but not while it is moving. The negative modulo limit must always be less than the positive modulo limit.

Associated Functions

The associated functions of this parameter are:

- GetRolloverLimit which sends a read request for the actual positive and negative limit values (parameter function PARAM_CMD_1) of the designated axis,
- SetRolloverLimit which sends a write request for the new positive and negative limit values (parameter function PARAM_CMD_1) of the designated axis.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Language objects

The table below presents the different language objects linked to the RolloverLimit parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.45	ROLLOVER_MAX	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MFr.m.c.47	ROLLOVER_MIN	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWFr.m.c.26	ACTION_CMD	1539	WRITE_CMD	Read command
		2539		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: 1 : positive limit -1 : negative limit

Object	Name	Function code	Instruction type	Description
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write: modulo
%MWrr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Position Limit and Modulo Parameter: EnableRollover

Description

This parameter is used, for a given axis, to take account of position values or not.

The position arguments for move commands specify the desired target position for the axis in terms of rolled-over positions.

It is possible to change this parameter when the following is on. The positive and negative rollover limits must be configured to the desired values.

Language objects

This function uses, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

The table below presents the different language objects linked to the Enable_Rollover parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.35.0	ENABLE_ROLLOVER	1	WRITE_PARAM	Write command: Activation
		0		Write command: Deactivation
%MWr.m.c.26	ACTION_CMD	411	WRITE_CMD	Write command: Activation
		412		Write command: Deactivation
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

About the In-Position Band

In-Position Band Functions

The in-position band functions are `GetInPositionBand` and `SetInPositionBand` (see `Axis Position Adjustment`). In-position band is the tolerance around a target position used to determine the `IN_POSITION` and `AXIS_AT_TARGET` states. If the absolute difference between the actual position and the target position is less than the in-position band, the module word reports `IN_POSITION` and `AXIS_AT_TARGET` values as follows:

- The `IN_POSITION` bit is set after the axis has stopped (`STOPPING` status bit) at a target position and if the axis position is in the in-position band of the target position. It is not set to 1 if the position goes outside of the in-position band, or at the start of the next movement. It is not set to 1 during a `Hold` or `FastStop`, but is set when `Halt` has finished, aborting the movement profile. `IN_POSITION` is set to 0 while deactivated, and is set when re-activated.
- `AXIS_AT_TARGET` is set after the axis has arrived at the end of a movement profile (`PROFILE_END` status bit) at a target position and the axis position is within the in-position band of the target position. It is not set to 1 if the position goes outside of the in-position band, or at the start of the next movement. It is not set if a `Halt`, `FastStop`, `disable drive`, or `fault` interrupts a movement profile before the axis stops at the target position, even if the axis position ends up within the in-position band of the target. Deactivating and re-activating the drive does not modify the state of the `AXIS_AT_TARGET` bit. This bit is not set to 1 if `AXIS_POS_LIMIT` or `AXIS_NEG_LIMIT` are set. `AXIS_AT_TARGET` is the same as `IN_POSITION`, except that:
 - it is not changed when the drive is deactivated.
 - it is set to 0 at the end of a halt and is also not changed after re-activating the drive,
 - it is set to 0 if the axis is at a limit,
 - it is checked after `PROFILE_END` is set while `IN_POSITION` is checked after `STOPPING` is set to 1.

The `AXIS_AT_TARGET` bit may not be set to 1 following the normal movement functions, such as `Halt`. Wait for `PROFILE_END` and `IN_POSITION` to be set, and then check for `AXIS_AT_TARGET` as a separate instruction.

Position Limit Parameter: InPositionBand

Description

This parameter is used to set the value of the in-position band of the specified axis.
This value is given in PositionUnits.

Associated Functions

The associated functions of this parameter are:

- GetInPositionBand which sends a read request for the actual position value,
- SetInPositionBand which sends a write request for the new position value.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Associated Language Objects

The table below presents the different language objects linked to the InPositionBand parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.41	IN_POSITION_BAND	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWFr.m.c.26	ACTION_CMD	1035	WRITE_CMD	Read command
		2035		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write
%MWFr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

About Enable Mode

Axis Enable Mode

The enable mode for an axis, specified via the SetEnableMode function, controls the value of the position command that is first issued to the drive as it is enabling.

The two modes, which select between SETPOINT and CURRENT VALUE, are described below:

Value	Mode	Description
0	MESURE	The feedback position of an axis is read from the drive, and the drive is commanded to servo to that position while it is enabling. This prevents the axis from moving when it is enabling. This is the default enabling mode and is also the highly recommended mode for enabling.
1	SETPOINT	The last commanded position while disabling the axis is used as the position command to the drive while enabling. This mode allows the axis to continue servoing to the same position when the axis is continually enabled and deactivated. If the axis moves while disabled, it will snap back to the prior setpoint as soon as the axis is enabled. The profile of the snap movement is controlled only by the servo gain parameters in the drive, which might cause abrupt movement. This mode should only be used when it can be determined that any movement during disabling is very small. The EnablePositionBand (tolerance) is used to control the maximum permissible deviation from the disabled commanded position. If the axis moves less than the tolerance while disabled, the disabled commanded position is used and the axis snaps back. If the axis moves more than the tolerance while disabled, then the present position of the axis is used and the axis remains still while disabling. An example situation is an axis with a brake that can only be moved under control module power.

Tolerance in SETPOINT Mode

In-position band is a tolerance band used when the SETPOINT enable mode is the defined EnableMode. If the CURRENTVALUE enable mode is defined, in-position band has no effect.

If the axis is configured in SETPOINT mode, and the axis moves less than the in-position band distance after the axis is deactivated, the axis control module:

1. Retrieves the last commanded position that was issued before the axis was deactivated.
2. Commands the axis to move to this position when the axis is re-activated.

If the axis moves more than the in-position band distance, the module re-activates the axis by provisionally changing to CURRENTVALUE mode. The module:

1. Reads the position of the axis.
2. Commands the axis to remain at this position when the axis is re-activated.

This prevents the axis from moving more than the in-position band when the axis is re-activated in SETPOINT mode.

Position Limit Parameter: EnableMode

Description

This parameter sets the axis activation mode: CURRENT VALUE or SETPOINT.
The default mode is ACTUAL mode and is the highly recommended mode.

WARNING

UNEXPECTED APPLICATION BEHAVIOR - ENABLEMODE BEHAVIOR

You should set the COMMANDED enable mode only after reading the enable mode description and clearly understanding that if the axis moves less than the EnablePositionBand while disabled, the axis will "snap" back to the last setpoint position prior to disabling. Adequate safeguards, such as axis brakes with no manual brake release and small, enabled following error limits, should be used to prevent undesirable motion while enabling.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Associated Functions

The associated functions of this parameter are:

- GetEnableMode which sends a read request for the actual activation mode of the designated axis,
- SetEnableMode which sends a write request for the new activation mode of the designated axis.

These functions use the WRITE_CMD instruction.

Associated Language Objects

The table below presents the different language objects linked to the EnableMode parameter.

Object	Name	Function code	Description
%MW.r.m.c.26	ACTION_CMD	1524	Read command
		2524	Write command
%MDr.m.c.20	RETURN_CMD_1	-	Result of the read.
%MDr.m.c.27	PARAM_CMD_1	-	Value to write: Activation mode. 0 : Measurement 1 : Setpoint
%MW.r.m.c.19	ERROR_CMD	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Behavior Slave Axis Disable

At a Glance

The aim of this development is to be able to choose the stop behavior of the drives when a real axis latching command is made:

- Either by immediately releasing the torque in "freewheel" (case of mechanically linked axes)
- Or by braking according to the incline of the emergency stop (IDN3022).

Two solutions are then possible:

- Configuring the drive
- Configuring the module

Choice to Configure the Drive

This solution enables this possibility to be used on all drives in the range.

Choice to Configure the Module

Depending on the value of bit 6 (`_FREEWHEEL_STOP`) of the `%MWr.m.c.35` (`FUNCTION_VALIDATION`) word in the real axis parameters, the results are:

- If the bit is equal to 0, the drive will stop by braking according to the emergency stop incline before switching to freewheel (if the drive was configured previously with Unilink, `ACTFAULT=1`),
- If the bit equals 1, the drive will switch to freewheel immediately.

NOTE: This choice will not be able to be accessed by the configuration or parameter screens. To activate the choice, you must use a `WRITE_PARAM` then a `SAVE_PARAM`.

13.3 Parameters of a Group of Slave Axes

Subject of this Section

This section describes the adjustment parameters for a group of slave axes.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description of the Adjustment Screen of a SERCOS® Module: Slave Axes	272
About a Group of Slave Axes	274
SlaveMode Data	277
Slave Group Parameters: FollowerMode	278
Slave Group Parameter: FollowerConfig	279
Slave Group Parameter: FollowerRatio	280
About the Master Offset	282
Slave Group Parameter: MasterOffset	283
Slave Group Parameter: MasterTriggerPosition	285
Slave Group Parameter: AbsFollowerBias	287
Slave Group Parameter: FollowerBias	290
Propagation of stops in a slave group	291
Alignment Function of the Emergency Stop Inclines	292

Description of the Adjustment Screen of a SERCOS® Module: Slave Axes

Introduction

This screen is used to directly modify the configuration parameters of a slave axis in local or online mode.

NOTE: It is not possible to delete a master or a slave.

NOTE: For more information about the common parts of the different adjustment screens, see *Description of the SERCOS® Module Adjustment Screens: General*, page 233.

Illustration

The diagram below shows the adjustment zone of a SERCOS® channel configured in a real or imaginary axis.

Description

The table below shows the adjustment zone of a SERCOS® channel configured in a slave axis.

Element	Associated parameter	Bit/ word language associated	Description
Master	MASTER_CHANNEL	%MWr.m.c.35	Number of the master axis (1 to 16, N is not accessible)
Slave	SLAVE_CHANNEL_Z	%MWr.m.c.36	Number of slave axis Z
Reading/setpoint	FOLL_TYPE_Z	%MWr.m.c.37:X2	Following type of slave Z
Gearing/Cam	FOLL_POSITION_Z)	%MWr.m.c.37:X1	Slave Z ratio or cam mode)
N°	NUMERATOR_Z	%MFr.m.c.38	Slave Z numerator
Offset	DENOMINATOR_Z	%MFr.m.c.40	Slave Z denominator
Start	FOLL_START_Z	%MWr.m.c.37:X8 to X10	Start conditions
Bias	FOLL_BIAS_REMAINS_Z	%MWr.m.c.37:X7	Bias value
Trigger	TRIGGER_POSITIO_Z	%MFr.m.c.42	Value of trigger for slave axis Z

Element	Associated parameter	Bit/ word language associated	Description
Stop on follow-on	-	%MWr.m.c.37:X3	Slave axis Z stopped if link deleted.
Master/def stopped	-	%MWr.m.c.37:X6	Master stopped because of follow-on error

About a Group of Slave Axes

Slave Group

A SlaveGroup is an axis set that allows a group of axes to follow the movement of a master axis. There are two main modes for following a master axis:

- **Ratio:** The slave axes follow the master axis by a ratio known as Ratio.
- **Cam:** The movement of the slave axes in relation to the master is listed in a cam profile.

The SlaveGroup is told which axis is the master axis, but the master axis is not installed within the SlaveGroup. There is a single master axis that all slaves follow. The master axis can be any type of individual axis. It is possible to change the axis that you want to designate as the master axis.

All the axes that you want to follow a master must be installed into the SlaveGroup. A total of six axes may be installed into a SlaveGroup. Each slave axis may follow the master axis according to its own Ratio or Cam profile. Multiple slaves can share one Cam profile.

Using a SlaveGroup

To use a SlaveGroup, generally proceed as follows:

- Configure all the individual axes, which belong to this SlaveGroup.
- Configure the SlaveGroup.
- Enable all the drives in the SlaveGroup by issuing the EnableDrive function on the SlaveGroup.
- Enable following for slave axes using the Acquire and FollowOn bits, or deactivate slave axes by executing the FollowOff and Release bits.

You may command the slave axes to move when following is activated or deactivated. Any movement commands issued to a slave axis while following is activated will result in additional movement as well as the movement resulting from the slave following the master.

Threshold Position

As an option, you may establish a master threshold position. The SetMasterTrigger-Position function sets the master position at which following will be triggered for the specified axis. The SlaveMode indicates if the master threshold position will be used. When this feature is used, and following is activated, following actually begins when the master reaches the master threshold position.

Bias Value

When following occurs, the value of bias indicates the distance between where the slave is and where it is supposed to be, according to the Cam or Slave Ratio. Make sure that following is actually occurring before calling the `GetFollowerBias` function. (Do not rely on the fact that following was activated). For a slave axis, its `AXIS_IS_LINKED` movement status bit will be set when following is taking place. If you call `GetFollowerBias` when following is not taking place, the return value will indicate the commanded position of the slave axis.

Use the Bias when you want to have the slave axis at a known point when the master is at a given point. This function is not useful for applications that command the slave to simply follow the master's speed by a certain ratio. It is also not useful when the slave needs to move a known incremental distance (for example, a "registration" movement).

In Ratio mode, the Bias value changes while the slave axis accelerates to the following speed after following is activated or the Ratio is modified.

The Bias value is latched when following begins. In Cam mode, the position where the slave is supposed to be is determined by the present master position and the corresponding entry in the Cam profile. In Ratio mode, the following formula indicates where the slave is supposed to be:

where the slave axis should be = present master position * Ratio

When following begins, the movement controller sets the Bias value for the distance between the present position of the slave axis and where it should be. This Bias prevents the slave from jumping to where it is supposed to be when following begins. The following movement of the slave axis will then be offset from where it is supposed to be, unless the Bias is zero. To reset the Bias, an incremental movement must be issued to the slave axis with the movement value equal to the negative of the value returned from the `GetFollowerBias` function.

Non-Zero Bias

You may want to have a non-zero Bias. In Cam mode, it is possible to offset the movement from the values listed in the table. In Ratio mode, it is possible to shift the linear relationship, so that a plot of master positions versus slave positions does not pass through the point:

slave position = 0 when master position = 0

That is, in Ratio mode, you may control where the slave should be for any particular master position. The Bias of a slave axis is modified by sending movement commands to that slave axis.

The `SlaveMode` can be changed at any time for any axis. This allows you to change from Cam to Ratio or vice versa, as well as changing Cam profiles or Slave Ratios.

Master Offset

Some applications are simplified by using the MasterOffset functions. Consider the case where one slave is supposed to follow a sinewave profile, while another slave is supposed to follow a cosine profile. For this case, a single Cam profile containing sinewave data could be shared by both slave axes, where MasterOffset values of 0 and 90 degrees would be set for the sinewave and cosine followers respectively.

Status Bits

The MovementStatus bits for a slave axis relate the state of a movement command sent to that axis. For example, the PROFILE_END bit being set does not indicate that the slave axis is stationary. The slave axis will move if the master axis moves. When a movement command is sent to the slave axis, the PROFILE_END bit clears until this movement by the slave axis completes itself. The other status bits have similar behavior that reflects the state of the slave axis movement command, not the status of the master axis.

SlaveMode Data

Description

Within a SlaveGroup, there are two main methods for following a master axis:

- Ratio: The slave axes follow the master axis using a ratio.
- Camming: The movement of the slave axes in relation to the master is defined by a CamProfile.

Once all the axes have been installed into a SlaveGroup, the axes of the group are configured for following by specifying which type of SlaveMode to use for each axis. The following table lists SlaveMode conditions. Bit 0 is the least significant bit (LSB).

SlaveMode Data

The following table lists the various data values:

Name	Bit	Description
FOLL_WHERE	0	Value = 0: Required
FOLL_TYPE	1	Determines the following type: 0 = Ratio 1 = Cam
FOLL_POSITION	2	Determines which position to follow: 0 = current value 1 = setpoint
FOLL_FOL_ON_HALT	3	Determines action to perform when following is turned on: 0 = do not halt any in-progress movement profile on the slave axis 1 = halt any in-progress movement profile on the slave axis
Reserved	4, 5	
FOLL_HALT_MASTER	6	Determines whether the master should be halted if a slave faults while the following is on: 0 = keep master moving 1 = halt master
FOLL_BIAS_REMAINS	7	Determines the action to perform when following is turned off: 0 = halt any additional movement profile on the slave axis 1 = do not halt any additional movement profile on the slave axis
FOLL_START	8... 10	Determines the start condition: 0 = immediately 1 = start when master position crosses master trigger position in the positive direction 2 = start when master position crosses master trigger position in the negative direction 3 = start when master position is greater than or equal to the master trigger position 4 = start when master position is less than or equal to the master trigger position

Slave Group Parameters: FollowerMode

Description

This parameter contains the following mode in progress for the specified slave axis.

Associated Functions

The function associated with this parameter is:

- GetFollowerMode which sends a read request for the actual following mode,
This function uses the WRITE_CMD instruction.

Language objects

The table below presents the different language objects linked to the FollowerMode parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	1529	WRITE_CMD	Read command
%MDr.m.c.20	RETURN_CMD_1	-	-	Result of the read.
%Mr.m.c.i.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)
Legend				
(1)	This identifier is contained in the constant word %KWr.m.c.0			

Slave Group Parameter: FollowerConfig

Description

This function is used to set the FollowerMode and corresponding slave profile (FollowerRatio or CamProfile) for the specified axis.

NOTE: A Cam profile must be configured before it is referred to.

NOTE: It is possible to change this parameter when the following is on. However, when the following is activated in Cam mode, or is switching to Cam mode, the slave's speed may change rapidly if the new movement profile is significantly different from the old profile.

Associated Functions

The function associated with this parameter is:

- SetFollowerConfig which sends a write request for the new following modes and movement profile,

This function uses the WRITE_CMD instruction.

Associated Language Objects

The table below presents the different language objects linked to the FollowerConfig parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	420	WRITE_CMD	Write command
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MDr.m.c.29	PARAM_CMD_2	-	-	Value to write: Follower mode. See <i>SlaveMode Data</i> , page 277
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write: Numerator for ratio mode, Movement profile for cam mode.
%MFr.m.c.33	PARAM_CMD_4	-	-	Value to write: Denominator for ratio
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors</i> , page 370)
Legend				
(1)	This identifier is contained in the constant word %KWr.m.c.0			

Slave Group Parameter: FollowerRatio

Description

This parameter is used to change the Ratio of the specified slave axis.

The ratio numerator indicates how much the slave will move when the master axis moves the distance showing on the denominator. The ratio value can be modified at any time.

The Ratio DOES NOT CHANGE instantaneously in order to obtain a smooth movement when the slave is activated and the following movement takes place.

Associated Functions

The associated functions of this parameter are:

- GetFollowerRatio which sends a read request for the tolerance of the actual position of the designated axis,
- SetFollowerRatio which sends a write request for the tolerance of the actual position of the designated axis,

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Language objects

The table below presents the different language objects linked to the FollowerRatio parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.38	NUMERATOR_1	-	READ_PARAM	Slave 1 read command
			WRITE_PARAM	Slave 1 write command
%MFr.m.c.40	DENOMINATOR_1	-	READ_PARAM	Slave 1 read command
			WRITE_PARAM	Slave 1 write command
%MFr.m.c.46	NUMERATOR_2	-	READ_PARAM	Slave 2 read command
			WRITE_PARAM	Slave 2 write command
%MFr.m.c.48	DENOMINATOR_2	-	READ_PARAM	Slave 2 read command
			WRITE_PARAM	Slave 2 write command
%MFr.m.c.54	NUMERATOR_3	-	READ_PARAM	Slave 3 read command
			WRITE_PARAM	Slave 3 write command
%MFr.m.c.56	DENOMINATOR_3	-	READ_PARAM	Slave 3 read command
			WRITE_PARAM	Slave 3 write command

Object	Name	Function code	Instruction type	Description
%MFr.m.c.62	NUMERATOR_4	-	READ_PARAM	Slave 4 read command
			WRITE_PARAM	Slave 4 write command
%MFr.m.c.64	DENOMINATOR_4	-	READ_PARAM	Slave 4 read command
			WRITE_PARAM	Slave 4 write command
%MFr.m.c.70	NUMERATOR_5	-	READ_PARAM	Slave 5 read command
			WRITE_PARAM	Slave 5 write command
%MFr.m.c.72	DENOMINATOR_5	-	READ_PARAM	Slave 5 read command
			WRITE_PARAM	Slave 5 write command
%MFr.m.c.78	NUMERATOR_6	-	READ_PARAM	Slave 6 read command
			WRITE_PARAM	Slave 6 write command
%MFr.m.c.80	DENOMINATOR_6	-	READ_PARAM	Slave 6 read command
			WRITE_PARAM	Slave 6 write command
%MWr.m.c.26	ACTION_CMD	1114	WRITE_CMD	Read command
		2114		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of reading the Numerator variable.
%MFr.m.c.24	RETURN_CMD_3	-	-	Result of reading the Denominator variable.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: Identifier of the axis of which the ratio is sent back.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write: ratio numerator.
%MFr.m.c.33	PARAM_CMD_4	-	-	Value to write: ratio denominator.
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

About the Master Offset

Value of the Master Offset

A FollowerAxis follows the sum of the master's position and the MasterOffset value. Typically, a MasterOffset value is changed while following is off. The MasterOffset may be changed while following is on, but be careful to avoid undesirable motion. The MasterOffset is applied immediately, without any blending. A large change in the MasterOffset with following on will cause the follower to move to the new following position rapidly.

Master Offset Features

MasterOffset optional features : GetMasterOffset and SetMasterOffset can simplify some applications. Consider the case where one follower axis is supposed to follow a sinewave profile, while another follower axis is supposed to follow a cosine profile. For this case, a single CamProfile containing sinewave data could be shared by both follower axes, where MasterOffset values of 0 and 90 degrees would be set for the sinewave and cosine followers, respectively.

Slave Group Parameter: MasterOffset

Description

This parameter is used to define the master offset value for the specified slave axis.

Each slave axis can have a unique offset applied to the master position. This is helpful when you want to use a single CamProfile for multiple followers, but each follower's cam shape is rotated some amount with respect to another follower's cam.

NOTE: The MasterOffset may be changed while following is on, but be careful to avoid undesirable motion. Large changes should be blended in as a series of small increments to avoid rapid, immediate changes in the follower position.

Associated Functions

The associated functions of this parameter are:

- GetMasterOffset which sends a read request to the actual value of the master offset of the designated axis,
- SetMasterOffset which sends a write request for the new value of the master offset of the designated axis.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Associated Language Objects

The table below presents the different language objects linked to the MasterOffset parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.40	DENOMINATOR_1	-	READ_PARAM	Slave 1 read command
			WRITE_PARAM	Slave 1 write command
%MFr.m.c.48	DENOMINATOR_2	-	READ_PARAM	Slave 2 read command
			WRITE_PARAM	Slave 2 write command
%MFr.m.c.56	DENOMINATOR_3	-	READ_PARAM	Slave 3 read command
			WRITE_PARAM	Slave 3 write command
%MFr.m.c.64	DENOMINATOR_4	-	READ_PARAM	Slave 4 read command
			WRITE_PARAM	Slave 4 write command
%MFr.m.c.72	DENOMINATOR_5	-	READ_PARAM	Slave 5 read command
			WRITE_PARAM	Slave 5 write command
%MFr.m.c.80	DENOMINATOR_6	-	READ_PARAM	Slave 6 read command
			WRITE_PARAM	Slave 6 write command

Object	Name	Function code	Instruction type	Description
%MW.r.m.c.26	ACTION_CMD	1532	WRITE_CMD	Read command
		2532		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read: Position variable.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write: Position.
%MW.r.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)
Legend				
(1)	This identifier is contained in the constant word %KWr.m.c.0			

Slave Group Parameter: MasterTriggerPosition

Description

This parameter is used to set the master position at which following will be triggered for the specified axis.

This value is only used if the follower mode indicates that the following start condition should use the master trigger position.

Associated Functions

The associated functions of this parameter are:

- GetMasterTriggerPosition which sends a read request for the actual position value of the master,
- SetTriggerPosition which sends a write request for the new position value of the master.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Associated Language Objects

The table below presents the different language objects linked to the MasterTriggerPosition parameter.

Object	Name	Function code	Instruction type	Description
%MFr.m.c.42	TRIGGER_POSITION_1	-	READ_PARAM	Slave 1 read command
			WRITE_PARAM	Slave 1 write command
%MFr.m.c.50	TRIGGER_POSITION_2	-	READ_PARAM	Slave 2 read command
			WRITE_PARAM	Slave 2 write command
%MFr.m.c.58	TRIGGER_POSITION_3	-	READ_PARAM	Slave 3 read command
			WRITE_PARAM	Slave 3 write command
%MFr.m.c.66	TRIGGER_POSITION_4	-	READ_PARAM	Slave 4 read command
			WRITE_PARAM	Slave 4 write command
%MFr.m.c.74	TRIGGER_POSITION_5	-	READ_PARAM	Slave 5 read command
			WRITE_PARAM	Slave 5 write command
%MFr.m.c.82	TRIGGER_POSITION_6	-	READ_PARAM	Slave 6 read command
			WRITE_PARAM	Slave 6 write command

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	1531	WRITE_CMD	Read command
		2531		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read: Position variable.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write: Position.
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)
Legend				
(1)	This identifier is contained in the constant word %KWr.m.c.0			

Slave Group Parameter: AbsFollowerBias

Description

This parameter is used to stop following and move a slave axis to an absolute position while not stopping the slave.

Absolute Bias

The Absolute Bias is the difference between the Bias - returned by the GetFollowerBias function - and the additive commanded position to a slave axis. The Absolute Bias can be used in conjunction with an absolute move command (ABS_MOVE) to position the follower axis to a position that can be calculated.

Position Commanded to a Slave Axis

The commanded position to a slave axis is the sum of the position command from following the master and the Bias:

$\text{Command\_pos} = \text{SlaveMethod}(\text{master\_pos} + \text{master\_offset}) + \text{Bias};$

where FollowerMethod is for:

- **Ratio**: multiply the master argument by the FollowerRatio.
- **Cam**: index into the CamProfile by the given master argument.

Bias

The Bias is also composed of two parts:

$\text{Bias} = \text{additive\_pos\_command} + \text{abs\_bias};$

where:

- **additive_pos_command** is the position command being generated by any move commands directed to the follower axis itself.
- **abs_bias** is a value that is latched whenever Follower Modes are changed (such as activating slave, or changing Cam Profiles or Follower Ratios). The latched value is calculated to avoid any jump in the position command when the new mode is used. This latched value is referred to as the Absolute Bias.

Using Absolute Bias

The Absolute Bias can be used to position the follower axis to some known absolute position if the value from the FollowerMethod function is known and is constant. One situation where the Absolute Bias can be used is during gearing to stop following the master axis but to keep moving the follower axis at a continuous speed. Set the FOLL_BIAS_REMAINS Follower Mode bit to 1, and use the SetFollowerConfig function to configure the mode. Deactivate following, and immediately issue a continuous immediate move command in the desired direction with the desired speed. The Slave Ratio decelerates to 0 at the deceleration rate while the immediate movement accelerates to the desired velocity with the acceleration rate. Both of these ramps will occur simultaneously with each other leaving the axis moving at the desired speed when both ramps are finished (the follower axis RAMPING bit is cleared). In Ratio mode with a FOLL_BIAS_REMAINS and following deactivated, the SlaveMethod function part of the position command calculation is 0, giving:

`pos_command = additive_pos_command + abs_bias;`

The follower axis can now be moved to some absolute target by issuing an absolute move command, compensated by the Absolute Bias, or:

`foll_axis.moveImmed (ABS_MOVE, target - abs_bias, speed);`

Once this move's profile is completed, issue a halt or disable drive command to the follower, then normal move commands can be issued, unaffected by a Bias value.

The value returned is the Absolute Bias. If modulus are activated, the Absolute Bias is returned with modulo.

Associated Functions

The function associated with this parameter is:

- GetFollowerMode which sends a read request for the actual following mode, This function uses the WRITE_CMD instruction.

Associated Language Objects

The table below presents the different language objects linked to the AbsFollow-erBias parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	1526	WRITE_CMD	Read command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)
Legend				
(1)	This identifier is contained in the constant word %KWr.m.c.0			

Slave Group Parameter: FollowerBias

Description

This parameter contains the Bias value.

If rollovers are enabled, the Bias is returned as a rolled value.

Associated Functions

The function associated with this parameter is:

- GetFollowerMode which sends a read request for the actual following mode,
This function uses the WRITE_CMD instruction.

Associated Language Objects

The table below presents the different language objects linked to the FollowerBias parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	1527	WRITE_CMD	Read command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)
Legend				
(1)	This identifier is contained in the constant word %KWr.m.c.0			

Propagation of stops in a slave group

Master Stop

Selecting the type of master stop when there is a slave fault:

List of choices:

- No master stop (default value)
- Normal master stop according to the configured STOP incline
- Master stop by latching command

NOTE: This choice will not be able to be accessed by the configuration or parameter screens. To activate the choice, you must use either a WRITE_PARAM or a SAVE_PARAM.

Description:

Management bit: %MW.r.m.c.37:X11 (FOLL_FAULT_MASTER_x).

- A value of 1 on bit 11 with bit 6 set to 0 will trigger no master stop
- A value of 1 on bit 11 (FOLL_FAULT_MASTER) and bit 6 (FOLL_HALT_MASTER) of the FOLLOWER_MODE word in the slave group parameters means the master will stop according to the emergency stop incline when there is a slave fault.
- A value of 0 on bit 11 with bit 6 set to 1 will trigger a normal master stop (equivalent to a STOP).

Slave Stop

Selecting a slave stop when there is a master fault:

- Stop on master sequence (default value)
- Slave stop by latching command

Description:

Management bit: %MW.r.m.c.37:X12 (FOLL_FAULT_SLAVE_x).

A value of 1 on bit 12 (FOLL_FAULT_SLAVE) of the FOLLOWER_MODE word in the slave group parameters means the slave will stop according to the emergency stop incline when there is a master fault.

(DECSTOP Deceleration ramp on Lexium configurable using Unilink, IDN P 3022 or ASCII code).

NOTE: The action of these bits is set only after writing to the group configuration %Kx constant words. In order to do this, perform a MOD-PARAM on the group.

Alignment Function of the Emergency Stop Inclines

At a Glance

This function manages calculation and modification of the IDN3022s (or of the DECESTOP parameter when running Unilink) in Lexium drives.

It allows emergency stop inclines for slave drives in a slave group to be configured so that all axes stop at the same moment when a fault occurs.

In order to do this, you create a new OPCODE (570) that can be used on the slave groups. This command will result in **calculation of the master and slave deceleration ramps** according to the incline that has been configured, then sending the data to all real axis drives of the group (including the master).

The parameter of this function is a **deceleration incline expressed in the unit selected** during configuration of the group master. If the master is an external setpoint, we will select as the unit the position unit/.²

Object	Type	Symbol	Explanation	Comment
%MW.r.m.c.19	Word	ERROR_CMD		Error code
%MDr.m.c.20	Double word	RETURN_CMD_1		not used
%MFr.m.c.22	Floating point	RETURN_CMD_2		not used
%MFr.m.c.24	Floating point	RETURN_CMD_3		not used
%MW.r.m.c.26	Word	ACTION_CMD	570	ADJUST_3022
%MDr.m.c.27	Double word	PARAM_CMD_1	0	not used
%MDr.m.c.29	Double word	PARAM_CMD_2	0	not used
%MFr.m.c.31	Floating point	PARAM_CMD_3	Deceleration	
%MFr.m.c.33	Floating point	PARAM_CMD_4	0	not used

Example

```

IF NOT %MW104.21.0:X1 THEN (* no WRITE_CMD in progress *)
%MW104.21.26:=570;                                     (* Action
570 = ADJUST_3022 *)
%MF104.21.31:=1000.0;                                (* Deceleration of master
= 1000 Rev/mn/s*)
WRITE_CMD %CH 104.21;
END_IF;

```

Command rejection:

- Slave group that has cam profile slaves => **error code 62**
- The deceleration for a slave or a master is above the maximum deceleration entered during configuration => **error code 63**. Note: This error code also appears when a zero incline is requested.
- The IDN "DECSTOP" calculated for an axis is refused by the drive (parameter out of range) => **error code 1002**.

NOTE: this function will only operate if the drives are Lexium drives. ≥ 5.50 . It will be accessible only by groups in which all the slaves are configured in ratio. Adjustment of the inclines is not saved and can be lost if there is a power outage or drive Reset.

13.4 Cam Profile Adjustment

Subject of this Section

This section describes adjustment functions for a cam profile.

What Is in This Section?

This section contains the following topics:

Topic	Page
About Cam Profiles	295
Cam Profile Parameter: Length	296
Cam Profile Parameter: LookUpFollowerPosition Function	297
Cam Profile Parameter: Coord	298
Cam Profile Parameter: InterpType	300
Cam Profile Parameter: ProfileId	301
Description of the TRF_RECIPÉ Screen for Cam Profiles	302
TRF_RECIPÉ Instruction	304

About Cam Profiles

Introduction

A cam profile relates the position of a slave axis for a given position of a master axis. The cam profile is a table of master and slave position co-ordinates. Position points that are not explicitly listed in the table are derived by interpolating between the given points. Interpolation can be linear or cubic.

Master Positions and Increments

The master positions must be in increasing order in the table from the first element in the profile until the last. For example, the master positions cannot go from 360 degrees to 0 degrees because these master positions are not increasing.

The increments between master positions can be variables. A fixed increment is not required. For example, the following are acceptable master position values: 0, 1, 2, 100, 300, 360. Any valid position units can be used.

Typical Applications

Applications require smooth movement, as movement proceeds from one iteration through a profile to the next. To make this transition as smooth as possible, the value of the slave position for the first data point should be the same value of the slave position for the last data point. For example, if master positions are in degrees, and slave positions are in inches, and if the first position is for the master at 0 degrees, the slave at 0 inches, then the corresponding last position would be the master at 360 degrees with the slave at 0 inches.

Establishing a Cam Profile

Each point in a cam profile table has a position value for both co-ordinate axes of the cam profile. This means a cam profile has two values per point: one for the master position and one for the slave position. The SetCoord function changes both values of an existing point in a cam profile. The GetCoord function returns both values of the designated point in a cam profile.

Modifying the Interpolation Type

In a cam profile table of points, positions that are not in these tables are derived by interpolating between the given points. Linear and cubic interpolation are supported. Modification of the interpolation type is supported via the GetInterpType and SetInterpType functions.

Cam Profile Parameter: Length

Description

This parameter contains the number of points specified for a profile.

The number of points varies according to the profile configuration.

The table below introduces the different configuration possibilities.

Configuration...	Then...
from a file	the number of points is automatically determined
from the holding register	one of these registers identifies the length
without data and should be defined by the SetCoord function	this function call contains the number of points

Associated Functions

The function associated with this parameter is:

- GetLength which sends a read request for the number of points in the profile

This function uses the WRITE_CMD instruction.

Associated Language Objects

The table below presents the different language objects linked to the Length parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	534	WRITE_CMD	Read command
%MDr.m.c.20	RETURN_CMD_2	-	-	Result of the read.
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Cam Profile Parameter: LookUpFollowerPosition Function

Description

This function contains the position of the profile slave, to provide a specified master position.

The slave position is derived by using the interpolation type that has been configured for this profile.

Associated Functions

The associated function is:

- GetLookUpFollowerPosition which sends a read request to the position of the profile slave,

This function uses the WRITE_CMD instruction.

Language objects

The table below presents the different language objects linked to the Length parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	537	WRITE_CMD	Read command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read: Slave position.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write: Master position
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Cam Profile Parameter: Coord

Description

This function is used to modify the co-ordinates of an existing cam profile point.

The point index starts at zero. Therefore, to modify the first entry in a cam profile, use an index value of 0.

Cam profiles are specified as a table of co-ordinate position points. These co-ordinates are accessed using the PosCoord data structure. Each point has a position value for both co-ordinate axes of the profile. This means a cam profile has two values per point: one for the master position and one for the slave position.

The master positions must be in increasing order from the first element in the cam profile until the last. For example, the master positions cannot go from 360 degrees to 0 degrees because these master positions are not increasing.

The increments between master position values can be variables. A fixed increment is not required. For example, the following are acceptable master position values: 0, 1, 2, 100, 300, 360. Any valid position units can be used.

CAUTION

UNEXPECTED EQUIPMENT BEHAVIOR

Do not modify a profile, do not call the functions Configure, SetCoord, or SetInterpType while following is on in a Follower Group which uses the CamProfile.

Failure to follow these instructions can result in injury or equipment damage.

NOTE: In some special situations, the SetCoord function may be called to change the profile while following is on. However, you must only change profile points that are separated by two profile point entries from the nearest profile points that are being used by the slave axis. Failure to ensure this restriction will cause undesired slave movement.

Associated Functions

The associated functions of this parameter are:

- GetCoord which sends a read request for the actual co-ordinates of an existing cam profile point,
- SetCoord which sends a write request for the new co-ordinates of an existing cam profile point.

These functions use the WRITE_CMD instruction.

Language objects

The table below introduces the different functions.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	532	WRITE_CMD	Read command
		533		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read: Master position point from the cam profile table.
%MFr.m.c.24	RETURN_CMD_3	-	-	Result of the read: Slave position point from the cam profile table.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: Required point index.
%MFr.m.c.31	PARAM_CMD_3			Value to write: Master position
%MFr.m.c.33	PARAM_CMD_4			Value to write: Slave position
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Cam Profile Parameter: InterpType

Description

This parameter is used to define the interpolation style used to generate the positions which do not have to feature in the profile data.

The cubic interpolation algorithms force the first derivative of the points in the table to be continuous, where the first derivative value at any point in the table matches the slope of a line segment between the previous point and the next point.

CAUTION

UNEXPECTED EQUIPMENT BEHAVIOR

Do not modify a profile, do not call the functions Configure, SetCoord, or SetInterpType while following is on in a FollowerSet which uses the CamProfile.

Failure to follow these instructions can result in injury or equipment damage.

Associated Functions

The associated functions of this parameter are:

- GetInterpType which sends a read request for the actual interpolation style,
- SetInterpType which sends a write request for the new interpolation style.

These functions use the WRITE_CMD instruction.

Associated Language Objects

The table below presents the different language objects linked to the InterpType parameter.

Object	Name	Function code	Instruction type	Description
%MW.r.m.c.26	ACTION_CMD	530	WRITE_CMD	Read command
		531		Write command
%MDr.m.c.20	RETURN_CMD_1	-	-	Result of the read.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: Interpolation Type 0: linear, 1: cubic.
%MW.r.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Cam Profile Parameter: ProfileId

Description

This parameter contains the profile identification.

Associated Functions

The function associated with this parameter is:

- GetProfileId which sends a read request to the position of the profile's slave follower,

This function uses the WRITE_CMD instruction.

Language objects

The table below presents the different language objects linked to the ProfileId parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	528	WRITE_CMD	Read command
%MDr.m.c.27	RETURN_CMD_1	-	-	Result of the read
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Description of the TRF_RECIPE Screen for Cam Profiles

Introduction

This screen, situated at the bottom of the adjustment screen of a cam profile is used to transfer adjustment parameters between the PLC at the servodrive

This function has to be in online mode.

NOTE: For more information about the common parts of the different adjustment screens, see *Description of the SERCOS® Module Adjustment Screens: General*, page 233.

Illustration

The diagram below shows the screen associated with the TRF_RECIPE function.

The screenshot shows a graphical user interface for the TRF_RECIPE function. It is divided into three main sections: Transfer direction, Parameters, and Results.

- Transfer direction:** Contains radio buttons for "Read" and "Write" (with "Write" selected), checkboxes for "Master" table and "Slave" table (both checked), a "Transfer" button, and a "Transfer in progress" radio button.
- Parameters:** Contains the text "Transfer the parameters of the servodrive to the PLC memory:", an "Address table" field with a dropdown menu set to "%MW" and a value of "100", and a "Length" field with a value of "200".
- Results:** Contains a "Transfer error:" field with a value of "0" and a "Length of the table read:" field with a value of "75".

Description

The table below shows the **Direction of transfer** of the TRF_RECIPE zone.

Element	Language word associated	Symbol/Description
Read/Write	%MWr.m.c.10	See TRF_RECIPE Choose the table(s) to be transferred
"Master" table		
"Slave" table		

The table below shows the **Parameters** of the TRF_RECIPE zone.

Element	Language word associated	Description
Address table	%MDr.m.c.11	Two parameters may be modified: the type of object memory for storing data: ● %MW in read write ● %KW in read only
	Immediate value	the address of the beginning of the table
Length	Immediate value	Table length

The table below shows the **Result** of the TRF_RECIPE zone.

Element	Language word associated	Description
Error transfer	%MWr.m.c.3	This field provides the error code (<i>see page 376</i>) in the case of an invalid transfer.
Length of the table read	%MDr.m.c.4	Real length of the table

TRF_RECIPE Instruction

Reminder

This instruction enables you to read or write the cam profiles and to begin the execution of special functions.

Note: The given parameter length of TRF_RECIPE defines a number of words (double or floating) whatever the type of information exchanged.

NOTE: This service is also used to read or write the variable speed controller parameters with the "Real axis" function, (See *TRF_RECIPE*, page 244).

NOTE: Parameter exchanges (READ_PARAM, WRITE_PARAM, SAVE_PARAM, RESTORE_PARAM) cannot be performed at the same time as the TRF_RECIPE instruction.

Syntax of TRF_RECIPE instruction

TRF_RECIPE (IODDT_VAR, length, address %MW): transfers the cam profile parameters from the table of %MW words. The length of this table to be transferred is defined by the length parameter. The action to be carried out is defined by the word %MWr.m.c.10 (ACTION_TRF).

Example: TRF_RECIPE (IODDT_VAR, 100,200) reads the "Slave" table of cam profile 25 of the module located in position 4 of rack 1, then transfers this to the PLC, from address 200. For an IODDT_VAR variable declared as type T_CSY_CAM and associated with channel 25 of module 4 from rack 1.

TRF_RECIPE Interface

The command to be carried out is defined in the word %MWr.m.c.10 and the result of the command is available in the word %MWr.m.c.3 to %MWr.m.c.8.

Address	Type	Symbol	Meaning
%MWr.m.c.10	Word	ACTION_TRF	Action to be carried out
%MWr.m.c.3	Word	ERROR_TRF	Write error of the TRF_RECIPE command
%MDr.m.c.4	Double word	RETURN_TRF_1	Return 1 of the function
%MFr.m.c.6	Floating point	RETURN_TRF_2	Return 2 of the function
%MFr.m.c.8	Floating point	RETURN_TRF_3	Return 3 of the function
%MDr.m.c.11	Double word	PARAM_TRF_1	Parameter 1

Address	Type	Symbol	Meaning
%MDr.m.c.13	Double word	PARAM_TRF_2	Parameter 2
%MFr.m.c.15	Floating point	PARAM_TRF_3	Parameter 3
%MFr.m.c.17	Floating point	PARAM_TRF_4	Parameter 4

Actions Carried Out by TRF_RECIPÉ

The actions that can be carried out using the TRF_RECIPÉ service are:

Function	ACTION_TRF (%MWr.m.c.10)	Meaning
Cam profile	1	Reading the "master" table.
Cam profile	2	Reading the "slave" table.
Cam profile	3	Reading the "master" and "slave" tables.
Cam profile	101	Writing the "master" table.
Cam profile	102	Writing the "slave" table.
Cam profile	103	Writing the "master" and "slave" tables.

Legend

Writing the drive into the module	PARAM_TRF_1 = 0. On MW
Writing the module into the drive	PARAM_TRF_1 = 1. On KW
Reading	PARAM_TRF_1 = 0.
	PARAM_TRF_2 = OFFSET
	PARAM_TRF_3 and PARAM_TRF_4 = 0
	RETURN_TRF_1(%MDr.m.c.4)=length of the table read or written
	RETURN_TRF_2 and RETURN_TRF_3 = 0

Monitoring the Exchange

The following 2 bits can be used to monitor the exchanges carried out with the help of TRF_RECIPÉ instruction:

Bit	Meaning
%MWr.m.c.0.3	Exchange in process.
%MWr.m.c.1.3	Incorrect exchange.

13.5 Sercos® ring parameters

Subject of this Section

This section introduces the adjustment screen of the SERCOS® ring and describes the associated adjustment parameters.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description of the Adjustment Screen of a SERCOS® Module: SERCOS® ring	307
Adjustment graph for optical power	310
Sercos® ring parameter: OpticalPower	311

Description of the Adjustment Screen of a SERCOS® Module: SERCOS® ring

Introduction

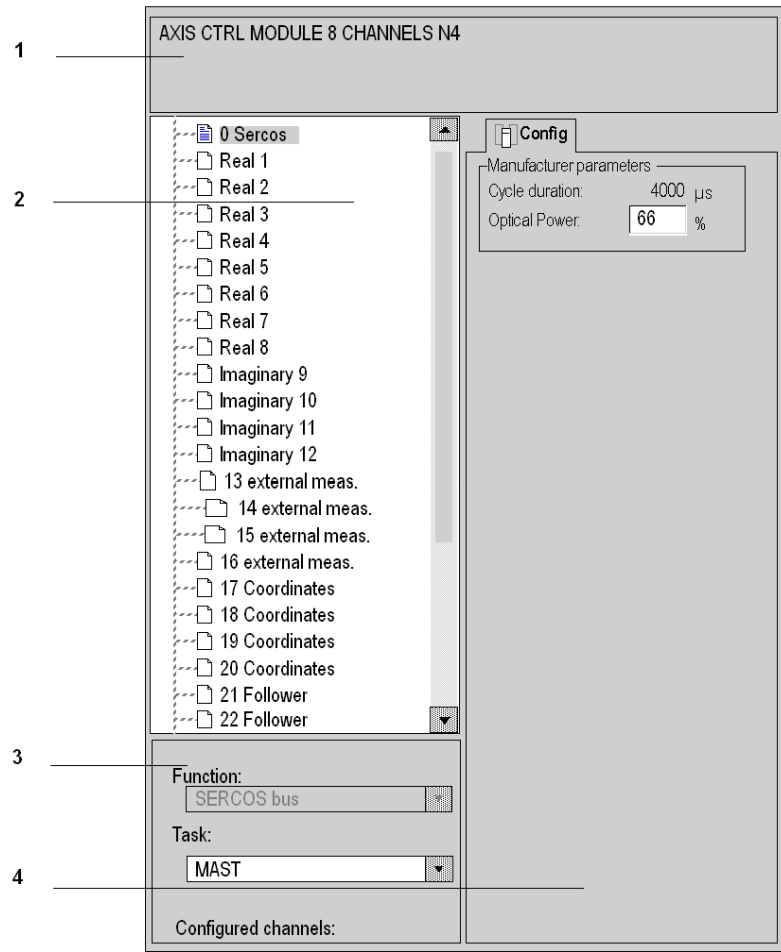
This screen is used to directly modify the value of the optic power in offline or online mode.

It recalls the value of the current ring cycle time.

NOTE: For more information about the common parts of the different adjustment screens, see *Description of the SERCOS® Module Adjustment Screens: General*, page 233.

Illustration

The following figure shows the adjustment zone of the SERCOS® ring on a TSX CSY 84 module (the configuration is the same on a TSX CSY 164 module):



Description

Screen description

Field	Description
1	This band gives the catalog reference of the module and its geographical address in the PLC (rack number and position in the rack).
2	This field known as the Channel field is used to select the channel to be configured.
3	This field known as the General parameters field is used to configure the general parameters associated with the channel selected.
4	This field depends on the tab selected: in this case, it is the configuration zone for the channel selected in zone 2. Only the configuration tab is available in local mode.

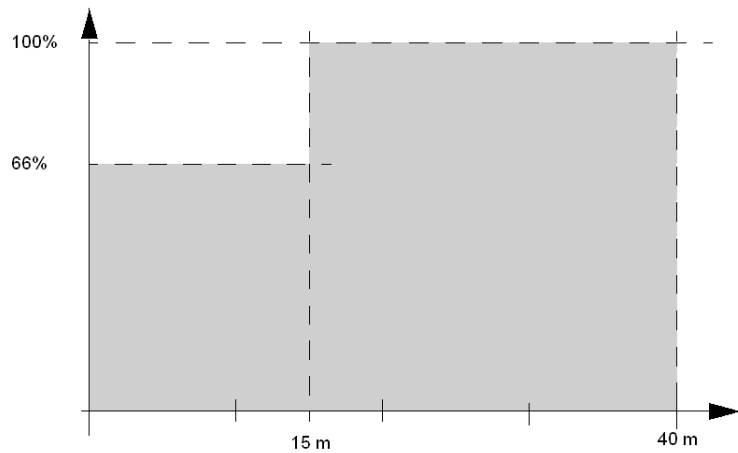
Tab description **Configuration**

Element	Associated parameter	Bit/ word associated language	Symbol/Description
Cycle time	-	-	Value of the current ring cycle time.
Optical power	-	MWr.m.c.37	Optical power adjustment

Adjustment graph for optical power

Optical power

The optical power for the TSX CSY 84 / 164 modules, given as a percentage, is a function of the length of the first segment (between the TSX CSY 84 module or the TSX CSY 164 module and the first drive.) The configuration or adjustment of the optical power must conform to the following graph:



Sercos® ring parameter: OpticalPower

Description

This parameter enables you to set the optical power (light output) of the SERCOS® loop transmitter (LED).

The value is given as a percentage, with an actual resolution of change in power of 20%.

The minimum non-0 optical power is supplied at 66% percent. Full power is supplied at 100%. The LED output is cut if the value is 0%.

NOTE: The default value for the optical power is 66% percent. This value normally works with all SERCOS® drives and all cable lengths. It is possible that the optical power setting may need to be reduced if the length of optical fiber from between the movement controller and the first drive is extremely short. In this case, reducing the optical power may prevent overdriving the drive's receiver LED circuitry.

Associated Functions

The associated functions of this parameter are:

- GetOpticalPower which sends a read request for the actual optical power of the transmitter,
- SetOpticalPower, which sends a write request for the new optical power of the transmitter.

These functions use, according to the language objects, the WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

Associated Language Objects

The table below presents the different language objects linked to the OpticalPower parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	1547	WRITE_CMD	Read command
		2547		Write command
%MFr.m.c.22	RETURN_CMD_2	-	-	Result of the read.
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

13.6 SERCOS® IDN Read/Write Functions

Title of this Section

This section describes the read/write functions for SERCOS® IDs and lists the standard IDs.

What Is in This Section?

This section contains the following topics:

Topic	Page
Servodrive Parameters: IDN_S and IDN_P	313
Servodrive Parameters: IDN_US and IDN_UP	315
Standard SERCOS Identification Numbers (IDN)	317
IDNs for customized telegrams	320

Servodrive Parameters: IDN_S and IDN_P

Description

These parameters enable you to directly control the servodrives by the project (independently of the TSX CSY 84 module).

They can be divided into two categories:

- IDN_S deemed standards (SERCOS® standard), See (*Standard SERCOS Identification Numbers (IDN)*, page 317),
- IDN_P deemed owners (specific to drive constructors, see information on servodrives).

NOTE: This manufacturer information should specify:

- the size of the parameter (16 or 32 bit),
- whether the parameter is signed or unsigned,
- the SERCOS® loop phase in which the parameter is authorized,
- the legal values of the parameter,
- what type of scaling (if any) must be used to interpret the data.

Associated Functions

The associated functions of this parameter are:

- GetIDN_S and GetIDN_P, which send a read request of the current parameters with SERCOS® identification numbers (IDN)). These parameters are read via the axis control module.
- SetIDN_S and SetIDN_P which send a write request for the new parameters with SERCOS® identification numbers (IDN)).

NOTE: (IDN) SERCOS®: S-0-xxxx (standard parameters) and P-0-xxxx (product specific parameters) respectively

These functions can only be used to read or write 16-bit or 32-bit parameters to the SERCOS® servodrive. SERCOS® variable length text parameters cannot be accessed using these functions.

When read, the 16 and 32-bit parameters are returned in the form of 32-bit values. The 16-bit SERCOS® IDN values are extended to a 32-bit returned value based on the type of the 16-bit SERCOS® IDN.

For example, if the 16-bit SERCOS® parameter is a signed parameter, then the 16-bit value is sign-extended to a 32-bit signed value. Equally, if the 16-bit SERCOS® parameter is an unsigned parameter, then the 16-bit value is extended by placing zeroes in the uppermost 16 bits of the 32-bit returned value.

NOTE: Some SERCOS® IDN parameters have units and scale factors associated with them. In addition, these functions do not allow the parameters to be converted (values in axis default units into drive units or vice versa). (Therefore use the IDN_US and IDN_UP functions)

These functions use the WRITE_CMD instruction.

Language Objects

The table below presents the different language objects linked to the OpticalPower parameter.

Object	Name	Function code	Instruction type	Description
%MWr.m.c.26	ACTION_CMD	1556	WRITE_CMD	Read command IDN_S
%MWr.m.c.26		2556		Write command IDN_S
		1557		Read command IDN_P
		2557		Write command IDN_P
%MDr.m.c.20	RETURN_CMD_1	-	-	Result of the read.
%MDr.m.c.27	PARAM_CMD_1	-	-	Value to write: IDNSercos. Format type of 1 to 4095. However, use of the bits 12 to 14 is allowed. These bits indicate which parameter sets are being selected. (See your servodrive manual to find out what parameter sets are supported
%MDr.m.c.29	PARAM_CMD_2	-	-	Value to write to the servodrive
%MFr.m.c.31	PARAM_CMD_3	-	-	Value to write: AddressSercos. This parameter is required if you are using the GetIDN function on the BusSERCOSaxisID 999.
%MWr.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)

Read command IDN_S

The current parameters with SERCOS identification number (IDN) are read by the Read command via the axis control module. The following example illustrates the reading of IDN100 value using this command and all IDN with 2 -insigned byte length.

Example:

The IDN100 is 2-unsigned byte length. The PLC register (Read command) puts unsigned values in signed ones.

If the IDN100 value is higher than 32767, the PLC register will be returned as follows:

- 32768 -> -32768
- 32769 -> -32767
- 35000 -> -30536

When the Read command returns a negative value, it may be converted to its positive equivalent by adding 65536.

Servodrive Parameters: IDN_US and IDN_UP

Description

These parameters are associated with units and scale factors.

They enable you to directly control the servodrives by the project (independently of the TSX CSY 84 module).

They can be divided into two categories:

- IDN_US deemed user standards (Standard SERCOS®), See (*Standard SERCOS Identification Numbers (IDN)*, page 317),
- IDN_UP deemed user owners (specific to servodrive constructors, see information on drives).

NOTE: This manufacturer information should specify:

- whether the parameter has units,
- the SERCOS® loop phase in which the parameter is authorized,
- the legal values of the parameter,

The information in the servodrive manual on how the parameter is scaled and how the units are determined is not applicable in this case. The movement controller will convert the parameter using the information supplied by the servodrive.

Associated Functions

The associated functions of this parameter are:

- GetIDN_US and GetIDN_UP which send a read request of the current parameters with the identification numbers (IDN) SERCOS®) in the form of a floating point number which has been converted from drive units into default axis units.
- SetIDN_US and SetIDN_UP which send a write request of the new parameters with the SERCOS® identification numbers (IDN)) in the form of a floating point number expressed in default axis units for this parameter. The movement controller converts this data value from default axis units to the drive units and writes the converted value to the servodrive.

NOTE: (IDN) SERCOS®: S-0-xxxx (standard parameters) and P-0-xxxx (product specific parameters) respectively

NOTE: These functions do not let you access SERCOS® parameters which do not have units or variable length text parameters.

NOTE: These parameters can also be read/written by the Get/SetIDN_S or Get/SetIDN_P functions, so it is necessary to watch the units that are used.

These functions use the WRITE_CMD instruction.

Language Objects

The table below presents the different language objects linked to the OpticalPower parameter.

Object	Name	Function code	Instruction type	Description
%MW.r.m.c.26	ACTION_CMD	1558	WRITE_CMD	Read request IDN_US
		2558	WRITE_CMD	Write request IDN_US
%MW.r.m.c.26	ACTION_CMD	1559	WRITE_CMD	Read request IDN_UP
		2559	WRITE_CMD	Write request IDN_UP
%MFr.m.c.22	RETURN_CMD_2	-	-	Read report: Value (1) of the floating point parameter.
%MDr.m.c.27	PARAM_CMD_1	-	-	IDNSercos. Format type of 1 to 4095. However, use of the bits 12 to 14 is allowed. These bits indicate which parameter sets are being selected. (See your servodrive manual to find out what parameter sets are supported)
%MFr.m.c.31	PARAM_CMD_3	-	-	AddressSercos. This parameter is required if you are using the GetIDN function on the BusSERCOSaxisID 999.
%MFr.m.c.33	PARAM_CMD_4	-	-	Contents of the variable: Value (1) of the floating point parameter.
%MW.r.m.c.19	ERROR_CMD	-	-	Error report. See (<i>WRITE_CMD Command: Programming Errors, page 370</i>)
Legend				
(1)	This value is expressed in default axis units for this type of parameter (e.g. units of position, speed, acceleration, torque and time),			

Standard SERCOS Identification Numbers (IDN)

Position Parameters

The following table lists the IDNs for the position parameters:

IDN	Standard IDN Name
S-0-0032	Primary operating mode
S-0-0053	Position feedback value 2 (external feedback)
S-0-0055	Position polarity parameter
S-0-0116	Resolution of the rotational feedback 1
S-0-0117	Resolution of the rotational feedback 2
S-0-0121	Input revolutions of load gear
S-0-0122	Output revolutions of load gear
S-0-0123	Feed constant
S-0-0206	Drive on delay time
S-0-0207	Drive off delay time
S-0-0256	Multiplication factor 1
S-0-0257	Multiplication factor 2

Speed Parameters

The following table lists the IDNs for the speed parameters:

IDN	Standard IDN Name
S-0-0037	Additive velocity command value
S-0-0040	Velocity feedback value

Torque/Motor Parameters

The following table lists the IDNs for the torque/motor parameters:

IDN	Standard IDN Name
S-0-0084	Torque feedback value
S-0-0092	Bipolar torque limit value
S-0-0111	Motor continuous stall current

Limit Parameters

The following table lists the IDNs for the limit parameters:

IDN	Standard IDN Name
S-0-0038	Positive velocity limit value
S-0-0039	Negative velocity limit value
S-0-0049	Positive position limit value
S-0-0050	Negative position limit value
S-0-0091	Bipolar velocity limit value
S-0-0114	Load limit of the motor
S-0-0136	Positive acceleration limit
S-0-0138	Bipolar acceleration limit value

Diagnostics

The following table lists the IDNs for diagnostics:

IDN	Standard IDN Name
S-0-0011	Class 1 diagnostic (C1D)
S-0-0012	Class 2 diagnostic (C2D)
S-0-0013	Class 3 diagnostic (C3D)
S-0-0041	Homing velocity
S-0-0042	Homing acceleration
S-0-0147	Homing parameter
S-0-0148	Homing command by drive
S-0-0298	Home switch distance
S-0-0400	Home switch
S-0-0403	Position feedback value status

Scaling Parameters

The following table lists the IDNs for the scaling parameters:

IDN	Standard IDN Name
S-0-0044	Velocity data scaling type
S-0-0086	Torque data scaling type
S-0-0093	Torque/effort data scale factor
S-0-0094	Torque/effort data scale exponent
S-0-0160	Acceleration data scaling type

Probe

The following table lists the IDNs for the probe:

IDN	Standard IDN Name
S-0-0130	Probe value positive edge 1
S-0-0131	Probe value negative edge 1
S-0-0169	Probe control parameter
S-0-0170	Probe cycle procedure command
S-0-0179	Probe status

Gains Parameters

The following table lists the IDNs for the gains parameters:

IDN	Standard IDN Name
S-0-0296	Velocity feedforward gain

Drives, Drive Backup and Checksums

The following table lists the IDNs for drives, drive backup and checksums:

IDN	Standard IDN Name
S-0-0192	IDN list of backup operation data
S-0-0271	Drive ID
S-0-0263	Load working memory procedure command
S-0-0264	Backup working memory procedure command

IDNs for customized telegrams

Customized Telegrams

The following table lists the IDNs for customized telegrams:

IDN	Standard IDN Name
S-0-0016	Configuration list of AT cyclic data
S-0-0024	Configuration list of MDT
S-0-0130	Probe 1 positive edge
S-0-0131	Probe 1 negative edge
S-0-0037	Additive velocity command
S-0-0053	Position feedback value 2 (external encoder)
S-0-0084	Torque feedback value
S-0-0185	Length of the configurable data record in the AT
S-0-0187	IDN list of configurable data in the AT
S-0-0188	IDN list of configurable data in the MDT
S-0-0186	Length of the configurable data record in the MDT

Debugging the TSX CSY 84 / 164 modules

14

Aim of This Chapter

This chapter describes the debug functions of a SERCOS® channel.
All procedures for the TSX CSY 84 are identical for the TSX CSY 164 module.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description of the debug screen: General	322
Debug Screen User Interface	325
Debug: Enable	326
Debug: Diagnostics	327
Debug: Movement	330
Debug: Sending commands (Auto mode)	332
Debug: Manual command (Manual mode)	333
Debug: Reference	334
Debug: Following	335
Debug: Drive	336
Debug: Position	337
Module diagnostic	338
Channel Diagnostics	340

Description of the debug screen: General

Introduction

The different debug screens can be used, for all types of channels, to display the state of the inputs and outputs of a module channel as well as any faults.

It also enables you to command language objects (set to 1 or 0, forcing/unforcing a bit, etc.).

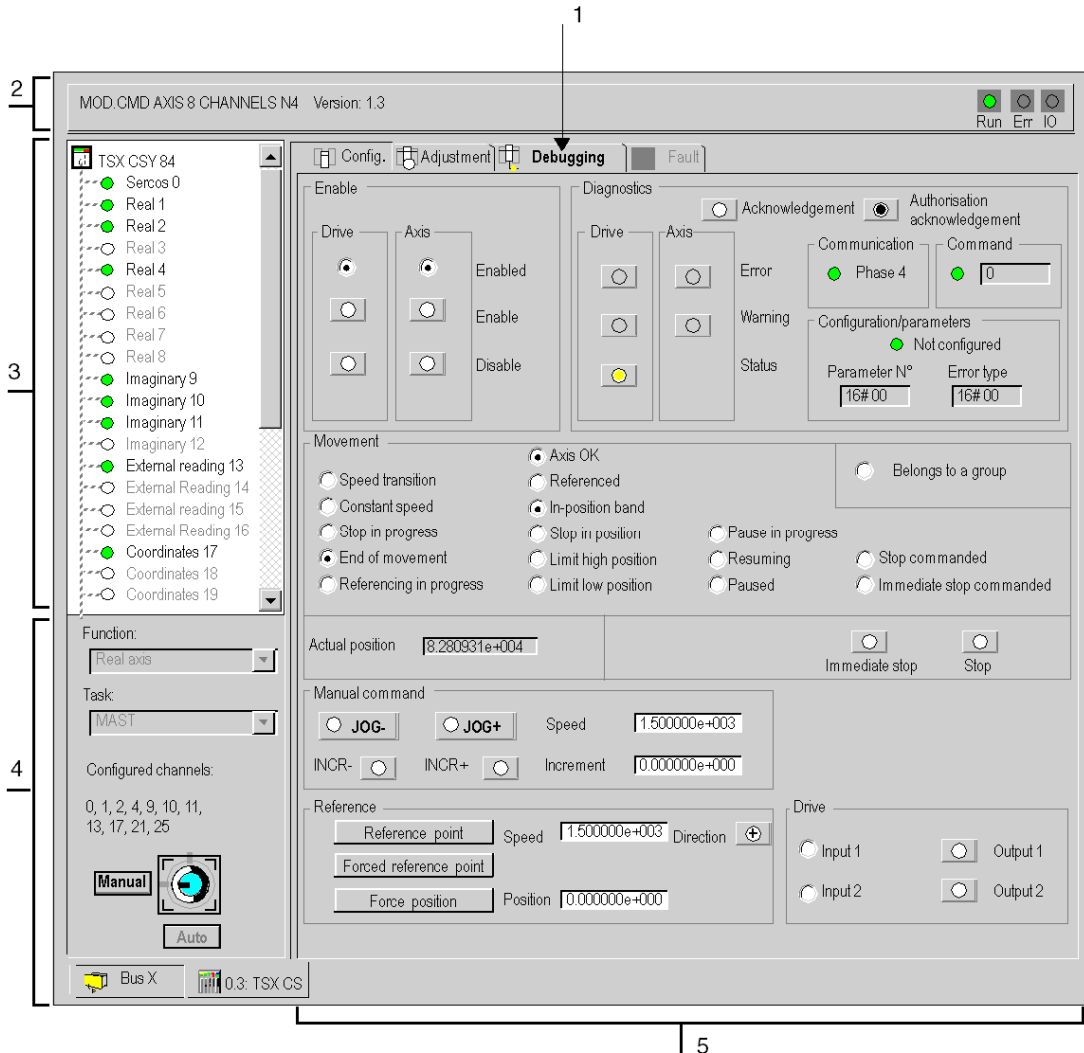
These parameters are accessible in online mode.

It also gives access to the configuration and adjustment screens.

NOTE: For an introduction to the processes of configuring, adjusting and debugging a project, see the section on Common application specific functions.

Illustration

The following figure shows an example of a debug screen for the TSX CSY 84 module (the debug screen for the TSX CSY 164 module is identical):



Description

The table below shows the different elements of the debug screen.

Address	Element	Function
1	Tabs	The tab in the foreground indicates the current mode (Debug in this example). Each mode can be selected using the corresponding tab. Available modes are: ● Debug, accessible only in online mode, ● Diagnostic (Fault), accessible only in online mode, ● Adjustment, ● Configuration.
2	Module zone	Displays the short title of the module. In the same zone there are 3 lamps showing the status of the module in online mode: ● RUN shows the operating mode for the module. ● ERR indicates an internal module error. ● I/O indicates a fault outside the module or an application fault.
3	Channel zone	Allows you to: ● by clicking on the device reference, displays the tabs: ● Description: provides the device characteristics. ● I/O objects (<i>see Unity Pro, Operating Modes</i>): provides a preview symbol of the input/output objects. ● Fault: gives access to the device faults (in online mode). ● chooses the channel, ● display the Symbol, channel name defined by the user (via the variables editor).
4	General parameters zone	Used to force bits and view the counting function: ● Unforce: button used to unforce forced bits. ● Function: recalls the configured counting function. This information cannot be modified. ● Task: displays the MAST or FAST task configured. This information cannot be modified.
5	Current parameters zone	This zone displays the state of the inputs, outputs and different parameters of the counting in progress. If the contents of the counting register are unusable following an input fault, the message or LED Invalid measure displays red.

Debug Screen User Interface

Introduction

The debug screen provides access to the state of the input and output bits through LEDs.

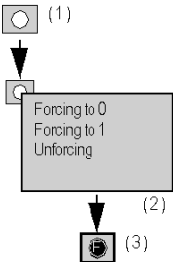
It also enables you to command language objects (%Q) using command buttons

NOTE: Position the mouse cursor over the required LED/button to view the corresponding language object.

The debug screen also enables you to modify and send motion controls.

Command buttons

The table below shows the different possible aspects of command buttons.

Command	Function
	The bit associated with this bit is in position 0. A left click will move the bit to position 1.
	The bit associated with this bit is in position 1. A left click will move the bit to position 0.
	When an object can be forced, a right click on the corresponding button (1) will display a menu (2) which provides access to the force commands: Forcing to 0, Forcing to 1 or Unforcing. Once a command has been selected by clicking on it, the forcing is applied and the forcing state is indicated at the level of the button (3): • F for forcing to 0, • F in inverse video (in the example) for forcing to 1. Note: When a language object is forced, the left mouse button is inoperative
	This button is associated with the error bits currently in position 0.
	This button is associated with the error bits currently in position 1. Left click on this to view the information about the fault.

Debug: Enable

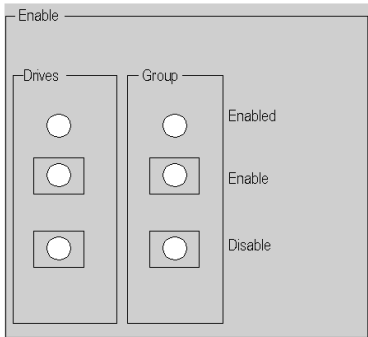
Introduction

This field is used to enable or disable an axis. In order to be effective an axis must be enabled at module level and at drive level.

NOTE: This field concerns real, imaginary and remote axes, as well as groups of co-ordinated and slave axes.

Illustration

The illustration below shows the Enable field of the debug screen.



Description

The table below describes the different **Drive** zone elements of the Enable field.

Element	Associated language object	Description
LED Enabled	See <i>DRIVE_ENABLED Bit</i> (%Ir.m.c.10), page 150.	
Button Enable	%Qr.m.c.10	Enabling the axis at drive level.
Button Disable	%Qr.m.c.26	Disabling the axis at drive level.

The table below describes the different **Axis** zone elements of the Enable field.

Element	Associated language object	Description
LED Enabled		See <i>AXIS_IN_CMD Bit</i> (%Ir.m.c.18), page 158.
Button Enable	%Qr.m.c.2	Enabling the axis at axis level.
Button Disable	%Qr.m.c.18	Disabling the axis at axis level.

Debug: Diagnostics

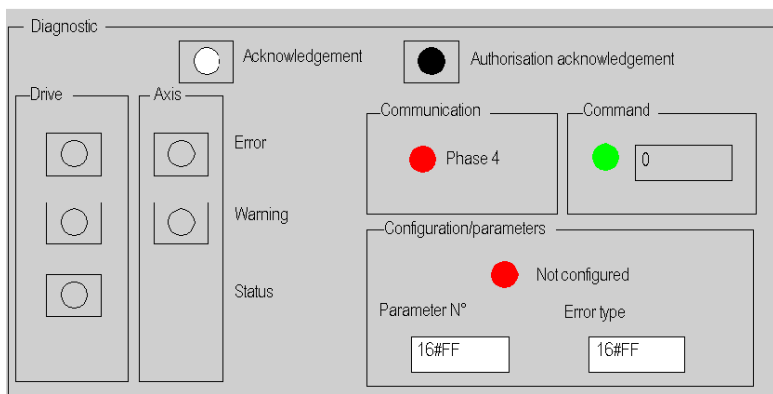
Introduction

This field is used to view and acknowledge the axis faults.

NOTE: This field is concerned with the different types of axes, groups of axes and cam profiles.

Illustration

The illustration below shows the debug screen Diagnostics field.



Description

The table below describes the different elements outside the Diagnostics field zone. These elements are not available in the Cam profile screen.

Element	Associated language object	Description
Acknowledgement Button	%Qr.m.c.15	Fault acknowledgement
Authorisation acknowledgement button	%Qr.m.c.31	Confirm faults

The table below describes the different **Drive** zone elements of the Diagnostics field. These elements are not available in the Cam profile screen.

Element	Description
Error LED Error button	See <i>DRIVE_FAULT Bit</i> (%Ir.m.c.13), page 153. (1)
Warning LED Warning button	See <i>DRIVE_WARNING Bit</i> (%Ir.m.c.12), page 152. (1)
Status LED Status Button	See <i>DRIVE_DIAG Bit</i> (%Ir.m.c.11), page 151. (1)
(1): When the LED is on, a left click on the button will open a window displaying the type of fault.	

The table below describes the different **Axis** zone elements of the Diagnostics field. This zone does not exist in the Cam profile screen.

Element	Description
Error LED Error Button	See <i>AXIS_SUMMARY_FLT Bit</i> (%Ir.m.c.15), page 155. (1), Note: The drive fault data is not to be taken into account if there is a SERCOS® communication error.
Warning LED Warning Button	See <i>AXIS_WARNING Bit</i> (%Ir.m.c.23), page 162. (1)
(1): When the LED is on, a left click on the button will open a window displaying the type of fault.	

The table below describes the different elements of the **Communication** zone of the Diagnostics field. This zone does not exist or is unavailable in the Cam profile screen and imaginary axis.

Element	Associated language object	Description
Phase 4 LED	%lr.m.c.16	See <i>AXIS_COMM_OK Bit</i> (%Ir.m.c.16), page 156.

The table below describes the different **Command** zone elements for the Diagnostics field.

Element	Associated language object	Description
LED	%MWr.m.c.1:X2	Faulty explicit command (Action_CMD).
Field	%MWr.m.c.19	Possible values: 0: parameters OK Otherwise an error code Seewrite_cmd:programming errors command chapter (<i>see page 370</i>)

The table below describes the different **Configuration/ Parameterization** zone elements for the Diagnostics field.

Element	Associated language object	Description
LED Not configured	%lr.m.c.32	Channel is configured.
N° parameter Field	%IW2r.m.c.:X0 to X7	Indicates the faulty parameter. See faulty registers chapter (<i>see page 365</i>)
Error type Field	%IW2r.m.c.:X8 to X15	Indicates the type of error. See (<i>see page 362</i>)

Debug: Movement

Introduction

This field is used to view module status bits and to control the movement of the selected axis.

NOTE: It concerns real or imaginary axes as well as co-ordinated axes groups.

Illustration

The illustration below shows the Motion field of the debug screen.

The illustration shows a 'Motion' control panel. It contains several radio buttons for status monitoring: Speed transition, Constant speed, Stop in progress, End of movement, Referencing in progress, Axis OK, Referenced, In-position band, Stop in position, Limit high position, Limit low position, Pause in progress, Resuming, Paused, Stop commanded, and Immediate stop commanded. Below these is an 'Actual position' field displaying '0.000000e+000 mm'. At the bottom are four square buttons with circular icons: 'Resume', 'Pause', 'Immediate stop', and 'Stop'.

Description

The table below describes the different elements of the Motion field.

Element	Associated language object	Description
Speed transitionLED	See <i>RAMPING Bit</i> (%Ir.m.c.0), page 140.	
Constant speedLED	See <i>STEADY Bit</i> (%Ir.m.c.1), page 141.	
Stop in progressLED	See <i>STOPPING Bit</i> (%Ir.m.c.2), page 142.	
End of movementLED	See <i>PROFILE_END Bit</i> (%Ir.m.c.3), page 143.	
Referencing in progressLED	See <i>AXIS_HOMING Bit</i> (%Ir.m.c.5), page 145.	
Axis OK LED (unitary axis only)	See <i>AXIS_READY Bit</i> (%Ir.m.c.31), page 170.	
Referenced (unitary axis only) LED	See <i>AXIS_HOMED Bit</i> (%Ir.m.c.6), page 146.	
In-position bandLED	See <i>IN_POSITION Bit</i> (%Ir.m.c.4), page 144.	
Stop in positionLED	See <i>AXIS_AT_TARGET Bit</i> (%Ir.m.c.20), page 159.	
High limit position LED	See <i>AXIS_POS_LIMIT Bit</i> (%Ir.m.c.21), page 160.	
Low limit position LED	See <i>AXIS_NEG_LIMIT Bit</i> (%Ir.m.c.22), page 161.	
Pause in progressLED	See <i>HOLDING Bit</i> (%Ir.m.c.8), page 148.	

Element	Associated language object	Description
Resume in progress LED	See <i>RESUMING Bit</i> (%Ir.m.c.9), page 149.	
Pause done LED	See <i>AXIS_HOLD Bit</i> (%Ir.m.c.28), page 167.	
Stop commanded LED	See <i>AXIS_HALT Bit</i> (%Ir.m.c.29), page 168.	
Immediate stop commanded LED	See <i>AXIS_FASTSTOP Bit</i> (%Ir.m.c.30), page 169.	
Belongs to a group (unitary axis only) LED	See <i>AXIS_IS_LINKED Bit</i> (%Ir.m.c.17), page 157.	
Resume Button	%Qr.m.c.12	Resumes control after a stop.
Pause Button	%Qr.m.c.28	Temporary axis stop.
Immediate stop LED	%Qr.m.c.30	Emergency stop command.
Stop LED	%Qr.m.c.29	Stop axis.
Current position (unitary axis only) Field	%IFr.m.c.0	Value of current position.

Debug: Sending commands (Auto mode)

Introduction

This field, which is specific to Auto mode, is used to launch a motion control using explicit exchanges.

Illustration

The illustration below shows the Send command field of the debug screen.

The illustration shows a 'Send command' dialog box. At the top, there is a 'Code' field with the value '0' and a 'Command in progress' LED indicator. Below this, there are four rows of input fields: 'Parameter 1' (0), 'Parameter 2' (0), 'Parameter 3' (0.000000e+000), and 'Parameter 4' (0.000000e+000). To the right of these are three 'Feedback' fields: 'Feedback 1' (0), 'Feedback 2' (0.000000e+000), and 'Feedback 3' (0.000000e+000). At the bottom right, there is a 'SEND COMMAND' button.

Description

The table below describes the different elements of the Send command field.

Element	Associated language object	Description
Code field	%MWr.m.c.26	Function code
Command in progress LED	%MWr.m.c.0:X1	(see page 403)
Parameter fields	%MWr.m.c.27 to33 (1) %MWr.m.c.27 to33 (2)	Function parameters
Return N fields	%MDr.m.c.20 %MFr.m.c.22 %MFr.m.c.24	Results of the command
Command button	-	Explicit exchange via WRITE_CMD.
Note: Consult the documentation corresponding to the desired function for more information.		
(1): Real axes, imaginary axes and groups of slave axes.		
(2): Co-ordinated axis groups		

Debug: Manual command (Manual mode)

Introduction

This field, which is specific to manual mode, is used to launch a continuous (JOG) or incremental (INC) motion control.

NOTE: It is concerned with real and imaginary axes.

Illustration

The illustration below shows the Manual Command field of the debug screen.

Manual command

☒ JOG- ☐ JOG+ Speed: 1.000000e+003

☐ INCR- ☐ INCR+ Increment: 0.000000e+000

Description

The table below describes the different elements of the Manual command field.

Element	Associated language object	Description
JOG+	%Qr.m.c.4	See <i>Manual Mode, page 179</i> . The button associated with this command is a push button.
JOG-	%Qr.m.c.5	See <i>Manual Mode, page 179</i> . The button associated with this command is a push button.
INC+	%Qr.m.c.13	See <i>Manual Mode, page 179</i> .
INC-	%Qr.m.c.14	See <i>Manual Mode, page 179</i> .
Speed	-	value configured by default (1/2 of Smax). This can be modified. This one can be modified either by the screen or by the command SetDefaultSpeed (2065) via PARAM_CMD_3.
Increment	%QDr.m.c	desired increment value.

Debug: Reference

Introduction

This field, which is specific to manual mode, can be used to reference an axis to a given position deemed to be "Home".

Illustration

The illustration below shows the Reference field of the debug screen.

The illustration shows a software interface titled "Reference". Inside the interface, there are three buttons stacked vertically on the left: "Reference point", "Forced reference point", and "Force position". To the right of these buttons, there are two input fields. The top one is labeled "Speed" and contains the value "1.000000e+003". The bottom one is labeled "Position" and contains the value "0.000000e+000". To the right of the "Speed" input field, there is a "Direction" button with a circular icon containing a plus and minus sign.

Description

The table below describes the different elements of the Reference field.

Element	Description
Reference point	Sets a homing reference point (see <i>Home Function</i> , page 182).
Speed	Movement speed used to reach home.
Direction	Direction of rotation (positive or negative) used to reach home
Forced reference point	The position of the axis becomes the homing point (see <i>ForcedHomed Function</i> , page 185).
Force position	Modifies the homing point without referencing axis to that point see (<i>SetPosition Function</i> , page 187).
Position	Home position forcing value.

Debug: Following

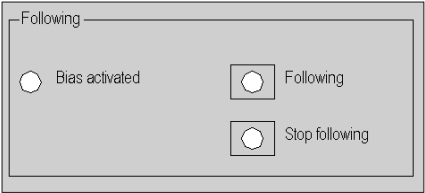
Introduction

This field is used to activate or deactivate the following of slave axes.

NOTE: This field is concerned with real and imaginary axes.

Illustration

The illustration below shows the Following field of the debug screen.



Description

The table below describes the different elements of the Following field.

Element	Associated language object	Description
LED Bias activated	%lr.m.c.24	Offset added to the command position
Button Follow	%Qr.m.c.11	<i>Control Bits Modifiable on Change of State, page 177</i>
Button Stop follow	%Qr.m.c.27	<i>MovementControl Bits Modifiable on Edge, page 176</i>

Debug: Drive

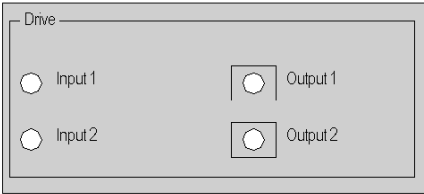
Introduction

This field is used to control drive inputs and outputs. These inputs/outputs are defined by the user.

NOTE: This field only exists in the Real axis screen.

Illustration

The illustration below shows the Drive field of the debug screen.



Description

The table below describes the different elements of the Drive field.

Element	Associated language object	Description
LED Input 1	%I26	See <i>DRIVE_REALTIME_BIT1 Bit</i> (§Ir.m.c.26), page 165
LED Input 2	%I27	See <i>DRIVE_REALTIME_BIT2 Bit</i> (§Ir.m.c.27), page 166
Button Output 1	%Q6	Drive command
Button Output2	%Q7	

Debug: Position

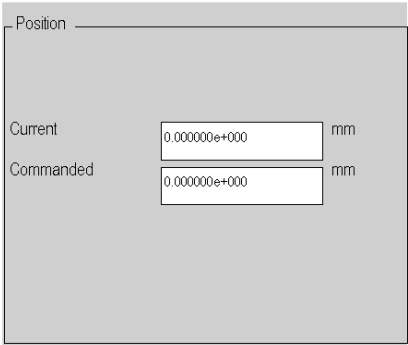
Introduction

This field is used to view the position of an external axis from the project.

NOTE: This field only exists in the Remote axis screen.

Illustration

The illustration below shows the Position field of the debug screen.



Description

The table below describes the different elements of the Position field.

Element	Associated language object	Description
Field Current	%IF0	Value of current position.
Field Commanded	%QD0	Value defined by the project.

Module diagnostic

At a Glance

The Module diagnostic function displays the current faults, if any, classified according to category:

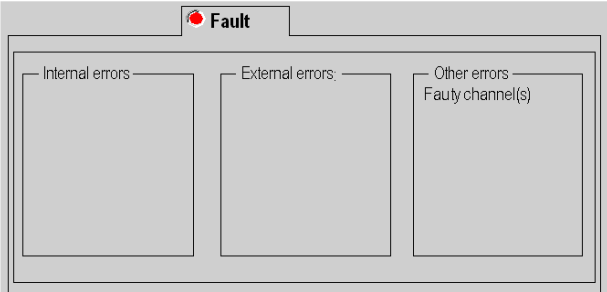
- **internal errors:**
 - module malfunction.
 - autotest in progress
- **external errors:**
 - faulty terminal block,
- **other errors:**
 - configuration fault:
 - module missing or off,
 - faulty channel (*see Premium and Atrium using Unity Pro, Analog input/output modules, User manual*).

A faulty module is shown by certain LEDs turning red, e.g.:

- in the configuration level rack editor:
 - the rack number LED,
 - the module rack location number LED.
- in the module level configuration editor:
 - the **Err** and **I/O** LEDs according to the type of fault,
 - the **Channel** LED in the **Channel** zone,
 - the **Fault** tab.

Procedure

The table below introduces the procedure for accessing the Module Fault screen.

Step	Action
1	Access the "debug module" screen.
2	Click on the module reference number in the channel zone and select the Fault tab. Result: The list of module faults appears.  Please note: The module diagnostic screen is not available in the event of a configuration fault, a major breakdown or a missing module. The following message is then displayed: The module is missing or different to the one configured in this position.

Channel Diagnostics

At a Glance

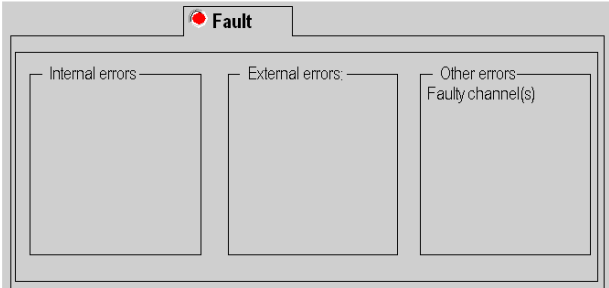
The Diagnostic function channel displays the current faults, if any exist, classified according to their category:

- **internal errors:**
 - channel has broken down.
- **external errors:**
 - faulty link or sensor supply,
- **other errors:**
 - faulty terminal block
 - configuration fault
 - communication fault

A faulty channel is shown by the **Diag** LED, located in the **Err** column of the configuration editor, turning red.

Procedure

The table below introduces the procedure for accessing the Diagnostic channel screen.

Step	Action
1	Access the debug module screen.
2	Click on the Diag button located in the Err column for the channel fault. Result:The list of channel faults appears.  Please note: Diagnostic information about the channel can also be accessed by program (instruction READ_STS).

Subject of this Chapter

This chapter describes the various diagnostic functions, gives the list of errors generated by the axis control module and describes the status bits.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
15.1	SERCOS® Phase Diagnostics Introduction	342
15.2	Reading Actual Data for the Ring Introduction	346
15.3	Axis Identification	350
15.4	Status and fault information	351
15.5	Faults and Warnings	359
15.6	Performances of the TSX CSY 84	382

15.1 SERCOS® Phase Diagnostics Introduction

Subject of this Section

This section describes the diagnostics parameter functions for the SERCOS® phase.

What Is in This Section?

This section contains the following topics:

Topic	Page
About SERCOS® Phases Standard	343
GetActualPhase Function	344
GetCommandedPhase Function	345

About SERCOS® Phases Standard

Standard IEC 1491

The IEC 1491 Standard (the SERCOS®-based international servo drive communications protocol standard) defines the following five phases of communicating with a SERCOS® drive.

SERCOS® phases

The following table defines the phases:

Phase	Meaning
0	The loop is not communicating.
1	The controller is finding the drives which are on the loop.
2	The SERCOS® loop timing parameters are exchanged with the drives.
3	The service channels to the drives are functioning and all timing parameters are being used. The GetIDN and SetIDN functions are ready for operation.
4	The motion controller is sending cyclic communication information to the drives. The loop is up and motion commands are allowed.

GetActualPhase Function

Description

This function returns the actual SERCOS® phase of the loop.

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via WRITE_CMD: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	550	Command
%MDr.m.c.20	RETURN_CMD_1	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

GetCommandedPhase Function

Description

The GetCommandedPhase function returns the SERCOS® loop phase that the movement controller is trying to establish.

NOTE: Use the GetActualPhase function to determine the current loop phase.

The SetCommandedPhase function requests the movement controller to change the SERCOS® loop to the specified phase.

The SERCOS® phase must always switch phases in an increasing manner. If the SetCommandedPhase value is lower than the present loop phase, the movement controller will switch the phase to phase 0 and sequence the loop up to the specified phase. If the loop is at phase 4, and a lower phase is commanded, the movement axes will fault.

NOTE: The movement controller can only progress to the next SERCOS® phase if all the drives on the loop allow a switch to the next phase. Therefore, it is possible that the commanded phase might never be achieved. Use the GetCommandedPhase function to find out which phase was commanded, and use the GetActualPhase function to discover the actual phase of the SERCOS® loop.

How to Use these Functions

These functions are implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via WRITE_CMD: The table below presents the different language objects linked to these functions.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	1545	Read command
		2545	Write command
%MDr.m.c.20	RETURN_CMD_1	-	Result of the read command
%MDr.m.c.27	PARAM_CMD_1	-	Value to write
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

15.2 Reading Actual Data for the Ring Introduction

Subject of this Section

This section describes the functions for reading actual data for the ring (SERCOS® address, number of drives, etc).

What Is in This Section?

This section contains the following topics:

Topic	Page
GetSercosAddress Function	347
GetNumberOfDrivesInRing function	348
IsLoopUp function	349

GetSercosAddress Function

Description

This function returns the SERCOS® address for this axis.

NOTE: This function only exists for real axes and remote axes.

NOTE: The SERCOS® address of an axis is also available in constant word %KW_r.m.c.0 for the corresponding channel.

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via WRITE_CMD: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	549	Command
%MDr.m.c.20	RETURN_CMD_1	-	Result of the command (1)
%MDr.m.c.27	PARAM_CMD_1	-	Value to write (2)
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.
Legend			
(1) If Address = 25350: no axis configured at this address or invalid address.			
(2) : This value corresponds to the index of the table of real axes present in the loop (0 = 1st axis, 1 = 2nd axis, etc.). This parameter is not necessary if the function is issued on the channel corresponding to the axis for which the address is requested.			

GetNumberOfDrivesInRing function

Description

This function returns the number of axes in the fiber-optic SERCOS® network ring.

How to Use this Function

This function is implemented with the help of the `WRITE_CMD` instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

Exchange via `WRITE_CMD`: The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	548	Command
%MDr.m.c.20	RETURN_CMD_1	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

IsLoopUp function

Description

This function signals whether the SERCOS® network ring (SERCOS® ring) is ready to communicate with the movement controller and if it is in phase 4.

Possible values are:

- 1 = (TRUE) if the loop is in "top" status (active communication),
- 0 = (FALSE) if the loop is "low" (phase 4 communication not established).

Use the GetActualPhase function to determine the current loop phase.

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	543	Command
%MDr.m.c.20	RETURN_CMD_1	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

15.3 Axis Identification

GetNumberInSet Function

Description

This function transfers the number of axes contained in a group in the following way:

- SERCOS® function: The function transfers the total number of real axes, imaginary axes, remote axes, co-ordinated axis groups and slave axis groups configured in the project,
- Co-ordinated and slave axis group: the function transfers the number of independent axes contained in the group,
- Real, imaginary and remote axis: the function transfers the value 1, which means that the axis is independent.

How to Use this Function

This function is implemented with the help of the `WRITE_CMD` instruction (see *WRITE_CMD*, page 402).

Associated Language Objects

The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	541	Command
%MDr.m.c.20	PARAM_CMD_1	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors</i> , page 370.

15.4 Status and fault information

Subject of this Section

This section describes the read functions of the fault and status information.

What Is in This Section?

This section contains the following topics:

Topic	Page
Implicit Exchange Objects	352
Faults Accessible via the GetMotionFault Command	353
GetMotionFault Function	355
Faults Accessible via the GetMotionWarning Command	356
GetMotionWarning Function	358

Implicit Exchange Objects

Faulty Module

The %lr.m.MOD.ERR bit at status 1 indicates that the module situated in position r.m is faulty. The causes of the fault are given in the %MWr.m.MOD.2 internal word.

Faulty channel

The bit %lr.m.c.ERR at status 1 indicates that channel i of the module situated in position r.m is faulty. The causes of the fault are given in the %MWr.m.c.2 internal word.

Faults Accessible via the GetMotionFault Command

Description

The MotionFault data indicates the type of error that caused AXIS_SUMMARY_FLT status in the MotionStatus word of an axis. This type of error is recorded by setting a bit in the MotionFault word, one unique bit per type of error. Multiple different errors set their corresponding bits in the MotionFault word. Multiple errors of the same type set the same bit so the MotionFault word does not indicate how many errors of the same kind have occurred. Issuing a ClearFault function to a motion axis clears the MotionFault word.

Motion Faults

The following table lists the motion faults contained in the MotionFault data type:

Name	Bit	Description
MF_MOVE_BUT_NOT_ENABLED	0	A move command has reached the motion interpolator but the drive is disabled. This should only occur if the drive becomes disabled as a move command is starting.
MF_SERCOS_RATE_TOO_FAST	3	Too many channels are configured relative to chosen cycle time.
MF_CONTROL_CONFLICT	10	Control conflict with drive configuration tool.
MF_DRIVE_FAULT	13	A drive fault has occurred. Use the GetIDN_S_ and GetIDN_P function with the SERCOS® Standard IDN S-0-0011 parameter to determine the cause.
MF_REQUESTED_FAULT	15	The MotionControl ALLOW_NOT_FAULT bit is not set. The axis remains faulty until the ALLOW_NOT_FAULT bit is set and a ClearFault function is issued.
MF_COMM_FAULT	16	The SERCOS® communication fiber-optic loop has lost communication to the drive.
MF_AXIS_LIMIT_FAULT	21	The axis has reached a positive or negative software position limit.
MF_PHASE3_CONFIG_PROBLEM	23	Configuration problem phase 3
MF_PHASE0_SERCOS_ERROR	24	Phase 0 error.
MF_PHASE0_SERCOS_ERROR	25	Phase 1 error.
MF_PHASE0_SERCOS_ERROR	26	Phase 2 error.
MF_PHASE0_SERCOS_ERROR	27	Phase 3 error.
MF_PHASE0_SERCOS_ERROR	28	Phase 4 error.
MF_MOVE_WHILE_HALT	29	A move command has reached the interpolator but the MotionControl ALLOW_MOVE bit is not set to 1. This should only happen if the drive is stopped at the exact moment when a move command starts.

Drive Fault

Drive fault data (MF_DRIVE_FAULT) is not to be taken into account if there is also a SERCOS® communication error (MF_COMM_FAULT). In fact, changing to phase 3 triggers a communication error, even though communication is still possible.

GetMotionFault Function

Description

This function transfers the MovementFault information concerning an axis. The MovementFault word (*see page 353*) resets to zero by means of the ClearFault function sent to an axis.

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (*see WRITE_CMD, page 402*).

Associated Language Objects

The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	5510	Command
%MDr.m.c.20	RETURN_CMD_1	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors, page 370</i> .

Faults Accessible via the GetMotionWarning Command

Description

MotionWarnings are generated by the motion controller when a motion command is issued to a MotionSet via the MotionControl registers, and that command is not allowed at the current time. The MotionWarning data type information records these warnings.

The warnings are recorded by setting a bit in the MotionWarning register, one unique bit per type of warning. Multiple warnings of different kinds set their corresponding bits in the MotionWarning word. Multiple warnings of the same type set the same bit so the MotionWarning word does not indicate how many warnings of the same kind have occurred. Issuing a ClearFault function to the axis resets the MotionWarning word to zero. Use the GetMotionWarning function to read the MotionWarnings.

Motion Warnings

The following table lists the motion warnings contained in the MotionWarning data type:

Name	Bit	Description
MW_SURV_WARNING	0	If the 2 axes have a movement differing by more than the alarm threshold, the module triggers a warning on the 2 faulty axes.
MW_SURV_WARNING_PROP	1	If the 2 axes have a movement differing by more than the alarm threshold, the module triggers a fault on the 2 faulty axes, stops all of the list axes and triggers a warning (MW_SURV_WARNING_PROP) on the non-faulty axes.
MW_STOP_BY_SET	2	Fault on axis linked. (The axes that were stopped following propagation of faults will be reported as a warning).
MW_MERGE_UNLINK	7	Motion aborted following deactivation of the sequence.
MW_AXIS_NOT_HOMED	6	Not used.
MW_AXIS_TRAVEL_LIMIT	13	The axis has reached a limit switch or the target position is outside the travel range. Copy of the IDN 12 from the Lexium drive 15. To re-validate a movement, it must first be acknowledge.

Name	Bit	Description
MW_AXIS_IS_LINKED	17	The motion control command was not performed because the motion axis is a member of a CoordinatedGroup or SlaveGroup.
MW_AXIS_POS_LIMIT	21	The axis has exceeded its positive position limit.
MW_AXIS_NEG_LIMIT	22	The axis has exceeded its negative positions limit.
MW_SIMULTANEOUS_MANUAL_CMDS	24	Simultaneous manual commands
MW_AXIS_MANUAL_MODE	25	Not used.
MW_STOP_TO_MANUAL_MODE	26	The manual mode has been activated during the axis movement.
MW_NOTALLOWED_AT_THIS_TIME	31	Command doManualCmd not available: Axis not ready for manual command.

For a Coordinated Group

The following table lists the motion warnings contained in the MotionWarning data type for a CoordinatedGroup:

Name	Bit	Description
MW_AXIS_IS_MOVING	3	The motion control command was not performed because the motion axis was moving.
MW_MEMBER_WARNING	4	Warning on a member. (Enabled on non-enabled groups that have a warning axis will be reported as a warning).
MW_MEMBER_FAULT	5	Fault on group member. (Enabled on non-enabled groups that have a faulty axis will be reported as a warning).
MW_CANNOT_ENABLE	10	The drive will refuse confirmation.
MW_LINK_TARGET_MISSED	11	Groups only: Function DynamicLink (576). Deviation position error between SETPOINT and position actually reached by master and slaves axes.
MW_ACQUIRE_DISALLOWED	18	The CoordinatedGroup or SlaveGroup could not acquire control of its members because one or more of the motion axis members is already a member of a CoordinatedGroup or SlaveGroup.

GetMotionWarning Function

Description

This function transfers the MovementWarning bits (*see page 356*) concerning a movement axis.

These bits give warnings. These are prefixed by MW_.

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (*see WRITE_CMD, page 402*).

Associated Language Objects

The table below presents the different language objects linked to this function.

Object	Name	Function code	Description
%MWr.m.c.26	ACTION_CMD	5511	Command
%MDr.m.c.20	RETURN_CMD_1	-	Result of the command
%MWr.m.c.19	ERROR_CMD	-	Error report. See <i>WRITE_CMD Command: Programming Errors, page 370</i> .

15.5 Faults and Warnings

Title of this Section

This section shows the list of faults and warnings that can be accessed by explicit reading.

What Is in This Section?

This section contains the following topics:

Topic	Page
Module faults which can be accessed by explicit reading	360
Channel faults which can be accessed by explicit reading	361
List of Configuration and Adjustment Errors	362
Faulty Registers	365
List of Error Codes for a WRITE_CMD Command	369
WRITE_CMD Command: Programming Errors	370
WRITE_CMD Command: Communication errors	372
WRITE_CMD Command: System errors	373
WRITE_CMD Command: System warnings	375
TRF_RECIPE Command Explicit Writing Faults	376
Operating modes	377
Logic diagrams of faults	378
Servodrive faults	380

Module faults which can be accessed by explicit reading

Introduction

Word %MWr.m.MOD.2 indicates any fault on the module in position r.m.

List of faults

Bits %MWr.m.MOD.2.0 to %MWr.m.MOD.2.15 are used to diagnose module faults:

Bit	Meaning
0	Internal fault: module inoperative
1	Functional fault: external fault, communication fault or application fault (see the channel status word %MWr.m.c.2)
2	Terminal block fault
3	Module performing self-tests
4	Reserved
5	Configuration fault: hardware and software configuration different
6	Module missing or off
7 to 15	Reserved

Channel faults which can be accessed by explicit reading

Introduction

Word %MWr.m.c.2 indicates any fault on channel i of the module in position r.m.

List of faults

Bits %MWr.m.c.2.0 to %MWr.m.c.2.15 are used to diagnose channel faults:

Bit	Meaning
0	External fault 0: servodrive fault
1	External fault 1: communication fault with the axis
2	Reserved
3	External fault 2
4	Internal fault
5	Configuration fault: hardware and software configuration different
6	Communication fault
7	Faulty project configuration, adjustment or command fault
8	Fan fault (channel 0 only)
9	Overtemperature (channel 0 only)
10	Temperature sensor fault (channel 0 only)
11	Creation of move in progress object
12	Configuration fault (except for channel 0)
13	Reserved
14	Channel LED status: steady
15	Channel LED status: flashing

List of Configuration and Adjustment Errors

Introduction

Word %IW.r.m.c.2 indicates a possible programming fault.

- The least significant byte (LSB) contains the return code of the error type,
- The most significant byte (MSB) contains the address in the registers of the field which caused the error.

List of Faults

The following table gives the content of the LSB (error code, in hexadecimal), the name and the description of the fault:

Error code	Name	Description
1	ERREUR_PLAGE	Attempt to assign value out of range
2	CORRESPONDANCE_UNITES	Attempt to assign incompatible units
3	UNITE_NON_SUPPORTEE	Unit not supported or unknown
4	ERREUR_DE_CHARGEMENT	Servodrive fault while uploading
5	ERREUR_DE_DECHARGEMENT	Servodrive fault while downloading
6	OBJET_VIDE	Unexpected empty pointer to object
7	ERREUR_DEFINITION_UNITES	Failed to set units in servodrive
8	UNITES_NON_DEFINIES	Units not set
9	CHAINE_TROP_Grande	String too large for MovementString
A	INDEX_INCORRECT	Invalid index in a group
B	VALEUR_INCORRECTE	Incorrect value in command
C	VALEUR_ENUM_INCORRECTE	Incorrect value in a sequence
D	JETON_INCORRECT	Invalid token in input
E	VOIE_RETOUT_INCORRECTE	Invalid feedback channel for command
F	PERIPHERIQUE_RETOUT_INCORRECT	Invalid feedback peripheral for a command
10	FREQ_HORLOGE_RETOUT_INCORRECTE	Invalid feedback clocking rate
11	ALIMENTATION_RETOUT_INCORRECTE	Invalid feedback supply
12	RESOLUTION_RETOUT_INCORRECTE	Invalid feedback resolution

Error code	Name	Description
13	ADR_REG_MAINTIEN_INCORRECTE	Invalid holding register address
14	REGISTRES_MAINTIEN_NON_CONFIGURES	Holding register database not configured
15	REGISTRES_MAINTIEN_VIDE	Holding register database empty
16	BLOC_REG_TROP_GRAND	Holding register block too big
17	BLOC_REG_NE_CORRESPOND_PAS	Holding register block does not correspond to the database
18	AUTORISATION_ACCES_IMPOSSIBLE	Cannot grant access to holding register block
19	LIBERATION_ACCES_IMPOSSIBLE	Cannot release access to holding register block
1A	ERREUR_OUVERTURE_FICHER	File open failed
1B	ERREUR_ECRITURE_FICHER	File write failed
1C	ERREUR_LECTURE_FICHER	File read failed
1D	ERREUR_FERMETURE_FICHER	File close failed
1E	ERREUR_RECHERCHE_FICHER	File seek failed
1F	ERREUR_SYNTAXE	Malformed input
20	ERREUR_EFFACEMENT_DEFAULT	Clear fault function failed to clear faults
21	TAG_MANQUANT	Missing tag in tags.cfg
22	AUCUN_AXE_DISPONIBLE	No axis object available
23	AXES_TROP_NOMBREUX	Too many axes in configuration
24	AXES_EN_DOUBLE	Duplicate axes in configuration
25	AXE_INCORRECT	Missing or invalid axis
26	AXE_INTROUVABLE	Axis object or config file not found
27	NOMBRE_COORD_ERRONNE	Value has different number of coordinates from axis
28	PERIPHERIQUE_HORS_COMMANDE	Motion axis not active
29	ERREUR_DEFAULT_MOUVEMENT	A move fault has occurred on the motion controller
2A	AXE_NON_ACTIF	Servodrive is not enabled
2B	DEPASSEMENT_DELAI_COMMANDE	Command timed out
2C	BUS_SERCOS_INCORRECT	Only one SERCOS® bus may be configured
2D	ERREUR_CHANGEMENT_NOM	Axis rename failed

Error code	Name	Description
2E	CMD_IMPOSSIBLE_AVEC_CETTE_CONFIG	Cannot perform command in current configuration
2F	CORRESPONDANCE_TYPE	Object is wrong type
30	VARIATEUR_DOIT_ETRE_DESACTIVE	Servodrive must be deactivated to perform command
31	VARIATEUR_DOIT_ETRE_ACTIVE	Servodrive must be activated to perform command
32	CMD_NON_AUTORISEE	Command not allowed at this time
33	ERREUR_DEFAULT_VARIATEUR	Command cannot be completed due to servodrive fault

Faulty Registers

Faulty Register in the Case of the SERCOS® Channel

The following table gives the content of the MSB (error code, in hexadecimal), and the faulty register, in the case of channel 0:

Error code	Register
0	REG_VERSION_AXE
1	REG_ID_AXE
2	REG_BITS_CONFIGURATION
3	REG_NOMBRE_AXES
4	REG_TEMPS_CYCLE
5	REG_DEBIT
6	REG_PUISSANCE_OPTIQUE
7	REG_MEMBRE_S1

Faulty Register in the Case of an Individual Axis

The following table gives the content of the MSB (error code, in hexadecimal), and the faulty register, in the case of an individual axis:

Error code	Register
0	REG_VERSION_AXE
1	REG_ID_AXE
2	REG_BITS_CONFIGURATION
3	REG_ADRESSE_SERCOS
4	REG_ACCELERATION
6	REG_DECELERATION
8	REG_TYPE_ACCELERATION
9	REG_FENETRE_AU_POINT
B	REG_FENETRE_DE_CONTROLE
D	REG_MODULO_MAX
F	REG_MODULO_MIN
11	REG_ACCELERATION_MAX
13	REG_DECELERATION_MAX
15	REG_VITESSE_MAX
17	REG_POSITION_MAX
19	REG_POSITION_MIN

Error code	Register
1B	REG_NUMERATEUR_FACTEUR_ECHELLE
1D	REG_DENOMINATEUR_FACTEUR_ECHELLE
1F	REG_UNITES_ACCELERATION
20	REG_UNITES_VITESSE
21	REG_UNITES_POSITION
22	REG_RETOUR_RA

Faulty Register in the Case of a Group of Co-ordinated Axes

The following table gives the content of the MSB (error code, in hexadecimal), and the faulty register, in the case of a group of co-ordinated axes:

Error code	Register
0	REG_VERSION_AXE
1	REG_ID_AXE
2	REG_BITS_CONFIGURATION
3	REG_NOMBRE_MEMBRES
4	REG_MEMBRE_C1
5	REG_MEMBRE_C2
6	REG_MEMBRE_C3
7	REG_MEMBRE_C4
8	REG_MEMBRE_C5
9	REG_MEMBRE_C6
A	REG_MEMBRE_C7
B	REG_MEMBRE_C8
C	REG_ACCELERATION
E	REG_DECELERATION
10	REG_TYPE_ACCELERATION
11	REG_ACCELERATION_MAX
13	REG_DECELERATION_MAX
15	REG_VITESSE_MAX
17	REG_UNITES_ACCELERATION
18	REG_UNITES_VITESSE

Faulty Register in the Case of a Group of Slave Axes

The following table gives the content of the MSB (error code, in hexadecimal), and the faulty register, in the case of a group of slave axes:

Error code	Register
0	REG_VERSION_AXE
1	REG_ID_AXE
2	REG_BITS_CONFIGURATION
3	REG_NOMBRE_MEMBRES
4	REG_ID_MAÎTRE
5	REG_MEMBRE_S1
6	REG_MODE_SUIVEUR_S1
7	REG_CAME_OU_NUMERATEUR_S1
9	REG_OFFSET_OU_DENOMINATEUR_S1
B	REG_TRIGGER_S1
D	REG_MEMBRE_S2
E	REG_MODE_SUIVEUR_S2
F	REG_CAME_OU_NUMERATEUR_S2
11	REG_OFFSET_OU_DENOMINATEUR_S2
13	REG_TRIGGER_S2
15	REG_MEMBRE_S3
16	REG_MODE_SUIVEUR_S3
17	REG_CAME_OU_NUMERATEUR_S3
19	REG_OFFSET_OU_DENOMINATEUR_S3
1B	REG_TRIGGER_S3
1D	REG_MEMBRE_S4
1E	REG_MODE_SUIVEUR_S4
1F	REG_CAME_OU_NUMERATEUR_S4
21	REG_OFFSET_OU_DENOMINATEUR_S4
23	REG_TRIGGER_S4
25	REG_MEMBRE_S5
26	REG_MODE_SUIVEUR_S5
28	REG_CAME_OU_NUMERATEUR_S5
2A	REG_OFFSET_OU_DENOMINATEUR_S5
2C	REG_TRIGGER_S5
2E	REG_MEMBRE_S6

Error code	Register
2F	REG_MODE_SUIVEUR_S6
30	REG_CAME_OU_NUMERATEUR_S6
32	REG_OFFSET_OU_DENOMINATEUR_S6
34	REG_TRIGGER_S6
36	REG_MEMBRE_S7
37	REG_MODE_SUIVEUR_S7
38	REG_CAME_OU_NUMERATEUR_S7
3A	REG_OFFSET_OU_DENOMINATEUR_S7
3C	REG_TRIGGER_S7
3E	REG_MEMBRE_S8
3F	REG_MODE_SUIVEUR_S8
40	REG_CAME_OU_NUMERATEUR_S8
42	REG_OFFSET_OU_DENOMINATEUR_S8
44	REG_TRIGGER_S8

List of Error Codes for a WRITE_CMD Command

Introduction

The %MWr.m.c.19 word signals a possible error while explicitly writing a WRITE_CMD command.

Three types of error are generated by the axis control module:

- the programming errors (*see page 370*),
- the communication errors (*see page 372*),
- the system errors (*see page 373*).

WRITE_CMD Command: Programming Errors

List of Programming Errors

The following table lists the error codes, names, and descriptions of programming errors. Error codes 1 to 999 are reserved for this type of error.

Error code	Description
1	Attempt to assign value out of range
2	Attempt to assign incompatible units
3	Unit not supported or unknown
4	Servodrive fault during download
5	Servodrive fault during upload
6	Unexpected empty pointer to object
7	Failed to set units in servodrive
8	Units not set
9	String too large for MotionString
10	Invalid index in a collection
11	Incorrect value in command
12	Incorrect value in a sequence
13	Invalid token in input
14	Invalid feedback channel for command
15	Invalid feedback peripheral for a command
16	Invalid feedback clocking rate
17	Invalid feedback supply
18	Invalid feedback resolution
19	Invalid holding register address
20	Holding register database not configured
24	Cannot grant access to holding register block
25	Cannot release access to holding register block
26	File open failed
27	File write failed
28	File read failed
29	File close failed
30	File seek failed
31	Malformed input
32	Clear fault function failed to clear faults

Error code	Description
33	Missing tag in tags.cfg
34	No axis object available
35	Too many axes in configuration
36	Duplicate axes in configuration
37	Missing or invalid axis
38	Axis object or config file not found
39	Value has different number of coordinates from axis
40	Motion axis not active
41	A motion fault has occurred on the motion controller
42	Servodrive not enabled
43	Command timed out
44	Only one SERCOS® bus may be configured
45	Axis rename failed
46	Cannot perform command in current configuration
47	Object is wrong type
48	Servodrive must be disabled to perform command
49	Servodrive must be enabled to perform command
50	Command not authorized at this time
51	Command cannot be completed due to servodrive fault
60	Manual Mode refused on axis linked to a coordinated or slave group
61	Auto Control refused on axis in Manual Mode
62	Slave group has cam profile slaves
63	Deceleration for a slave or a master exceeds maximum configured deceleration value.
64	TRF_ RECIPE 26200 rejected: Instance already enabled
65	TRF_ RECIPE 26200 rejected: alarm threshold exceeds fault threshold
66	TRF_ RECIPE 26200 rejected: negative threshold
67	TRF_ RECIPE 26200 rejected: nil value denominator.
68	TRF_ RECIPE 16200 rejected: disabled instance

WRITE_CMD Command: Communication errors

List of Communication Errors

The following table lists the error codes, names, and descriptions of communication errors. Error codes 1000 to 1999 are reserved for this type of error.

Error code	Description
1000	Target not responding
1001	Garbled communications
1002	SERCOS® error
1003	No opcode echo from drive
1004	SERCOS® loop not ready
1005	SERCOS® error
1006	SERCOS® read failed (cyclic channel)

WRITE_CMD Command: System errors

List of System Errors

The following table lists the error codes, names, and descriptions of system errors. Error codes 2000 to 2999 are reserved for this type of error.

Error code	Description
2000	Unclassified error
2001	Error in the components
2002	Error in the components
2003	System fault used in debugging
2004	Drive got Command-Not-Allowed fault
2005	Drive got illegal command fault
2006	Drive got Programming Error fault
2007	Invalid command ID
2008	Failed to link object to drive interface
2009	Semaphore create failed
2010	Semaphore delete failed
2011	Semaphore lock failed
2012	Semaphore unlock failed
2013	Semaphore inquiry failed
2014	The motion axis was never successfully configured.
2015	Command not implemented yet
2016	scx_unique failed
2017	Queue create failed
2018	Queue is full
2019	Invalid queue ID
2020	Unknown queue status
2021	Queue inquiry failed
2022	Event group create failed
2023	Event group pend error
2024	Event group clear error
2025	Event flags create failed
2026	Object lookup failed
2027	Object manager not found
2028	Invalid motion planner state

Error code	Description
2029	Memory allocation failed
2030	Error occurred upon attempting to get task ID from the operating system
2031	Attempt to install too many error handlers
2032	Thread create failed
2033	Thread destructor failed
2034	Problem with configuring a thread has occurred
2035	Error occurred while attempting to suspend a thread
2036	Error occurred while attempting to resume a thread
2037	Error occurred while creating object
4000	RMS current fault
4001	Drive overtemperature condition
4002	Drive overtemperature condition
4005	Resolver or encoder feedback fault
4006	General drive fault (phase error)
4007	Drive short circuit fault
4009	Drive voltage fault
4011	Following error fault
4012	Drive has detected communications fault
4013	Hardware end of travel fault
4015	Homing, digital output or control conflict fault (from 2 sources)
4016	SERCOS® master has detected communication fault
5001	The watchdog expired and all axes disabled

WRITE_CMD Command: System warnings

List of system warnings

The following table lists the error codes, names, and descriptions of system warnings.

Error code	Description
7001	The program does not recognize this subroutine number
7002	The axis ID is not valid for this subroutine
7003	Parameter data is out of range
7004	SubNum/SubNumEcho protocol error
7005	The subroutine does not complete in time
7010	The program does not recognize this subroutine number
7011	Wrong value entered within the command
7777	A call was made to the Sample_User subroutine

TRF_RECIPE Command Explicit Writing Faults

Introduction

The %MW.r.m.c.3 word signals a possible error while explicitly writing a TRF_RECIPE command.

The faults generated are the same as WRITE_CMD command explicit writing faults (see previous pages), with the addition of a few faults specific to the TRF_RECIPE command.

List of Errors Specific to the TRF_RECIPE Command

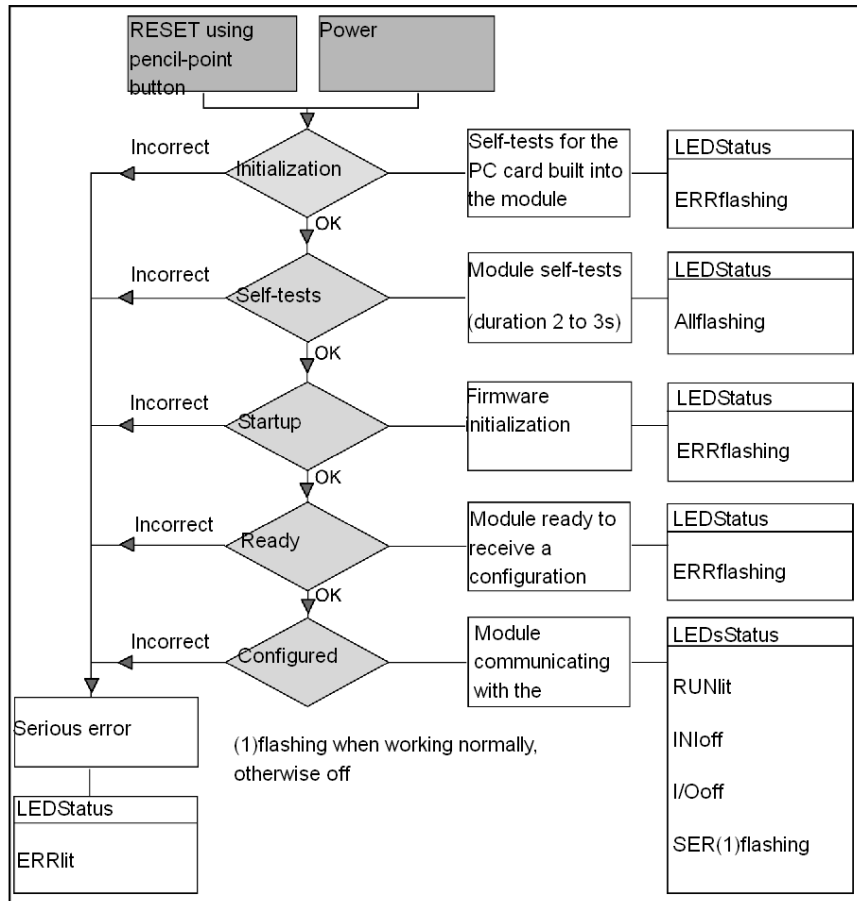
The following table gives the error code, the name and the description of the errors specific to the TRF_RECIPE command:

Error code	Name	Description
6	TABLE_DOES_NOT_EXIST	The CamProfile is configured to be fixed, not variable. A CamProfile table cannot be read or written.
19	PLC_CAPACITY_EXCEEDED	The transfer requested exceeds the capacity of the %MW or %KW of the PLC.
22	LONGUEUR_INSUFFISANTE	The table associated with the TRF_RECIPE is not large enough for the data to be exchanged.

Operating modes

Overview of the Operating Modes

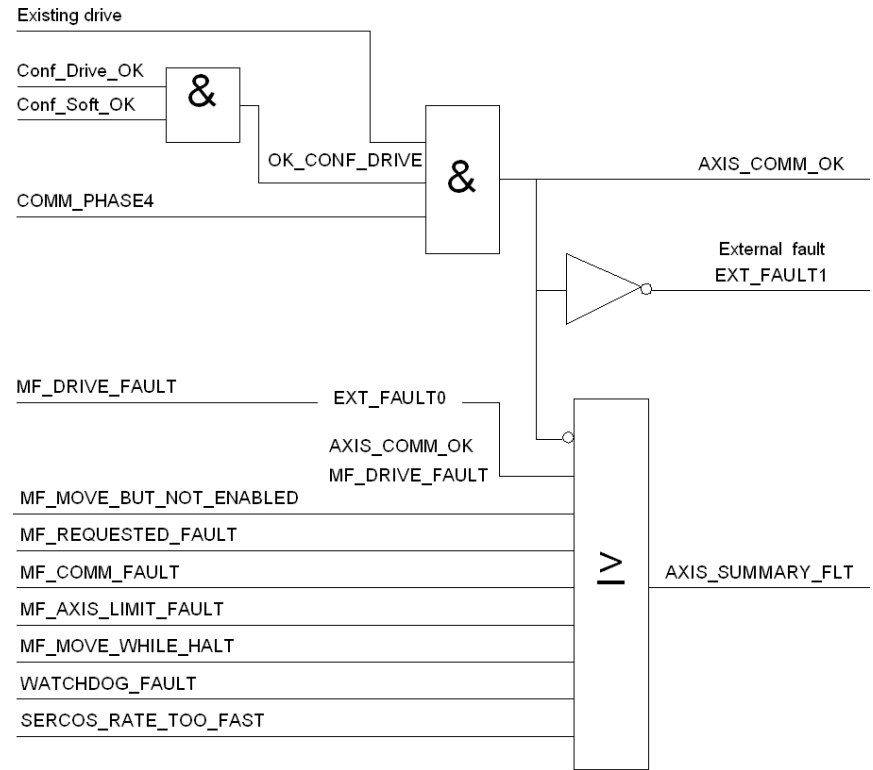
The operating modes for the TSX CSY 84 / 164 modules are as follows:



Logic diagrams of faults

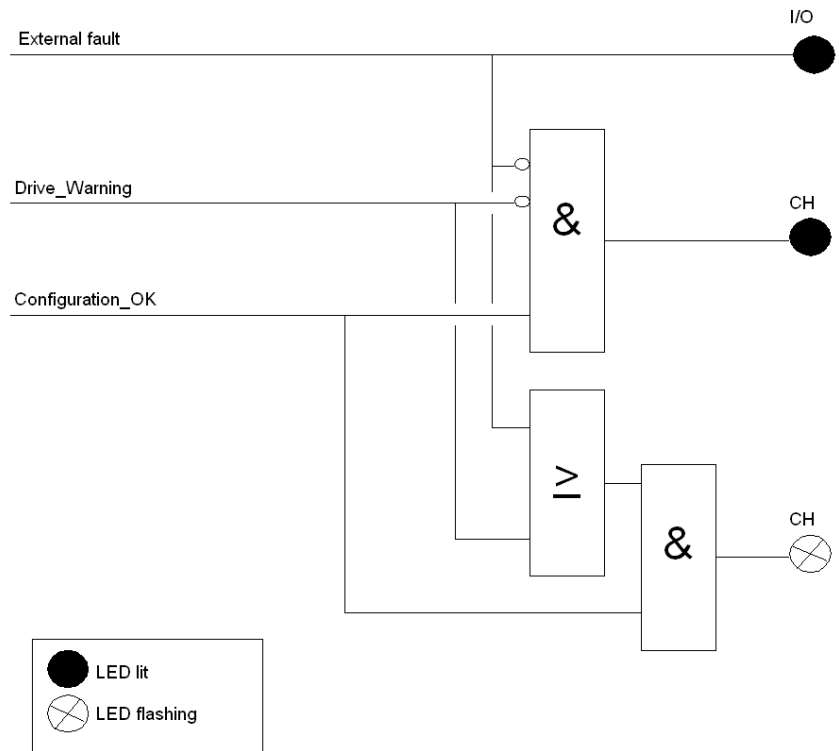
Fault Generation

The following logic diagram illustrates the generation of AXIS_SUMMARY_FLT and external faults:



Activation of LEDs

The following logic diagram illustrates the activation of the I/O and CH LEDs:



Servodrive faults

Introduction

Certain functions can be used upload the fault information to the project.

These functions are:

- GET_IDN for IDN_11,
- GET_IDN for IDN_12,
- GET_IDN for IDN_13.

NOTE: For more information on GET_IDN functions, see (*SERCOS® IDN Read/Write Functions, page 312*).

"IDN11" faults

The faults below are a result of the execution of the GET_IDN function for IDN_11.

Bit	Description
0	Stop due to overload (IDN 114)
1	Stop temperature of the amplifier (IDN 203)
2	Stop temperature of the motor (IDN 204)
3	Stop due to faulty cooling (IDN 205)
4	Command voltage error
5	Feedback error
6	Error in the communication system
7	Over intensity error
8	Over-voltage error
9	Under voltage error
10	Electric supply phase error
11	Excessive position deviation (IDN 159)
12	Transmission error (IDN 14)
13	Limits exceeded (IDN 49 and 50)
14	Reserved
15	Manufacturer specific error (IDN 129)

"IDN11" faults

The faults below are a result of the execution of the GET_IDN function for IDN_12.

Bit	Description
0	Overload alarm (IDN 314)
1	Amplifier overheated alarm (IDN 311)
2	Motor overheating alarm (IDN 312)
3	Alarm due to faulty cooling (IDN 313)
4 to 14	Reserved
15	Manufacturer specific alarm (IDN 129)

"IDN13" faults

The faults below are a result of the execution of the GET_IDN function for IDN_13.

Bit	Description
0	N return status = N command (IDN 330)
1	Status of [absolute value of return N] = 0 (IDN 331)
2	N return status < [absolute value of Nx] (IDN 332)
3	Status of [absolute value of T] >= [absolute value of Tx] (IDN 333)
4	Status of [absolute value of T] >= [absolute limit value of T] (IDN 334)
5	Status of [absolute value of N command] >= [absolute limit value of N] (IDN 335)
6	Status in position (IDN 336)
7	Status of [absolute value of P] <= [absolute value of Px] (IDN 337)
8	Reserved
9	Status of [absolute value of real N] >= maximum spindle speed (IDN 339)
10	Status of [absolute value of real N] >= maximum spindle speed (IDN 340)
11 to 14	Reserved
15	Manufacturer specific alarm (IDN 129)

15.6 Performances of the TSX CSY 84

Performances of the TSX CSY 84

At a Glance

The performances of the module are:

- SERCOS® bus cycle time:
 - SERCOS® tick equals 4 ms by default.
- Processing of an immediate command = 2 SERCOS® ticks
- Processing of a queued command = 2 SERCOS® ticks * Number of queued commands
- Processing of Remote measurement = 2 Sercos ticks + 1 SERCOS® tick for acknowledgment of the reference command on the slave
- Monitoring period of the deviations between axes: 20 ms independent of the Tick,
- Modification on the fly of the cycle time via the Write-cycle command to be chosen from the following:
 - SERCOS® Tick at 2 ms,
 - SERCOS® Tick at 3 ms,
 - SERCOS® Tick at 4 ms,

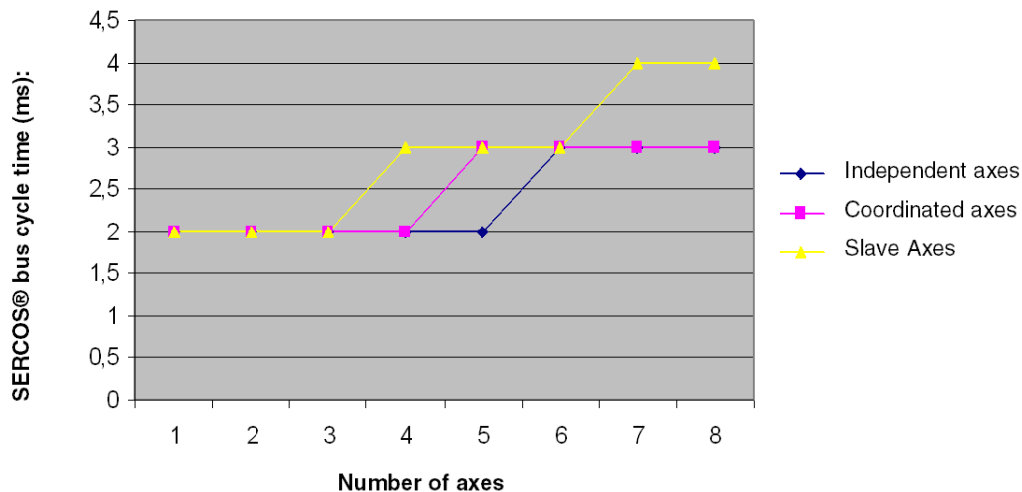
with acknowledgement during forcing in phase 0 of the SERCOS® ring.

The SERCOS® Tick is defined according to the number of axes configured and the type of applications (See the SERCOS® Cycle Time Selection Guide).

NOTE: The Lexium carries out an interpolation every 250 µs.

SERCOS® Cycle Time Selection Guide:

Performances of the TSX CSY 84

**Test conditions:**

NOTE: The performances shown were achieved under certain conditions guaranteeing a time buffer of approximately 25% of the SERCOS® CPU time. The Period of the MAST task is configured at 20ms. Transmission of the motion commands via MOVE_QUEUE is carried out once every 10 PLC cycles.

As a consequence,

- for the "independent axes" configuration: a MOVE_QUEUE command is sent to all axes every 200ms
- For the "slave axes" configuration, a MOVE_QUEUE command is sent to the master every 200ms
- For the "coordinated axes" configuration, a MOVE_QUEUE command is sent to all groups of coordinated axes every 200ms.

A WRITE_PARAM command is sent during this time period, in order to carry out a parameter adjustment.

Modification of Cycle Time using a Program

```
(* change of the SERCOS® cycle time => %MW30 = 2000 or 3000 or
4000 *)
IF %M30 THEN      %MW101.0.26:=2565;
    %MD101.0.27:=%MW30;
    IF NOT %MW101.0:X1 THEN
        WRITE_CMD %CH101.0;
        RESET %M30;
        SET %M31;
    END_IF;
END_IF;

(* transition into SERCOS® phase 0 *)
IF %M31 THEN      %MW101.0.26:=2545;
    %MD101.0.27:=0;
    IF NOT %MW101.0:X1 THEN
        WRITE_CMD %CH101.0;
        RESET %M31;
        SET %M32;
    END_IF;
END_IF;

(* New transition into SERCOS® phase 4 *)
IF %M32 THEN      %MW101.0.26:=2545;
    %MD101.0.27:=4;
    IF NOT %MW101.0:X1 THEN
        WRITE_CMD %CH101.0;
        RESET %M32;
    END_IF;
END_IF;

(* Here, we can expect transition to phase 4 (OPCODE 550 result
in %MD101.0.20) then acknowledge the fault caused by the
transition to phase 0 *)
```

Subject of this Chapter

This chapter describes the language objects associated with SERCOS modules.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
16.1	The language objects and IODDTs of the SERCOS module	386
16.2	Language Objects and IODDTs Associated Specifically with the SERCOS Module	408
16.3	The IODDT Type T_GEN_MOD Applicable to All Modules	450

16.1 The language objects and IODDTs of the SERCOS module

Subject of this Section

This section covers general points relating to the language objects and IODDTs of the SERCOS module.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introduction to IODDTs associated with TSX CSY 84 modules	387
Implicit Exchange Language Objects Associated with the Application-Specific Function	388
Explicit Exchange Language Objects Associated with the Application-Specific Function	389
Management of Exchanges and Reports with Explicit Objects	391
Interface Language	396
Parameter Management	398
WRITE_PARAM and READ_PARAM	400
SAVE_PARAM and RESTORE_PARAM	401
WRITE_CMD	402
WRITE_CMD Examples	404
READ_STS	406
TRF_RECIPe and MOD_PARAM	407

Introduction to IODDTs associated with TSX CSY 84 modules

General

The IODDTs are predefined by the manufacturer. They contain input/output language objects belonging to a channel of an application-specific module.

The TSX CSY 84 module has 7 types of IODDT:

- T_CSY_CMD,
- T_CSY_RING,
- T_CSY_TRF,
- T_CSY_IND,
- T_CSY_FOLLOW,
- T_CSY_COORD,
- T_CSY_CAM.

NOTE: There are two ways to create an IODDT variable:

- **I/O Objects tab** (see *Unity Pro, Operating Modes*),
- data editor (see *Unity Pro, Operating Modes*).

Language object types

Each IODDT contains a set of language objects which can be used to control and check how they operate.

There are two types of language object:

- **implicit exchange objects**, which are exchanged automatically for every cycle of the task associated with the module,
- **explicit exchange objects**, which are exchanged at the request of the project, by using explicit exchange instructions.

Implicit exchanges are concerned with the status of modules, communication signals, slaves, etc.

Explicit exchanges are used to set the parameters of the module and to perform module diagnostics.

Implicit Exchange Language Objects Associated with the Application-Specific Function

At a Glance

An integrated application-specific interface or the addition of a module automatically enhances the language objects application used to program this interface or module.

These objects correspond to the input/output images and software data of the module or integrated application-specific interface.

Reminders

The module inputs (%I and %IW) are updated in the PLC memory at the start of the task, the PLC being in RUN or STOP mode.

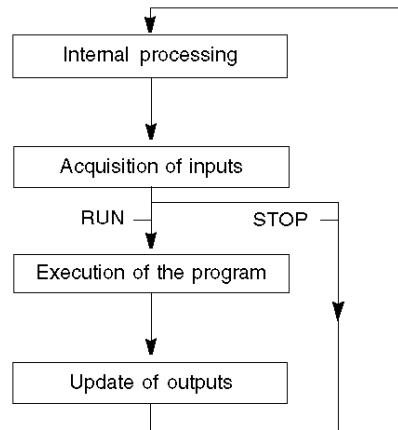
The outputs (%Q and %QW) are updated at the end of the task, only when the PLC is in RUN mode.

NOTE: When the task occurs in STOP mode, either of the following are possible, depending on the configuration selected:

- outputs are set to fallback position (fallback mode)
- outputs are maintained at their last value (maintain mode)

Figure

The following diagram shows the operating cycle of a PLC task (cyclical execution).



Explicit Exchange Language Objects Associated with the Application-Specific Function

Introduction

Explicit exchanges are performed at the user program's request using these instructions:

- READ_STS (*see Unity Pro, I/O Management, Block Library*) (read status words)
- WRITE_CMD (*see Unity Pro, I/O Management, Block Library*) (write command words)
- WRITE_PARAM (*see Unity Pro, I/O Management, Block Library*) (write adjustment parameters)
- READ_PARAM (*see Unity Pro, I/O Management, Block Library*) (read adjustment parameters)
- SAVE_PARAM (*see Unity Pro, I/O Management, Block Library*) (save adjustment parameters)
- RESTORE_PARAM (*see Unity Pro, I/O Management, Block Library*) (restore adjustment parameters)

These exchanges apply to a set of %MW objects of the same type (status, commands or parameters) that belong to a channel.

These objects can:

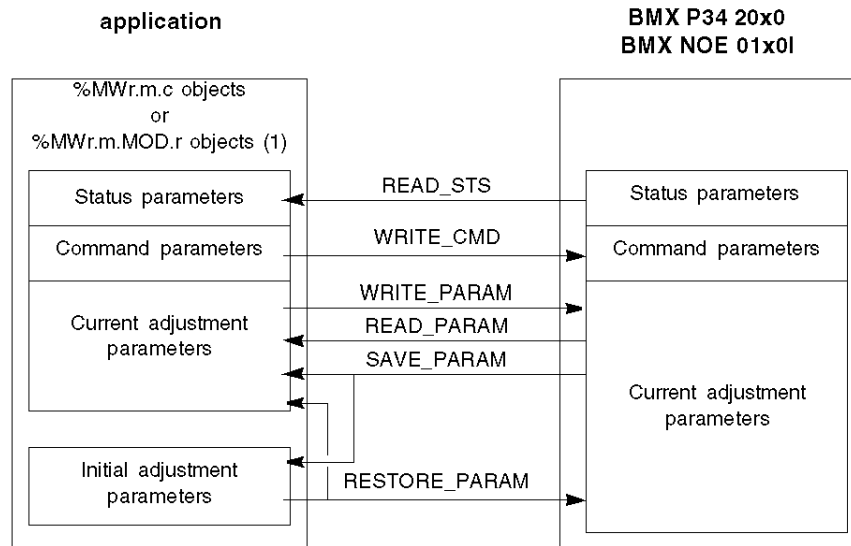
- provide information about the module (for example, type of error detected in a channel)
- have command control of the module (for example, switch command)
- define the module's operating modes (save and restore adjustment parameters in the process of application)

NOTE: To avoid several simultaneous explicit exchanges for the same channel, it is necessary to test the value of the word EXCH_STS (%MW_{r.m.c.0}) of the IODDT associated to the channel before calling any EF addressing this channel.

NOTE: Explicit Exchanges are not supported when Modicon M340 Analog and Digital I/O modules are configured behind a M340 Ethernet Remote I/O adapter module in a Quantum EIO Ethernet Configuration. As a consequence, it is not possible to setup a module's parameters from the PLC application during operation.

General Principle for Using Explicit Instructions

The diagram below shows the different types of explicit exchanges that can be made between the application and module.



(1) Only with READ_STS and WRITE_CMD instructions.

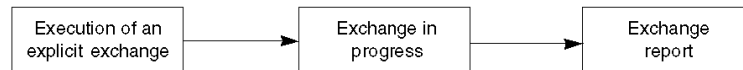
Managing Exchanges

During an explicit exchange, check performance to see that the data is only taken into account when the exchange has been correctly executed.

To do this, two types of information is available:

- information concerning the exchange in progress (*see page 394*)
- the exchange report (*see page 394*)

The following diagram describes the management principle for an exchange.



NOTE: In order to avoid several simultaneous explicit exchanges for the same channel, it is necessary to test the value of the word EXCH_STS (%MW_{r.m.c.}.0) of the IODDT associated to the channel before calling any EF addressing this channel.

Management of Exchanges and Reports with Explicit Objects

At a Glance

When data is exchanged between the PLC memory and the module, the module may require several task cycles to acknowledge this information. All IODDTs use two words to manage exchanges:

- EXCH_STS (%MW_{r.m.c.0}): exchange in progress
- EXCH_RPT (%MW_{r.m.c.1}): report

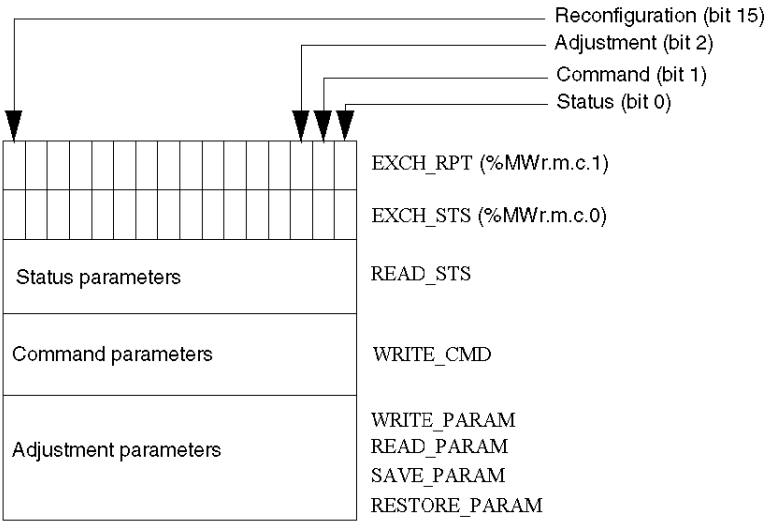
NOTE:

Depending on the localization of the module, the management of the explicit exchanges (%MW0.0.MOD.0.0 for example) will not be detected by the application:

- For in-rack modules, explicit exchanges are done immediately on the local PLC Bus and are finished before the end of the execution task. So, the READ_STS, for example, is always finished when the %MW0.0.mod.0.0 bit is checked by the application.
- For remote bus (Fipio for example), explicit exchanges are not synchronous with the execution task, so the detection is possible by the application.

Illustration

The illustration below shows the different significant bits for managing exchanges:



Description of Significant Bits

Each bit of the words `EXCH_STS` (%MW_{r.m.c.0}) and `EXCH_RPT` (%MW_{r.m.c.1}) is associated with a type of parameter:

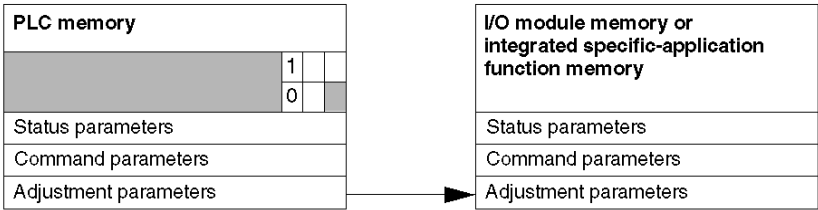
- Rank 0 bits are associated with the status parameters:
 - The `STS_IN_PROGR` bit (%MW_{r.m.c.0.0}) indicates whether a read request for the status words is in progress.
 - The `STS_ERR` bit (%MW_{r.m.c.1.0}) specifies whether a read request for the status words is accepted by the module channel.
- Rank 1 bits are associated with the command parameters:
 - The `CMD_IN_PROGR` bit (%MW_{r.m.c.0.1}) indicates whether command parameters are being sent to the module channel.
 - The `CMD_ERR` bit (%MW_{r.m.c.1.1}) specifies whether the command parameters are accepted by the module channel.
- Rank 2 bits are associated with the adjustment parameters:
 - The `ADJ_IN_PROGR` bit (%MW_{r.m.c.0.2}) indicates whether the adjustment parameters are being exchanged with the module channel (via `WRITE_PARAM`, `READ_PARAM`, `SAVE_PARAM`, `RESTORE_PARAM`).
 - The `ADJ_ERR` bit (%MW_{r.m.c.1.2}) specifies whether the adjustment parameters are accepted by the module. If the exchange is correctly executed, the bit is set to 0.
- Rank 15 bits indicate a reconfiguration on channel **c** of the module from the console (modification of the configuration parameters + cold start-up of the channel).
- The *r*, *m* and *c* bits indicates the following elements:
 - the **r** bit represents the rack number.
 - The **m** bit represents the position of the module in the rack.
 - The **c** bit represents the channel number in the module.

NOTE: **r** represents the rack number, **m** the position of the module in the rack, while **c** represents the channel number in the module.

NOTE: Exchange and report words also exist at module level `EXCH_STS` (%MW_{r.m.MOD}) and `EXCH_RPT` (%MW_{r.m.MOD.1}) as per IODDT type `T_GEN_MOD`.

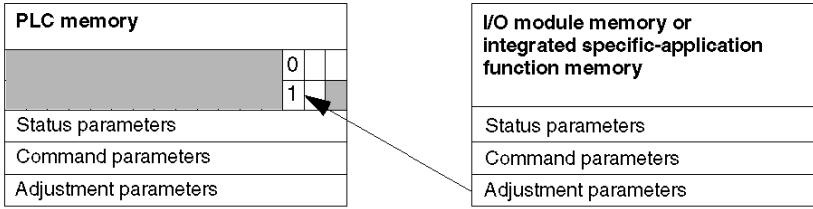
Example

Phase 1: Sending data by using the `WRITE_PARAM` instruction



When the instruction is scanned by the PLC processor, the **Exchange in progress** bit is set to 1 in `%MWr.m.c`.

Phase 2: Analysis of the data by the I/O module and report.



When the data is exchanged between the PLC memory and the module, acknowledgement by the module is managed by the `ADJ_ERR` bit (`%MWr.m.c.1.2`).

This bit makes the following reports:

- **0**: correct exchange
- **1**: faulty exchange)

NOTE: There is no adjustment parameter at module level.

Execution Indicators for an Explicit Exchange: EXCH_STS

The table below shows the control bits of the explicit exchanges: EXCH_STS (%MWr.m.c.0)

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Reading of channel status words in progress	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjust parameters exchange in progress	%MWr.m.c.0.2
RECONF_IN_PROGR	BOOL	R	Reconfiguration of the module in progress	%MWr.m.c.0.15

NOTE: If the module is not present or is disconnected, explicit exchange objects (READ_STS for example) are not sent to the module (STS_IN_PROG (%MWr.m.c.0.0) = 0), but the words are refreshed.

Explicit Exchange Report: EXCH_RPT

The table below shows the report bits: EXCH_RPT (%MWr.m.c.1)

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Error reading channel status words (1 = failure)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Error during a command parameter exchange (1 = failure)	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Error during an adjust parameter exchange (1 = failure)	%MWr.m.c.1.2
RECONF_ERR	BOOL	R	Error during reconfiguration of the channel (1 = failure)	%MWr.m.c.1.15

Counting Module Use

The following table describes the steps realised between a Counting Module and the system after a power-on.

Step	Action
1	Power on.
2	The system sends the configuration parameters.
3	The system sends the adjust parameters by WRITE_PARAM method. Note: When the operation is finished, the bit %MWr.m.c.0.2 switches to 0.

If, in the beginning of your application, you use a WRITE_PARAM command, you must wait until the bit %MWr.m.c.0.2 switches to 0.

Interface Language

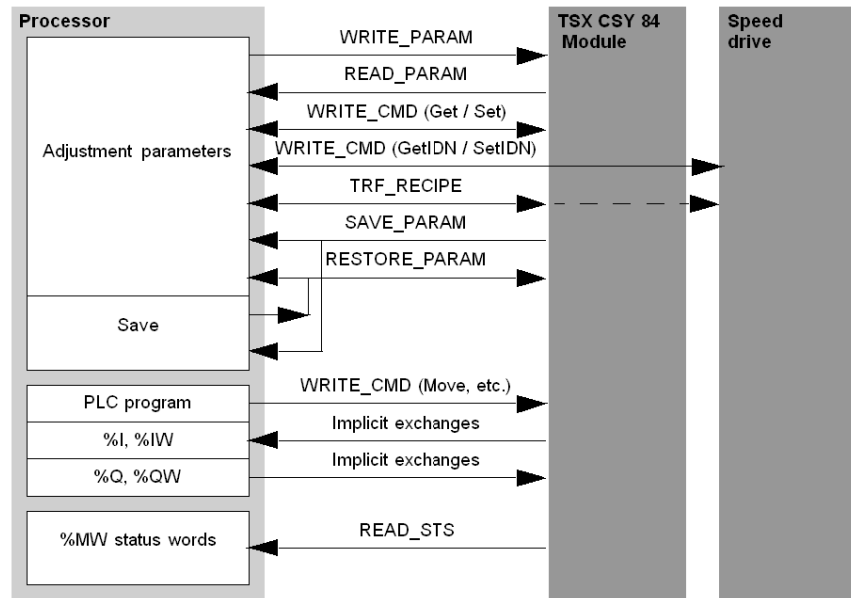
Module Channels

The TSX CSY 84 module is made up of up to 32 channels which support the following functions:

Channels	Function supported
%CHr.m.0	SERCOS® function
%CHr.m.1 to %CHr.m.8	Real axes
%CHr.m.9 to %CHr.m.12	Imaginary axes
%CHr.m.13 to %CHr.m.16	Remote Axes
%CHr.m.17 to %CHr.m.20	Co-ordinated axis group
%CHr.m.21 to %CHr.m.24	Slave axis group
%CHr.m.25 to %CHr.m.31	Cam Profiles

Overview of Exchanges

The exchanges between the processor, the axis control module and the variable speed controllers are performed as follows:



Parameter Management

General

The parameters of an axis control module and those of servodrives are managed separately via Unity Pro and UniLink. For example, it is possible to define the different values for limits in an axis control module and servodrives.

The parameters of an axis control module may be divided into 2 categories:

- Parameters used only by the axis control module,
- Parameters used both by the axis control module and the servodrives.

NOTE: Servodrives that operate with a linear type unit are not supported by the module TSX CSY 84/164.

Axis Control Module Parameters

Type "C" (Controller) parameters are exchanged whilst floating, in the axis control module units. These parameters are related to the path calculation or cam profiles. They are configured in Unity Pro configuration screens, then modified by the application as required. The units of these parameters are the command axis module units as defined in the configuration screens.

Servodrive Parameters

When the application modifies a parameter in the axis control module, the corresponding parameter in the servodrives is not updated (e.g. the minimum and maximum position limits, minimum and maximum acceleration, maximum speed, debugging window).

These parameters are considered to be servodrive system parameters. The user configures them via UniLink in order to protect the operational part and it is not necessary to change them when operating (the application program modifies the equivalent "C" type parameters in the command axis module).

Parameters used by the Axis Control module and the Servodrives

The axis control module does not update the servodrive parameters according to these equivalent parameter values.

Type "S" parameters (identifiers (IDN) SERCOS® Standard) and type "P" parameters (identifiers (IDN) SERCOS® Owner) are exchanged whilst floating with the axis control module and in full with the servodrives (e.g. acceleration, speed and position values).

If the application requires the value of the axis control module parameters to be synchronized with the servodrive parameters, it can read "S" or "P" type parameters (TRF_RECIPE) and then write them in the "C" type parameters (WRITE_PARAM).

"Unit" Parameters

Units are parameters used both by the axis control module and the servodrives.

Units are configured using Unity Pro. The conversion of axis control module units into servodrive units is automatically performed by the module TSX XSY 84/164 using a scale factor configured by the user.

WRITE_PARAM and READ_PARAM

Reminder

These services allow the exchange of adjustment parameters between the processor (project) and the axis control module.

`READ_PARAM`: explicit reading of the parameters in the axis control module and updating of the adjustment words %MW/D/Fr.m.c.r. via IODDTs.

`WRITE_PARAM`: explicit writing of parameters in the axis control module. This instruction allows programmed modification of the adjustment values defined in the configuration.

These two instructions are applied to an IODDT variable associated with the TSX CSY 84 module. In the following paragraphs we will take the example of a variable called `Sercos_Channel` of type `T_CSY_IND`.

Syntax of the instruction `READ_PARAM`

`READ_PARAM (Sercos_Channel)`: reads the adjustment parameters of the channel associated with the IODDT `Sercos_Channel`.

Syntax of instruction `WRITE_PARAM`

`WRITE_PARAM (Sercos_Channel)`: writes the adjustment parameters of the channel associated with the IODDT `Sercos_Channel`.

Monitoring the Exchange

The 2 following bits of the IODDT may be used to monitor the exchanges of adjustment parameters between the processor and the module:

Standard symbol	Meaning	Bit
ADJ_IN_PROGR	This bit is set to 1 while the exchange is in progress. It is reset to 0 when the exchange has been completed.	%MWr.m.c.0.2
ADJ_ERR	This bit is set to 1 if the parameters transmitted are out of range or erroneous.	%MWr.m.c.1.2

SAVE_PARAM and RESTORE_PARAM

Reminder

These services can be used to save or restore adjustment parameters.

SAVE_PARAM: explicit backup of the axis control module parameters. These parameters replace the initial values defined during configuration.

RESTORE_PARAM: explicit restoration of the initial adjustment parameters (written at the time of configuration or last backup).

These two instructions are applied to an IODDT variable associated with the TSX CSY 84 module. In the following paragraphs we will take the example of a variable called `Sercos_Channel` of type **T_CSY_IND**.

Syntax of instruction SAVE_PARAM

SAVE_PARAM (`Sercos_Channel`) : saves the adjustment parameters of the channel associated with the IODDT `Sercos_Channel`.

Syntax of instruction RESTORE_PARAM

RESTORE_PARAM (`Sercos_Channel`): restores the adjustment parameters of the channel associated with the IODDT `Sercos_Channel`.

Monitoring the Exchange

The 2 following bits of the IODDT may be used to monitor the exchanges of adjustment parameters between the processor and the module:

Standard symbol	Meaning	Bit
ADJ_IN_PROGR	This bit is set at 1 when the exchange is in process. It is reset to 0 when the exchange has been completed.	%MWr.m.c.0.2
ADJ_ERR	This bit is set at 1 if the parameters passed on are out of range or erroneous.	%MWr.m.c.1.2

WRITE_CMD

Reminder

This service allows a command to be sent to the motion controller.

WRITE_CMD: Explicit writing of command words in the module. This operation is carried out via internal words %MW, which contain the command to be carried out and its parameters (a motion control, for example).

This instruction is applied to an IODDT type variable associated with the TSX CSY 84 and TSX CSY 164 modules. For an independent axis, we will take the example of a variable called *Sercos_Channel* of type **T_CSY_IND**. For a co-ordinated axis, we shall take the example of a variable called *Sercos_Channel_coord* of type **T_CSY_COORD**.

Syntax of WRITE_CMD Instruction

WRITE_CMD (Sercos_Channel) : writes the command information of the channel associated with the IODDT *Sercos_Channel* (co-ordinated axis).

WRITE_CMD (Sercos_Channel_coord) for the co-ordinated axis.

WRITE_CMD Interface

The command to be performed is defined in the **ACTION_CMD** word (%MWr.m.c.26) of the *Sercos_Channel* IODDT and the result of the command is available in the words described in the following table:

Standard symbol	Type	Access	Meaning	Address
ERROR_CMD	INT	RW	WRITE_CMD command write error	%MWr.m.c.19
RETURN_CMD_1	DINT	RW	Return 1 of the function	%MDr.m.c.20
RETURN_CMD_2	REAL	RW	Return 2 of the function	%MFr.m.c.22
RETURN_CMD_3	REAL	RW	Return 3 of the function	%MFr.m.c.24
ACTION_CMD	INT	RW	Action to be carried out	%MWr.m.c.26
PARAM_CMD_1	DINT	RW	Parameter 1	%MDr.m.c.27
PARAM_CMD_2	DINT	RW	Parameter 2	%MDr.m.c.29
PARAM_CMD_3	REAL	RW	Parameter 3	%MFr.m.c.31
PARAM_CMD_4	REAL	RW	Parameter 4	%MFr.m.c.33

WRITE_CMD interface in the case of a coordinated axis group

In the case of a coordinated axis group, the movement functions make it necessary to send 2 parameters per coordinated axis (position and speed). The IODDT associated with this axis is thus a little more sophisticated. The result of the command and the parameters are located in the following words of the `Sercos_Channel_Coord` IODDT:

Standard symbol	Type	Access	Meaning	Address
...	...	...	...	...
PARAM_CMD_1	DINT	RW	Parameter 1	%MDr.m.c.27
PARAM_CMD_2	DINT	RW	Parameter 2	%MDr.m.c.29
PARAM_CMD_3	REAL	RW	Parameter 3	%MFr.m.c.31
PARAM_CMD_4	REAL	RW	Parameter 4	%MFr.m.c.33
PARAM_CMD_5	REAL	RW	Parameter 5	%MFr.m.c.35
...	...	...	...	...
PARAM_CMD_18	REAL	RW	Parameter 18	%MFr.m.c.61

Monitoring the Exchange

The 2 following bits of the IODDT may be used to monitor the writing of command information in the module:

Standard symbol	Meaning	Bit
CMD_IN_PROGR	This bit is set to 1 while the exchange is in progress. It is reset to 0 when the exchange has been completed.	%MWr.m.c.0.1
CMD_ERR	This bit is set to 1 if the parameters transmitted are out of range or erroneous.	%MWr.m.c.1.1

WRITE_CMD Examples

Example 1: Initializing Faults

To initialize faults on the bus ring of module 4 in rack 1: declare the Ring IODDT as type T_CSY_RING and associate to channel 0 of the TSX CSY 84 or TSX CSY 164 module in rack 1, slot 4:

```
! (*If WRITE_CMD not in progress, then initialization of faults*)
  IF NOT Ring.CMD_IN_PROGR THEN Ring.ACTION_CMD:= 409;
  WRITE_CMD(Ring);
  END_IF;
```

Example 2: Writing the Optical Power

To define (write) the optical power. We use the same IODDT:

```
! (*If WRITE_CMD not in progress, then writing (set) of the optical
power*)
  IF NOT Ring.CMD_IN_PROGR THEN Ring.ACTION_CMD:= 2545;
  Ring.PARAM_CMD_3:= 51.5;      //Parameter 3: optical power = 51.5
  WRITE_CMD(Ring);
  END_IF;
```

Example 3: Reading the transmission ratio

To read the transmission ratio (BAUD_RATE):

```
! (*If WRITE_CMD not in progress, then reading (get) of the
transmission ratio*)
  IF NOT Ring.CMD_IN_PROGR THEN Ring.ACTION_CMD:= 1551;
  WRITE_CMD(Ring);
  END_IF;

! (*If WRITE_CMD has been completed and if there is no fault, then
the value of the transmission ratio is accessible in return 1*)
  IF NOT Ring.CMD_IN_PROGR AND NOT Ring.CMD_ERR
  THEN BAUD_RATE:=           //transmission ratio in return 1
  Ring.RETURN_CMD_1;
  END_IF;
```

Example 4: Moving a real axis

Moving the real axis 3 from module 4 set in rack 1, at position 105.2, at speed 5 with the command "MOVE absolute immediate". We declare the `Axis_3` IODDT of type `T_CSY_IND` and associate it with channel 3 of module 4 of rack 1:

```
! (*If WRITE_CMD not in progress, then move real axis 3*)
  IF NOT Axis_3.CMD_IN_PROGR THEN Axis_3.ACTION_CMD:= 513;
  Axis_3.PARAM_CMD_1:= 0;           //Parameter 1: type of movement = absolute
  Axis_3.PARAM_CMD_3:= 105.2;       //Parameter 3: position = 105.2
  Axis_3.PARAM_CMD_4:= 5.0;         //Parameter 4: speed = 5
  WRITE_CMD(Axis_3);
  END_IF;
```

Example 5: Reading the position of a slave

Read the relative position of a follower slave when the master is at 102.5. We declare the IODDT `Follow` of type `T_CSY_FOLLOW` and associate it with the channel corresponding to the follower slave:

```
! (*If WRITE_CMD not in progress, then reading the position of the
  follower slave*)
  IF NOT Follow.CMD_IN_PROGR THEN Follow.ACTION_CMD:= 537;
  Follow.PARAM_CMD_3:= 102.5;       //Parameter 3: position of the master =
                                     102.5
  WRITE_CMD(Follow);
  END_IF;
! (*If WRITE_CMD has been completed and if there is no fault, then
  the position of the slave is accessible in return 2*)
  IF NOT Follow.CMD_IN_PROGR AND NOT Follow.CMD_ERR
  THEN                               //position in return 2
    SLAVE_POSITION:=Follow.RETURN_CM
    D_2;
  END_IF;
```

READ_STS

Reminder

This service allows one to read the status words associated with the axis control module or with the different channels explicitly.

This instruction may be applied to all types of IODDT that can be associated with the TSX CSY 84 or TSX CSY 164 module. For an independent axis, we will take the example of a variable called `Serco_Channel` of type **T_CSY_IND**.

It is also possible to apply this instruction to a TSX CSY 84 or TSX CSY 164 module. In this case, it has to be applied to an IODDT of type **T_CSY_CMD**. Let us use the example of the variable called `Sercos_module`.

Syntax of the READ_STS (module) instruction

`READ_STS (Sercos_Module):` reading of general diagnostics information from the module associated with the `Serco_Module` IODDT.

Syntax of the READ_STS (channel) instruction

`READ_STS (Sercos_Channel):` reading of general diagnostics information from the channel associated with the `Sercos_Channel` IODDT.

Monitoring the Exchange

The 2 following bits of the IODDT may be used to monitor the exchanges of status words between the processor and the module:

Standard symbol	Meaning	Bit
STS_IN_PROGR	This bit is set at 1 when the reading is in process. It is reset to 0 when the exchange has been completed.	%MWr.m.c.0.0
STS_ERR	This bit provides the exchange report. It is set to 1 in the case of fault.	%MWr.m.c.1.0

TRF_RECIPE and MOD_PARAM

Reminder

- **TRF_RECIPE:**

this instruction is explained in this manual (*see page 302*).

- **MOD_PARAM:**

this instruction is explained in this manual (*see page 482*).

16.2 Language Objects and IODDTs Associated Specifically with the SERCOS Module

Subject of this Section

This section presents the language objects and IODDTs associated with the SERCOS module.

What Is in This Section?

This section contains the following topics:

Topic	Page
Details concerning T_CSY_CMD-type IODDT implicit exchange objects	409
Details concerning T_CSY_CMD-type IODDT explicit exchange objects	411
Details concerning T_CSY_RING-type IODDT implicit exchange objects	413
Details concerning T_CSY_RING-type IODDT explicit exchange objects	416
Details concerning T_CSY_TRF-type IODDT explicit exchange objects	419
Details concerning T_CSY_IND-type IODDT implicit exchange objects	421
Details concerning T_CSY_IND-type IODDT explicit exchange objects	425
Details concerning T_CSY_FOLLOW-type IODDT implicit exchange objects	429
Details concerning T_CSY_FOLLOW-type IODDT explicit exchange objects	432
Details concerning T_CSY_COORD-type IODDT implicit exchange objects	440
Details concerning T_CSY_COORD-type IODDT explicit exchange objects	443
Details concerning T_CSY_CAM-type IODDT implicit exchange objects	446
Details concerning T_CSY_CAM-type IODDT explicit exchange objects	447

Details concerning T_CSY_CMD-type IODDT implicit exchange objects

At a Glance

The T_CSY_CMD IODDT possesses implicit exchange objects, which are described below. This type of IODDT applies to the TSX_CSY_84 module.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_CMD

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

List of implicit exchange input objects

The following table introduces the implicit exchange input objects of the T_CSY_CMD IODDT which apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CH_ERROR	EBOOL	R	Channel error bit.	%lr.m.c.ERR
PROFILE_END	EBOOL	R	The last profile command has been sent to the module	%lr.m.c.3
IN_POSITION	EBOOL	R	The axis is within the in-position band	%lr.m.c.4
AXIS_HOMED	EBOOL	R	The axis position reading is referenced to the home position	%lr.m.c.6
HOLDING	EBOOL	R	The axis is holding in wait position	%lr.m.c.8
RESUMING	EBOOL	R	The axis is moving after a hold	%lr.m.c.9
DRIVE_ENABLED	EBOOL	R	The drive is enabled	%lr.m.c.10
DRIVE_FLT	EBOOL	R	The drive is performing a class 1 diagnostic	%lr.m.c.13
AXIS_SUMMARY_FLT	EBOOL	R	Drive fault	%lr.m.c.15
AXIS_IN_CMD	EBOOL	R	The axis is active and can be controlled	%lr.m.c.18
AXIS_HOLD	EBOOL	R	The axis is stopped and waiting for a command	%lr.m.c.28
AXIS_HALT	EBOOL	R	The axis has stopped	%lr.m.c.29
AXIS_FASTSTOP	EBOOL	R	The axis has faststopped	%lr.m.c.30
AXIS_READY	EBOOL	R	The axis is ready to respond to a command	%lr.m.c.31

List of implicit exchange output objects

The following table introduces the implicit exchange output objects of the T_CSY_CMD IODDT that apply to TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CONTROL_ACQUIRE	EBOOL	RW	Control acquisition	%Qr.m.c.2
CONTROL_ENABLE	EBOOL	RW	Control enable	%Qr.m.c.10
CONTROL_RESUME	EBOOL	RW	Resumes control after a stop	%Qr.m.c.12
CONTROL_CLEAR_FLT	EBOOL	RW	Fault clear control	%Qr.m.c.15
ALLOW_ACQUIRE	EBOOL	RW	Acquisition enable control	%Qr.m.c.18
ALLOW_ENABLE	EBOOL	RW	Disable axis control	%Qr.m.c.26
ALLOW_RESUME	EBOOL	RW	Authorizes a movement to continue after a stop using the HOLD command	%Qr.m.c.28
ALLOW_MOVE	EBOOL	RW	Authorizes a movement to continue after a stop using the HALT command	%Qr.m.c.29

Details concerning T_CSY_CMD-type IODDT explicit exchange objects

At a Glance

This part introduces explicit exchange objects of the T_CSY_CMD IODDT which apply to the TSX CSY 84 module. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_CMD

Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

Indicators of explicit exchange execution: EXCH_STS

The table below explains the meanings of the exchange control bits for the EXCH_STS channel (%MWr.m.c.0).

Standard symbol	Type	Access	Meaning	Address
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1

Explicit exchange report: EXCH_RPT

The following table explains the meaning of the EXCH_RPT report bits (%MWr.m.c.1).

Standard symbol	Type	Access	Meaning	Address
CMD_ERR	BOOL	R	Fault during exchange of command parameters	%MWr.m.c.1.1

WRITE_CMD interface words

The following table explains the meanings of the variables associated with the `WRITE_CMD` whose action was specified in the word `ACTION_CMD`. These variables are updated by a `WRITE_CMD (IODDT_VAR1)`.

Standard symbol	Type	Access	Meaning	Address
<code>ERROR_CMD</code>	INT	RW	Error during <code>WRITE_CMD</code>	%MWr.m.c.19
<code>RETURN_CMD_1</code>	DINT	RW	Return 1 of the function	%MDr.m.c.20
<code>RETURN_CMD_2</code>	REAL	RW	Return 2 of the function	%MFr.m.c.22
<code>RETURN_CMD_3</code>	REAL	RW	Return 3 of the function	%MFr.m.c.24
<code>ACTION_CMD</code>	INT	RW	Action to be carried out	%MWr.m.c.26
<code>PARAM_CMD_1</code>	DINT	RW	Parameter 1	%MDr.m.c.27
<code>PARAM_CMD_2</code>	DINT	RW	Parameter 2	%MDr.m.c.29
<code>PARAM_CMD_3</code>	REAL	RW	Parameter 3	%MFr.m.c.31
<code>PARAM_CMD_4</code>	REAL	RW	Parameter 4	%MFr.m.c.33

Details concerning T_CSY_RING-type IODDT implicit exchange objects

At a Glance

The T_CSY_RING IODDT possesses implicit exchange objects, which are described below. This type of IODDT applies to channel 0 of the TSX_CSY_84 module.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_RING

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

List of implicit exchange input objects

The following table introduces the implicit exchange input objects of the T_CSY_RING IODDT which apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CH_ERROR	EBOOL	R	Channel error bit.	%lr.m.c.ERR
RAMPING	EBOOL	R	Indicates whether the axis is accelerating or decelerating	%lr.m.c.0
STEADY	EBOOL	R	The speed is steady	%lr.m.c.1
STOPPING	EBOOL	R	The movement is decelerating to a stop	%lr.m.c.2
PROFILE_END	EBOOL	R	The last profile command has been sent to the module	%lr.m.c.3
IN_POSITION	EBOOL	R	The axis is within the in-position band	%lr.m.c.4
AXIS_HOMING	EBOOL	R	The axis is homing. For an imaginary axis, this bit is inactive	%lr.m.c.5
AXIS_HOMED	EBOOL	R	The axis position reading is referenced off the home position	%lr.m.c.6
AXIS_NOT_FOLLOWING	EBOOL	R	The drive is not recognizing module commands	%lr.m.c.7
HOLDING	EBOOL	R	The axis is holding in wait position	%lr.m.c.8
RESUMING	EBOOL	R	The axis is moving after a hold	%lr.m.c.9
DRIVE_ENABLED	EBOOL	R	The drive is enabled	%lr.m.c.10
DRIVE_DIAG	EBOOL	R	The drive is performing a class 3 diagnostic	%lr.m.c.11
DRIVE_WARNING	EBOOL	R	The drive is performing a class 2 diagnostic	%lr.m.c.12
DRIVE_FLT	EBOOL	R	The drive is performing a class 1 diagnostic	%lr.m.c.13
DRIVE_DISABLED	EBOOL	R	The drive is disabled	%lr.m.c.14
AXIS_SUMMARY_FLT	EBOOL	R	Drive fault	%lr.m.c.15

Standard symbol	Type	Access	Meaning	Address
AXIS_COM_OK	EBOOL	R	Communication between the drive and the module is OK	%Ir.m.c.16
AXIS_IS_LINKED	EBOOL	R	The axis belongs to a set of axes	%Ir.m.c.17
AXIS_IN_CMD	EBOOL	R	The axis is active and can be controlled	%Ir.m.c.18
AXIS_AT_TARGET	EBOOL	R	The axis is within the in-position band for the target position	%Ir.m.c.20
AXIS_POS_LIMIT	EBOOL	R	The axis has reached the positive limit	%Ir.m.c.21
AXIS_NEG_LIMIT	EBOOL	R	The axis has reached the negative limit	%Ir.m.c.22
AXIS_WARNING	EBOOL	R	MotionWarning status returned by the drive	%Ir.m.c.23
AXIS_HOLD	EBOOL	R	The axis is stopped and waiting for a command	%Ir.m.c.28
AXIS_HALT	EBOOL	R	The axis has stopped	%Ir.m.c.29
AXIS_FASTSTOP	EBOOL	R	The axis has faststopped	%Ir.m.c.30
AXIS_READY	EBOOL	R	The axis is ready to respond to a command	%Ir.m.c.31
CONF_OK	EBOOL	R	The channel is configured	%Ir.m.c.32

List of implicit exchange output objects

The following table introduces the implicit exchange output objects of the T_CSY_RING IODDT that apply to TSX_CSY_84 module.

Symbole standard	Type	Access	Meaning	Address
CONTROL_ACQUIRE	EBOOL	RW	control acquisition	%Qr.m.c.2
CONTROL_ENABLE	EBOOL	RW	Control enable	%Qr.m.c.10
CONTROL_FOLLOW	EBOOL	RW	Follow control for an axis or a set of follower axes	%Qr.m.c.11
CONTROL_RESUME	EBOOL	RW	Resumes control after a stop	%Qr.m.c.12
CONTROL_CLEAR_FLT	EBOOL	RW	Fault clear control	%Qr.m.c.15
ALLOW_ACQUIRE	EBOOL	RW	Acquisition enable control	%Qr.m.c.18
ALLOW_ENABLE	EBOOL	RW	Disable axis control	%Qr.m.c.26
ALLOW_FOLLOW	EBOOL	RW	Control to cancel following for an axis or a set of follower axes	%Qr.m.c.27
ALLOW_RESUME	EBOOL	RW	Authorizes a movement to continue after a stop using the HOLD command	%Qr.m.c.28
ALLOW_MOVE	EBOOL	RW	Authorizes a movement to continue after a stop using the HALT command	%Qr.m.c.29
ALLOW_NOT_FASTSTOP	EBOOL	RW	Control after a Faststop	%Qr.m.c.30
ALLOW_NOT_FLT	EBOOL	RW	Enable fault control	%Qr.m.c.31

Parameter report word

The table below introduces the implicit exchange input word of the T_CSY_RING IODDT which applies to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
PARAM_RPT	INT	R	Parameters report signaling a programming fault. The least significant byte contains the error code and the most significant byte contains the address in the registers of the field that triggered the error.	%IW.r.m.c.2

Details concerning T_CSY_RING-type IODDT explicit exchange objects

At a Glance

This part introduces explicit exchange objects of the T_CSY_RING IODDT which apply to channel 0 of the TSX CSY 84 module. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_RING

Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

Indicators of explicit exchange execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: EXCH_STS (%MWr.m.c.0). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Read channel status words in progress	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjust parameters exchange in progress	%MWr.m.c.0.2
TRF_IN_PROGR	BOOL	R	TRF_RECIPÉ function in progress	%MWr.m.c.0.3
RECONF_IN_PROGR	BOOL	R	Module reconfiguring	%MWr.m.c.0.15

Explicit exchange report: EXCH_RPT

The following table introduces the report bits: EXCH_RPT (%MWr.m.c.1). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Channel status words read fault (1 = failure)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Fault during exchange of command parameters (1 = failure)	%MWr.m.c.1.1

Standard symbol	Type	Access	Meaning	Address
ADJ_ERR	BOOL	R	Fault during an exchange of adjustment parameters (1 = failure)	%MWr.m.c.1.2
TRF_ERR	BOOL	R	Fault while TRF_RECIPE function is in progress	%MWr.m.c.1.3
RECONF_ERR	BOOL	R	Fault during channel reconfiguration (1 = failure)	%MWr.m.c.1.15

Channel fault word

The following table introduces the channel fault bits: `CH_FLT`. These variables are updated by a `READ_STS (IODDT_VAR1)`.

Standard symbol	Type	Access	Meaning	Address
EXT_FLT0	BOOL	R	External fault 0: servodrive fault	%MWr.m.c.2.0
EXT_FLT1	BOOL	R	External fault 1: communication fault with the axis	%MWr.m.c.2.1
EXT_FLT2	BOOL	R	External fault 2	%MWr.m.c.2.3
INT_FLT	BOOL	R	Internal fault	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Configuration fault: different hardware and software configurations	%MWr.m.c.2.5
COM_FLT	BOOL	R	Communication fault	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Faulty project configuration, adjustment or command fault	%MWr.m.c.2.7
FAN_STOPPED	BOOL	R	Fan fault (channel 0 only)	%MWr.m.c.2.8
OVER_TEMP	BOOL	R	Overtemperature (channel 0 only)	%MWr.m.c.2.9
SENSOR_FLT	BOOL	R	Temperature sensor fault (channel 0 only)	%MWr.m.c.2.10
PROCESS_CONF	BOOL	R	Creation of move in progress object	%MWr.m.c.2.11
PROCESS_CONF_FAILED	BOOL	R	Configuration fault (except for channel 0)	%MWr.m.c.2.12

WRITE_CMD interface words

The following table explains the meanings of the variables associated with the `WRITE_CMD` whose action was specified in the word `ACTION_CMD`. These variables are updated by a `WRITE_CMD (IODDT_VAR1)`.

Standard symbol	Type	Access	Meaning	Address
<code>ERROR_CMD</code>	INT	RW	Error during <code>WRITE_CMD</code>	%MWr.m.c.19
<code>RETURN_CMD_1</code>	DINT	RW	Return 1 of the function	%MDr.m.c.20
<code>RETURN_CMD_2</code>	REAL	RW	Return 2 of the function	%MFr.m.c.22
<code>RETURN_CMD_3</code>	REAL	RW	Return 3 of the function	%MFr.m.c.24
<code>ACTION_CMD</code>	INT	RW	Action to be carried out	%MWr.m.c.26
<code>PARAM_CMD_1</code>	DINT	RW	Parameter 1	%MDr.m.c.27
<code>PARAM_CMD_2</code>	DINT	RW	Parameter 2	%MDr.m.c.29
<code>PARAM_CMD_3</code>	REAL	RW	Parameter 3	%MFr.m.c.31
<code>PARAM_CMD_4</code>	REAL	RW	Parameter 4	%MFr.m.c.33

READ_PARAM, WRITE_PARAM interface words

The table below explains the meanings of the parameters which can be accessed using the `READ_PARAM` and `WRITE_PARAM` functions for channels 1 to 16. These variables are updated using `READ_PARAM (IODDT_VAR1)` or `WRITE_PARAM (IODDT_VAR1)`. You can also use the `SAVE_PARAM` and `RESTORE_PARAM` functions.

Standard symbol	Type	Access	Meaning	Address
<code>CYCLE_TIME</code>	INT	RW	SERCOSring cycle time	%MWr.m.c.35
<code>BAUD_RATE</code>	INT	RW	Baud rate on SERCOS bus (in Bauds)	%MWr.m.c.36
<code>OPTICAL_POWER</code>	INT	RW	Optical power in the fiber	%MWr.m.c.37

Details concerning T_CSY_TRF-type IODDT explicit exchange objects

At a Glance

This part introduces T_CSY_TRF IODDT explicit exchange objects that apply to the TSX CSY 84 module for channels 1 to 32. It groups the word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_TRF

Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

Execution flag for the TRF_RECIPE function

The following table introduces the control bit that indicates if the TRF_RECIPE is in progress.

Standard symbol	Type	Access	Meaning	Address
TRF_IN_PROGR	BOOL	R	TRF_RECIPE function in progress	%MWr.m.c.0.3

Explicit exchange report: EXCH_RPT

The table below introduces the report bit specific to the TRF_RECIPE function.

Standard symbol	Type	Access	Meaning	Address
TRF_ERR	BOOL	R	Fault while TRF_RECIPE function is in progress	%MWr.m.c.1.3

Objects of the TRF_RECIPE function

The following table introduces the objects associated with the TRF_RECIPE function. These objects are automatically updated by the system each time the TRF_RECIPE function is used.

Standard symbol	Type	Access	Meaning	Address
ERROR_TRF	INT	R	Read error for the TRF_RECIPE function	%MWr.m.c.3
RETURN_TRF_1	DINT	R	Return 1 of the TRF_RECIPE function	%MDr.m.c.4
RETURN_TRF_2	REAL	R	Return 2 of the TRF_RECIPE function	%MFr.m.c.6
RETURN_TRF_3	REAL	R	Return 3 of the TRF_RECIPE function	%MFr.m.c.8
ACTION_TRF	INT	R	Action to be carried out by the TRF_RECIPE function	%MWr.m.c.10
PARAM_TRF_1	DINT	R	Parameter 1 of the TRF_RECIPE function	%MDr.m.c.11
PARAM_TRF_2	DINT	R	Parameter 2 of the TRF_RECIPE function	%MDr.m.c.13
PARAM_TRF_3	REAL	R	Parameter 3 of the TRF_RECIPE function	%MFr.m.c.15
PARAM_TRF_4	REAL	R	Parameter 4 of the TRF_RECIPE function	%MFr.m.c.17

Details concerning T_CSY_IND-type IODDT implicit exchange objects

At a Glance

The T_CSY_IND IODDT possesses implicit exchange objects, which are described below. This type of IODDT is applied to the TSX_CSY_84 module for channels 21 to 16.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_IND

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

List of implicit exchange input objects

The following table introduces the implicit exchange input objects of the T_CSY_IND IODDT which apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CH_ERROR	EBOOL	R	Channel error bit.	%lr.m.c.ERR
RAMPING	EBOOL	R	Indicates whether the axis is accelerating or decelerating	%lr.m.c.0
STEADY	EBOOL	R	The speed is steady	%lr.m.c.1
STOPPING	EBOOL	R	The movement is decelerating to a stop	%lr.m.c.2
PROFILE_END	EBOOL	R	The last profile command has been sent to the module	%lr.m.c.3
IN_POSITION	EBOOL	R	The axis is within the in-position band	%lr.m.c.4
AXIS_HOMING	EBOOL	R	The axis is homing. For an imaginary axis, this bit is inactive	%lr.m.c.5
AXIS_HOMED	EBOOL	R	The axis position reading is referenced off the home position	%lr.m.c.6
AXIS_NOT_FOLLOWING	EBOOL	R	The drive is not recognizing module commands	%lr.m.c.7
HOLDING	EBOOL	R	The axis is holding in wait position	%lr.m.c.8
RESUMING	EBOOL	R	The axis is moving after a hold	%lr.m.c.9
DRIVE_ENABLED	EBOOL	R	The drive is enabled	%lr.m.c.10
DRIVE_DIAG	EBOOL	R	The drive is performing a class 3 diagnostic	%lr.m.c.11
DRIVE_WARNING	EBOOL	R	The drive is performing a class 2 diagnostic	%lr.m.c.12
DRIVE_FLT	EBOOL	R	The drive is performing a class 1 diagnostic	%lr.m.c.13
DRIVE_DISABLED	EBOOL	R	The drive is disabled	%lr.m.c.14
AXIS_SUMMARY_FLT	EBOOL	R	Drive fault	%lr.m.c.15

Standard symbol	Type	Access	Meaning	Address
AXIS_COM_OK	EBOOL	R	Communication between the drive and the module is OK	%lr.m.c.16
AXIS_IS_LINKED	EBOOL	R	The axis belongs to a set of axes	%lr.m.c.17
AXIS_IN_CMD	EBOOL	R	The axis is active and can be controlled	%lr.m.c.18
AXIS_AT_TARGET	EBOOL	R	The axis is within the in-position band for the target position	%lr.m.c.20
AXIS_POS_LIMIT	EBOOL	R	The axis has reached the positive limit	%lr.m.c.21
AXIS_NEG_LIMIT	EBOOL	R	The axis has reached the negative limit	%lr.m.c.22
AXIS_WARNING	EBOOL	R	MotionWarning status returned by the drive	%lr.m.c.23
BIAS_REMAIN	EBOOL	R	Offset added to the command position	%lr.m.c.24
AXIS_MANUAL_MODE	EBOOL	R	Axis operating in Manual Mode	%lr.m.c.25
DRIVE_REALTIME_BIT1	EBOOL	R	Drive bit	%lr.m.c.26
DRIVE_REALTIME_BIT2	EBOOL	R	Drive bit	%lr.m.c.27
AXIS_HOLD	EBOOL	R	The axis is stopped and waiting for a command	%lr.m.c.28
AXIS_HALT	EBOOL	R	The axis has stopped	%lr.m.c.29
AXIS_FASTSTOP	EBOOL	R	The axis has faststopped	%lr.m.c.30
AXIS_READY	EBOOL	R	The axis is ready to respond to a command	%lr.m.c.31
CONF_OK	EBOOL	R	The channel is configured	%lr.m.c.32

List of implicit exchange output objects

The following table introduces the implicit exchange output objects of the T_CSY_IND IODDT that apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CONTROL_ACQUIRE	EBOOL	RW	control acquisition	%Qr.m.c.2
CONTROL_JOG_POS	EBOOL	RW	Manual Mode: visible movement command in positive direction along axis	%Qr.m.c.4
CONTROL_JOG_NEG	EBOOL	RW	Manual Mode: visible movement command in negative direction along axis	%Qr.m.c.5
REAL_TIME_CTRL_BIT1	EBOOL	RW	Drive bit	%Qr.m.c.6
REAL_TIME_CTRL_BIT2	EBOOL	RW	Drive bit	%Qr.m.c.7
OPERATION_MODE_1	EBOOL	RW	Selection of operating mode	%Qr.m.c.8
OPERATION_MODE_2	EBOOL	RW	Selection of operating mode	%Qr.m.c.9
CONTROL_ENABLE	EBOOL	RW	Control enable	%Qr.m.c.10
CONTROL_FOLLOW	EBOOL	RW	Follow control for an axis or a set of follower axes	%Qr.m.c.11
CONTROL_RESUME	EBOOL	RW	Resumes control after a stop	%Qr.m.c.12

Standard symbol	Type	Access	Meaning	Address
CONTROL_INC_POS	EBOOL	RW	Manual Mode: incremental movement command in positive direction along axis	%Qr.m.c.13
CONTROL_INC_NEG	EBOOL	RW	Manual Mode: incremental movement command in negative direction along axis	%Qr.m.c.14
CONTROL_CLEAR_FLT	EBOOL	RW	Fault clear control	%Qr.m.c.15
ALLOW_ACQUIRE	EBOOL	RW	Acquisition enable control	%Qr.m.c.18
ALLOW_ENABLE	EBOOL	RW	Disable axis control	%Qr.m.c.26
ALLOW_FOLLOW	EBOOL	RW	Control to cancel following for an axis or a set of follower axes	%Qr.m.c.27
ALLOW_RESUME	EBOOL	RW	Authorizes a movement to continue after a stop using the HOLD command	%Qr.m.c.28
ALLOW_MOVE	EBOOL	RW	Authorizes a movement to continue after a stop using the HALT command	%Qr.m.c.29
ALLOW_NOT_FASTSTOP	EBOOL	RW	Control after a Faststop	%Qr.m.c.30
ALLOW_NOT_FLT	EBOOL	RW	Enable fault control	%Qr.m.c.31

Actual position

The following table introduces the implicit exchange input real of the `T_CSY_IND` IODDT which applies to the `TSX_CSY_84` module.

Standard symbol	Type	Access	Meaning	Address
POSITION	REAL	R	Actual position	%IFr.m.c.0

Parameter report word

The following table introduces the implicit exchange input word for the `T_CSY_IND` IODDT that applies to the `TSX_CSY_84` module.

Standard symbol	Type	Access	Meaning	Address
PARAM_RPT	INT	R	Parameters report signaling a programming fault. The least significant byte contains the error code and the most significant byte contains the address in the registers of the field that triggered the error.	%IWm.c.2

Simulated position

The following table introduces the implicit exchange double output word of the `T_CSY_IND` IODDT which applies to the `TSX_CSY_84` module.

Standard symbol	Type	Access	Meaning	Address
REMOTE_POSITION	DINT	RW	Real and imaginary axis: position increment in manual mode. Remote axis: simulated position	%QDr.m.c.0

Details concerning T_CSY_IND-type IODDT explicit exchange objects

At a Glance

This part introduces IODDT explicit exchange objects of the `T_CSY_IND` type that apply to the TSX CSY 84 module for channels 1 to16. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_IND`

Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

Indicators of explicit exchange execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: `EXCH_STS` (%MWr.m.c.0). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Read channel status words in progress	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjust parameters exchange in progress	%MWr.m.c.0.2
RECONF_IN_PROGR	BOOL	R	Module reconfiguring	%MWr.m.c.0.15

Explicit exchange report: EXCH_RPT

The following table introduces the report bits: `EXCH_RPT` (%MWr.m.c.1). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Channel status words read fault (1 = failure)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Fault during exchange of command parameters (1 = failure)	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Fault during an exchange of adjustment parameters (1 = failure)	%MWr.m.c.1.2
RECONF_ERR	BOOL	R	Fault during channel reconfiguration (1 = failure)	%MWr.m.c.1.15

Channel fault word

The following table introduces the channel fault bits: `CH_FLT`. These variables are updated by a `READ_STS` (IODDT_VAR1).

Standard symbol	Type	Access	Meaning	Address
EXT_FLT0	BOOL	R	External fault 0: servodrive fault	%MWr.m.c.2.0
EXT_FLT1	BOOL	R	External fault 1: communication fault with the axis	%MWr.m.c.2.1
EXT_FLT2	BOOL	R	External fault 2	%MWr.m.c.2.3
INT_FLT	BOOL	R	Internal fault	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Configuration fault: different hardware and software configurations	%MWr.m.c.2.5
COM_FLT	BOOL	R	Communication fault	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Faulty project configuration, adjustment or command fault	%MWr.m.c.2.7
PROCESS_CONF	BOOL	R	Creation of move in progress object	%MWr.m.c.2.11
PROCESS_CONF_FAILED	BOOL	R	Configuration fault (except for channel 0)	%MWr.m.c.2.12

Objects of the TRF_RECIPE function

The following table introduces the objects associated with the TRF_RECIPE function. These objects are automatically updated by the system each time the TRF_RECIPE function is used.

Standard symbol	Type	Access	Meaning	Address
ERROR_TRF	INT	R	Read error for the TRF_RECIPE function	%MWr.m.c.3
RETURN_TRF_1	DINT	R	Return 1 of the TRF_RECIPE function	%MDr.m.c.4
RETURN_TRF_2	REAL	R	Return 2 of the TRF_RECIPE function	%MFr.m.c.6
RETURN_TRF_3	REAL	R	Return 3 of the TRF_RECIPE function	%MFr.m.c.8
ACTION_TRF	INT	R	Action to be carried out by the TRF_RECIPE function	%MWr.m.c.10
PARAM_TRF_1	DINT	R	Parameter 1 of the TRF_RECIPE function	%MDr.m.c.11
PARAM_TRF_2	DINT	R	Parameter 2 of the TRF_RECIPE function	%MDr.m.c.13
PARAM_TRF_3	REAL	R	Parameter 3 of the TRF_RECIPE function	%MFr.m.c.15
PARAM_TRF_4	REAL	R	Parameter 4 of the TRF_RECIPE function	%MFr.m.c.17

WRITE_CMD interface words

The following table explains the meanings of the variables associated with the WRITE_CMD whose action was specified in the word ACTION_CMD. These variables are updated by a WRITE_CMD (IODDT_VAR1).

Standard symbol	Type	Access	Meaning	Address
ERROR_CMD	INT	RW	Error during WRITE_CMD	%MWr.m.c.19
RETURN_CMD_1	DINT	RW	Return 1 of the function	%MDr.m.c.20
RETURN_CMD_2	REAL	RW	Return 2 of the function	%MFr.m.c.22
RETURN_CMD_3	REAL	RW	Return 3 of the function	%MFr.m.c.24
ACTION_CMD	INT	RW	Action to be carried out	%MWr.m.c.26
PARAM_CMD_1	DINT	RW	Parameter 1	%MDr.m.c.27
PARAM_CMD_2	DINT	RW	Parameter 2	%MDr.m.c.29
PARAM_CMD_3	REAL	RW	Parameter 3	%MFr.m.c.31
PARAM_CMD_4	REAL	RW	Parameter 4	%MFr.m.c.33

READ_PARAM, WRITE_PARAM interface words

The table below explains the meanings of the parameters which can be accessed using the READ_PARAM and WRITE_PARAM functions for channels 1 to 16. These variables are updated using READ_PARAM (IODDT_VAR1) or WRITE_PARAM (IODDT_VAR1). You can also use the SAVE_PARAM and RESTORE_PARAM functions.

Standard symbol	Type	Access	Meaning	Address
FUNCTION_VALIDATION	INT	RW	Word containing the selective validation bits	%MWr.m.c.35
ACCEL	REAL	RW	Acceleration value	%MFr.m.c.36
DECEL	REAL	RW	Deceleration value	%MFr.m.c.38
ACCEL_TYPE	INT	RW	Acceleration type	%MWr.m.c.40
IN_POSITION_BAND	REAL	RW	Value of the in-position band	%MFr.m.c.41
ENABLE_POSITION_BAND	REAL	RW	Value of the monitoring window	%MFr.m.c.43
ROLLOVER_MAX	REAL	RW	Maximum rollover	%MFr.m.c.45
ROLLOVER_MIN	REAL	RW	Minimum rollover	%MFr.m.c.47
ACCEL_MAX	REAL	RW	Maximum acceleration	%MFr.m.c.49
DECEL_MAX	REAL	RW	Maximum deceleration	%MFr.m.c.51
SPEED_MAX	REAL	RW	Maximum speed	%MFr.m.c.53
POSITION_MAX	REAL	RW	Maximum position	%MFr.m.c.55
POSITION_MIN	REAL	RW	Minimum position	%MFr.m.c.57
SCALE_NUMERATOR	REAL	RW	Scale factor enumerator (configuration of an independent axis and GetGearRatio function)	%MFr.m.c.59
SCALE_DENOMINATOR	REAL	RW	Scale factor denominator (configuration of an independent axis and GetGearRatio function)	%MFr.m.c.61
ACCEL_UNIT	INT	RW	Acceleration unit.	%MWr.m.c.63
SPEED_UNIT	INT	RW	Velocity unit	%MWr.m.c.64
POSITION_UNIT	INT	RW	Position unit	%MWr.m.c.65

Details concerning T_CSY_FOLLOW-type IODDT implicit exchange objects

At a Glance

The T_CSY_FOLLOW IODDT possesses implicit exchange objects, which are described below. This type of IODDT is applied to the TSX_CSY_84 module for channels 21 to 24.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_FOLLOW

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

List of implicit exchange input objects

The following table introduces the implicit exchange input objects of the T_CSY_FOLLOW IODDT which apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CH_ERROR	EBOOL	R	Channel error bit.	%lr.m.c.ERR
RAMPING	EBOOL	R	Indicates whether the axis is accelerating or decelerating	%lr.m.c.0
STEADY	EBOOL	R	The speed is steady	%lr.m.c.1
STOPPING	EBOOL	R	The movement is decelerating to a stop	%lr.m.c.2
PROFILE_END	EBOOL	R	The last profile command has been sent to the module	%lr.m.c.3
IN_POSITION	EBOOL	R	The axis is within the in-position band	%lr.m.c.4
AXIS_HOMING	EBOOL	R	The axis is homing. For an imaginary axis, this bit is inactive	%lr.m.c.5
AXIS_HOMED	EBOOL	R	The axis position reading is referenced off the home position	%lr.m.c.6
AXIS_NOT_FOLLOWING	EBOOL	R	The drive is not recognizing module commands	%lr.m.c.7
HOLDING	EBOOL	R	The axis is holding in wait position	%lr.m.c.8
RESUMING	EBOOL	R	The axis is moving after a hold	%lr.m.c.9
DRIVE_ENABLED	EBOOL	R	The drive is enabled	%lr.m.c.10
DRIVE_DIAG	EBOOL	R	The drive is performing a class 3 diagnostic	%lr.m.c.11
DRIVE_WARNING	EBOOL	R	The drive is performing a class 2 diagnostic	%lr.m.c.12
DRIVE_FLT	EBOOL	R	The drive is performing a class 1 diagnostic	%lr.m.c.13
DRIVE_DISABLED	EBOOL	R	The drive is disabled	%lr.m.c.14
AXIS_SUMMARY_FLT	EBOOL	R	Drive fault	%lr.m.c.15

Standard symbol	Type	Access	Meaning	Address
AXIS_COM_OK	EBOOL	R	Communication between the drive and the module is OK	%Ir.m.c.16
AXIS_IS_LINKED	EBOOL	R	The axis belongs to a set of axes	%Ir.m.c.17
AXIS_IN_CMD	EBOOL	R	The axis is active and can be controlled	%Ir.m.c.18
AXIS_AT_TARGET	EBOOL	R	The axis is within the in-position band for the target position	%Ir.m.c.20
AXIS_POS_LIMIT	EBOOL	R	The axis has reached the positive limit	%Ir.m.c.21
AXIS_NEG_LIMIT	EBOOL	R	The axis has reached the negative limit	%Ir.m.c.22
AXIS_WARNING	EBOOL	R	MotionWarning status returned by the drive	%Ir.m.c.23
AXIS_HOLD	EBOOL	R	The axis is stopped and waiting for a command	%Ir.m.c.28
AXIS_HALT	EBOOL	R	The axis has stopped	%Ir.m.c.29
AXIS_FASTSTOP	EBOOL	R	The axis has faststopped	%Ir.m.c.30
AXIS_READY	EBOOL	R	The axis is ready to respond to a command	%Ir.m.c.31
CONF_OK	EBOOL	R	The channel is configured	%Ir.m.c.32

List of implicit exchange output objects

The following table introduces the implicit exchange output objects of the T_CSY_FOLLOW IODDT that apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CONTROL_ACQUIRE	EBOOL	RW	control acquisition	%Qr.m.c.2
CONTROL_ENABLE	EBOOL	RW	Control enable	%Qr.m.c.10
CONTROL_FOLLOW	EBOOL	RW	Follow control for an axis or a set of follower axes	%Qr.m.c.11
CONTROL_RESUME	EBOOL	RW	Resumes control after a stop	%Qr.m.c.12
CONTROL_CLEAR_FLT	EBOOL	RW	Fault clear control	%Qr.m.c.15
ALLOW_ACQUIRE	EBOOL	RW	Acquisition enable control	%Qr.m.c.18
ALLOW_ENABLE	EBOOL	RW	Disable axis control	%Qr.m.c.26
ALLOW_FOLLOW	EBOOL	RW	Cancels follow control for an axis or a set of follower axes	%Qr.m.c.27
ALLOW_RESUME	EBOOL	RW	Authorizes a movement to continue after a stop using the HOLD command	%Qr.m.c.28
ALLOW_MOVE	EBOOL	RW	Authorizes a movement to continue after a stop using the HALT command	%Qr.m.c.29
ALLOW_NOT_FASTSTOP	EBOOL	RW	Control after a Faststop	%Qr.m.c.30
ALLOW_NOT_FLT	EBOOL	RW	Enables fault control	%Qr.m.c.31

Actual position

The following table introduces the implicit exchange input objects of the T_CSY_FOLLOW IODDT that apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
POSITION	REAL	R	Actual position	%lFr.m.c.0

Parameter report word

The following table introduces the implicit exchanges entry word for the T_CSY_FOLLOW IODDT that applies to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
PARAM_RPT	INT	R	Parameters report signaling a programming fault. The least significant byte contains the error code and the most significant byte contains the address in the registers of the field that triggered the error.	%lWr.m.c.2

Details concerning T_CSY_FOLLOW-type IODDT explicit exchange objects

At a Glance

This part introduces the explicit exchange objects of the T_CSY_FOLLOW IODDT that apply to the TSX CSY 84 module for channels 21 to 24. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_FOLLOW

Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

Indicators of explicit exchange execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: EXCH_STS (%MWr.m.c.0). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Read channel status words in progress	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjust parameters exchange in progress	%MWr.m.c.0.2
RECONF_IN_PROGR	BOOL	R	Module reconfiguring	%MWr.m.c.0.15

Explicit exchange report: EXCH_RPT

The following table introduces the report bits: EXCH_RPT (%MWr.m.c.1). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Channel status words read fault (1 = failure)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Fault during exchange of command parameters (1 = failure)	%MWr.m.c.1.1

Standard symbol	Type	Access	Meaning	Address
ADJ_ERR	BOOL	R	Fault during an exchange of adjustment parameters (1 = failure)	%MWr.m.c.1.2
RECONF_ERR	BOOL	R	Fault during channel reconfiguration (1 = failure)	%MWr.m.c.1.15

Channel fault word

The following table introduces the channel fault bits: `CH_FLT`. These variables are updated by a `READ_STS` (`IODDT_VAR1`).

Standard symbol	Type	Access	Meaning	Address
EXT_FLT0	BOOL	R	External fault 0: servodrive fault	%MWr.m.c.2.0
EXT_FLT1	BOOL	R	External fault 1: communication fault with the axis	%MWr.m.c.2.1
EXT_FLT2	BOOL	R	External fault 2	%MWr.m.c.2.3
INT_FLT	BOOL	R	Internal fault	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Configuration fault: different hardware and software configurations	%MWr.m.c.2.5
COM_FLT	BOOL	R	Communication fault	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Faulty project configuration, adjustment or command fault	%MWr.m.c.2.7
PROCESS_CONF	BOOL	R	Creation of move in progress object	%MWr.m.c.2.11
PROCESS_CONF_FAILED	BOOL	R	Configuration fault (except for channel 0)	%MWr.m.c.2.12

WRITE_CMD interface words

The following table explains the meanings of the variables associated with the `WRITE_CMD` whose action was specified in the word `ACTION_CMD`. These variables are updated by a `WRITE_CMD` (`IODDT_VAR1`).

Standard symbol	Type	Access	Meaning	Address
ERROR_CMD	INT	RW	Error during <code>WRITE_CMD</code>	%MWr.m.c.19
RETURN_CMD_1	DINT	RW	Return 1 of the function	%MDr.m.c.20
RETURN_CMD_2	REAL	RW	Return 2 of the function	%MFr.m.c.22
RETURN_CMD_3	REAL	RW	Return 3 of the function	%MFr.m.c.24
ACTION_CMD	INT	RW	Action to be carried out	%MWr.m.c.26
PARAM_CMD_1	DINT	RW	Parameter 1	%MDr.m.c.27
PARAM_CMD_2	DINT	RW	Parameter 2	%MDr.m.c.29

Standard symbol	Type	Access	Meaning	Address
PARAM_CMD_3	REAL	RW	Parameter 3	%MFr.m.c.31
PARAM_CMD_4	REAL	RW	Parameter 4	%MFr.m.c.33

READ_PARAM, WRITE_PARAM interface words

The table below explains the meanings of the parameters which can be accessed using the READ_PARAM and WRITE_PARAM functions for channels 21 to 24. These variables are updated using READ_PARAM (IODDT_VAR1) or WRITE_PARAM (IODDT_VAR1). You can also use the SAVE_PARAM and RESTORE_PARAM functions.

Standard symbol	Type	Access	Meaning	Address
MASTER_CHANNEL	INT	RW	Number of the master axis (1 to 16, N is not accessible)	%MWr.m.c.35
SLAVE_CHANNEL_1	INT	RW	Number of slave axis 1	%MWr.m.c.36
FOLL_DESCRIPTION_1	INT	RW	Description of slave axis 1. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions: bits 8, 9 and 10 set to zero = immediate startup bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position	%MWr.m.c.37
FOLL_WHERE_1	BOOL	R	0 = controller	%MWr.m.c.37.0
FOLL_TYPE_1	BOOL	R	0 = ratio mode 1 = Cam mode	%MWr.m.c.37.1
FOLL_POSITION_1	BOOL	R	0 = following the actual position 1 = following the commanded position	%MWr.m.c.37.2
FOLL_FOL_ON_HALT_1	BOOL	R	1 = stop the follower axis if master/slave link is removed	%MWr.m.c.37.3
FOLL_HALT_MASTER_1	BOOL	R	1 = stop the master in the event of a following error	%MWr.m.c.37.6
FOLL_BIAS_REMAIN_1	BOOL	R	1 = dynamic offset on position of master	%MWr.m.c.37.7
NUMERATOR_1	REAL	RW	numerator for slave axis 1	%MFr.m.c.38

Standard symbol	Type	Access	Meaning	Address
DENOMINATOR_1	REAL	RW	Denominator for slave axis 1	%MFr.m.c.40
TRIGGER_POSITION_1	REAL	RW	Value of trigger for slave axis 1	%MFr.m.c.42
SLAVE_CHANNEL_2	INT	RW	Number of slave axis 2	%MWr.m.c.44
FOLL_DESCRIPTION_2	INT	RW	Description of slave axis 2. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions: bits 8, 9 and 10 set to zero = immediate startup bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position	%MWr.m.c.45
FOLL_WHERE_2	BOOL	R	0 = controller	%MWr.m.c.45.0
FOLL_TYPE_2	BOOL	R	0 = ratio mode 1 = Cam mode	%MWr.m.c.45.1
FOLL_POSITION_2	BOOL	R	0= following the actual position 1 = following the commanded position	%MWr.m.c.45.2
FOLL_FOL_ON_HALT_2	BOOL	R	1 = stop the follower axis if master/slave link is removed	%MWr.m.c.45.3
FOLL_HALT_MASTER_2	BOOL	R	1 = stop the master in the event of a following error	%MWr.m.c.45.6
FOLL_BIAS_REMAIN_2	BOOL	R	1 = dynamic offset on position of master	%MWr.m.c.45.7
NUMERATOR_2	REAL	RW	numerator for slave axis 2	%MFr.m.c.46
DENOMINATOR_2	REAL	RW	Denominator for slave axis 2	%MFr.m.c.48
TRIGGER_POSITION_2	REAL	RW	Value of trigger for slave axis 2	%MFr.m.c.50
SLAVE_CHANNEL_3	INT	RW	Number of slave axis 3	%MWr.m.c.52

Standard symbol	Type	Access	Meaning	Address
FOLL_DESCRIPTION_3	INT	RW	Description of slave axis 3. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions: bits 8, 9 and 10 set to zero = immediate startup bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position	%MWr.m.c.53
FOLL_WHERE_3	BOOL	R	0 = controller	%MWr.m.c.53.0
FOLL_TYPE_3	BOOL	R	0 = ratio mode 1 = Cam mode	%MWr.m.c.53.1
FOLL_POSITION_3	BOOL	R	0 = following the actual position 1 = following the commanded position	%MWr.m.c.53.2
FOLL_FOL_ON_HALT_3	BOOL	R	1 = stop the follower axis if master/slave link is removed	%MWr.m.c.53.3
FOLL_HALT_MASTER_3	BOOL	R	1 = stop the master in the event of a following error	%MWr.m.c.53.6
FOLL_BIAS_REMAIN_3	BOOL	R	1 = dynamic offset on position of master	%MWr.m.c.53.7
NUMERATOR_3	REAL	RW	numerator for slave axis 3	%MFr.m.c.54
DENOMINATOR_3	REAL	RW	Denominator for slave axis 3	%MFr.m.c.56
TRIGGER_POSITION_3	REAL	RW	Value of trigger for slave axis 3	%MFr.m.c.58
SLAVE_CHANNEL_4	INT	RW	Number of slave axis 4	%MWr.m.c.60

Standard symbol	Type	Access	Meaning	Address
FOLL_DESCRIPTION_4	INT	RW	Description of slave axis 4. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions: bits 8, 9 and 10 set to zero = immediate startup bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position	%MWr.m.c.61
FOLL_WHERE_4	BOOL	R	0 = controller	%MWr.m.c.61.0
FOLL_TYPE_4	BOOL	R	0 = ratio mode 1 = Cam mode	%MWr.m.c.61.1
FOLL_POSITION_4	BOOL	R	0= following the actual position 1 = following the commanded position	%MWr.m.c.61.2
FOLL_FOL_ON_HALT_4	BOOL	R	1 = stop the follower axis if master/slave link is removed	%MWr.m.c.61.3
FOLL_HALT_MASTER_4	BOOL	R	1 = stop the master in the event of a following error	%MWr.m.c.61.6
FOLL_BIAS_REMAIN_4	BOOL	R	1 = dynamic offset on position of master	%MWr.m.c.61.7
NUMERATOR_4	REAL	RW	numerator for slave axis 4	%MFr.m.c.62
DENOMINATOR_4	REAL	RW	Denominator for slave axis 4	%MFr.m.c.64
TRIGGER_POSITION_4	REAL	RW	Value of trigger for slave axis 4	%MFr.m.c.66
SLAVE_CHANNEL_5	INT	RW	Number of slave axis 5	%MWr.m.c.68

Standard symbol	Type	Access	Meaning	Address
FOLL_DESCRIPTION_5	INT	RW	Description of slave axis 5. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions: bits 8, 9 and 10 set to zero = immediate startup bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position	%MWr.m.c.69
FOLL_WHERE_5	BOOL	R	0 = controller	%MWr.m.c.69.0
FOLL_TYPE_5	BOOL	R	0 = ratio mode 1 = Cam mode	%MWr.m.c.69.1
FOLL_POSITION_5	BOOL	R	0 = following the actual position 1 = following the commanded position	%MWr.m.c.69.2
FOLL_FOL_ON_HALT_5	BOOL	R	1 = stop the follower axis if master/slave link is removed	%MWr.m.c.69.3
FOLL_HALT_MASTER_5	BOOL	R	1 = stop the master in the event of a following error	%MWr.m.c.69.6
FOLL_BIAS_REMAIN_5	BOOL	R	1 = dynamic offset on position of master	%MWr.m.c.69.7
NUMERATOR_6	REAL	RW	numerator for slave axis 6	%MFr.m.c.70
DENOMINATOR_6	REAL	RW	Denominator for slave axis 6	%MFr.m.c.72
TRIGGER_POSITION_6	REAL	RW	Value of trigger for slave axis 6	%MFr.m.c.74
SLAVE_CHANNEL_6	INT	RW	Number of slave axis 6	%MWr.m.c.76

Standard symbol	Type	Access	Meaning	Address
FOLL_DESCRIPTION_6	INT	RW	Description of slave axis 6. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions: bits 8, 9 and 10 set to zero = immediate startup bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position	%MWr.m.c.77
FOLL_WHERE_6	BOOL	R	0 = controller	%MWr.m.c.77.0
FOLL_TYPE_6	BOOL	R	0 = ratio mode 1 = Cam mode	%MWr.m.c.77.1
FOLL_POSITION_6	BOOL	R	0= following the actual position 1 = following the commanded position	%MWr.m.c.77.2
FOLL_FOL_ON_HALT_6	BOOL	R	1 = stop the follower axis if master/slave link is removed	%MWr.m.c.77.3
FOLL_HALT_MASTER_6	BOOL	R	1 = stop the master in the event of a following error	%MWr.m.c.77.6
FOLL_BIAS_REMAIN_6	BOOL	R	1 = dynamic offset on position of master	%MWr.m.c.77.7
NUMERATOR_6	REAL	RW	numerator for slave axis 6	%MFr.m.c.78
DENOMINATOR_6	REAL	RW	Denominator for slave axis 6	%MFr.m.c.80
TRIGGER_POSITION_6	REAL	RW	Value of trigger for slave axis 6	%MFr.m.c.82

Details concerning T_CSY_COORD-type IODDT implicit exchange objects

At a Glance

The T_CSY_COORD IODDT possesses implicit exchange objects, which are described below. This type of IODDT is applied to the TSX_CSY_84 module for channels 17 to 20.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_COORD

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

List of implicit exchange input objects

The following table introduces the implicit exchange input objects of the T_CSY_COORD IODDT which apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CH_ERROR	EBOOL	R	Channel error bit.	%lr.m.c.ERR
RAMPING	EBOOL	R	Indicates whether the axis is accelerating or decelerating	%lr.m.c.0
STEADY	EBOOL	R	The speed is steady	%lr.m.c.1
STOPPING	EBOOL	R	The movement is decelerating to a stop	%lr.m.c.2
PROFILE_END	EBOOL	R	The last profile command has been sent to the module	%lr.m.c.3
IN_POSITION	EBOOL	R	The axis is within the in-position band	%lr.m.c.4
AXIS_HOMING	EBOOL	R	The axis is homing. For an imaginary axis, this bit is inactive	%lr.m.c.5
AXIS_HOMED	EBOOL	R	The axis position reading is referenced off the home position	%lr.m.c.6
AXIS_NOT_FOLLOWING	EBOOL	R	The drive is not recognizing module commands	%lr.m.c.7
HOLDING	EBOOL	R	The axis is holding in wait position	%lr.m.c.8
RESUMING	EBOOL	R	The axis is moving after a hold	%lr.m.c.9
DRIVE_ENABLED	EBOOL	R	The drive is enabled	%lr.m.c.10
DRIVE_DIAG	EBOOL	R	The drive is performing a class 3 diagnostic	%lr.m.c.11
DRIVE_WARNING	EBOOL	R	The drive is performing a class 2 diagnostic	%lr.m.c.12
DRIVE_FLT	EBOOL	R	The drive is performing a class 1 diagnostic	%lr.m.c.13
DRIVE_DISABLED	EBOOL	R	The drive is disabled	%lr.m.c.14
AXIS_SUMMARY_FLT	EBOOL	R	Drive fault	%lr.m.c.15

Standard symbol	Type	Access	Meaning	Address
AXIS_COM_OK	EBOOL	R	Communication between the drive and the module is OK	%lr.m.c.16
AXIS_IS_LINKED	EBOOL	R	The axis belongs to a set of axes	%lr.m.c.17
AXIS_IN_CMD	EBOOL	R	The axis is active and can be controlled	%lr.m.c.18
AXIS_AT_TARGET	EBOOL	R	The axis is within the in-position band for the target position	%lr.m.c.20
AXIS_POS_LIMIT	EBOOL	R	The axis has reached the positive limit	%lr.m.c.21
AXIS_NEG_LIMIT	EBOOL	R	The axis has reached the negative limit	%lr.m.c.22
AXIS_WARNING	EBOOL	R	MotionWarning status returned by the drive	%lr.m.c.23
AXIS_HOLD	EBOOL	R	The axis is stopped and waiting for a command	%lr.m.c.28
AXIS_HALT	EBOOL	R	The axis has stopped	%lr.m.c.29
AXIS_FASTSTOP	EBOOL	R	The axis has faststopped	%lr.m.c.30
AXIS_READY	EBOOL	R	The axis is ready to respond to a command	%lr.m.c.31
CONF_OK	EBOOL	R	The channel is configured	%lr.m.c.32

List of implicit exchange output objects

The following table introduces the implicit exchange output objects of the T_CSY_COORD IODDT that apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CONTROL_ACQUIRE	EBOOL	RW	control acquisition	%Qr.m.c.2
CONTROL_ENABLE	EBOOL	RW	Control enable	%Qr.m.c.10
CONTROL_FOLLOW	EBOOL	RW	Follow control for an axis or a set of follower axes	%Qr.m.c.11
CONTROL_RESUME	EBOOL	RW	Resumes control after a stop	%Qr.m.c.12
CONTROL_CLEAR_FLT	EBOOL	RW	Fault clear control	%Qr.m.c.15
ALLOW_ACQUIRE	EBOOL	RW	Acquisition enable control	%Qr.m.c.18
ALLOW_ENABLE	EBOOL	RW	Disable axis control	%Qr.m.c.26
ALLOW_FOLLOW	EBOOL	RW	Control to cancel following for an axis or a set of follower axes	%Qr.m.c.27
ALLOW_RESUME	EBOOL	RW	Authorizes a movement to continue after a stop using the HOLD command	%Qr.m.c.28
ALLOW_MOVE	EBOOL	RW	Authorizes a movement to continue after a stop using the HALT command	%Qr.m.c.29
ALLOW_NOT_FASTSTOP	EBOOL	RW	Control after a Faststop	%Qr.m.c.30
ALLOW_NOT_FLT	EBOOL	RW	Enable fault control	%Qr.m.c.31

Parameter report word

The following table introduces the implicit exchange input word for the
T_CSY_COORD IODDT that applies to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
PARAM_RPT	INT	R	Parameters report signaling a programming fault. The least significant byte contains the error code and the most significant byte contains the address in the registers of the field that triggered the error.	%IW.r.m.c.2

Details concerning T_CSY_COORD-type IODDT explicit exchange objects

At a Glance

This part introduces the explicit exchange objects of the T_CSY_COORD IODDT that apply to the TSX CSY 84 module for channels 17 to 20. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_COORD

Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

Indicators of explicit exchange execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: EXCH_STS (%MWr.m.c.0). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Read channel status words in progress	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjust parameters exchange in progress	%MWr.m.c.0.2
RECONF_IN_PROGR	BOOL	R	Module reconfiguring	%MWr.m.c.0.15

Explicit exchange report: EXCH_RPT

The following table introduces the report bits: `EXCH_RPT` (%MWr.m.c.1). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Channel status words read fault (1 = failure)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Fault during exchange of command parameters (1 = failure)	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Fault during an exchange of adjustment parameters (1 = failure)	%MWr.m.c.1.2
RECONF_ERR	BOOL	R	Fault during channel reconfiguration (1 = failure)	%MWr.m.c.1.15

Channel fault word

The following table introduces the channel fault bits: `CH_FLT`. These variables are updated by a `READ_STS` (IODDT_VAR1).

Standard symbol	Type	Access	Meaning	Address
EXT_FLT0	BOOL	R	External fault 0: servodrive fault	%MWr.m.c.2.0
EXT_FLT1	BOOL	R	External fault 1: communication fault with the axis	%MWr.m.c.2.1
EXT_FLT2	BOOL	R	External fault 2	%MWr.m.c.2.3
INT_FLT	BOOL	R	Internal fault	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Configuration fault: different hardware and software configurations	%MWr.m.c.2.5
COM_FLT	BOOL	R	Communication fault	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Faulty project configuration, adjustment or command fault	%MWr.m.c.2.7
PROCESS_CONF	BOOL	R	Creation of move in progress object	%MWr.m.c.2.11
PROCESS_CONF_FAILED	BOOL	R	Configuration fault (except for channel 0)	%MWr.m.c.2.12

WRITE_CMD interface words

The following table explains the meanings of the variables associated with the `WRITE_CMD` whose action was specified in the word `ACTION_CMD`. These variables are updated by a `WRITE_CMD (IODDT_VAR1)`.

Standard symbol	Type	Access	Meaning	Address
<code>ERROR_CMD</code>	INT	RW	Error during <code>WRITE_CMD</code>	%MWr.m.c.19
<code>RETURN_CMD_1</code>	DINT	RW	Return 1 of the function	%MDr.m.c.20
<code>RETURN_CMD_2</code>	REAL	RW	Return 2 of the function	%MFr.m.c.22
<code>RETURN_CMD_3</code>	REAL	RW	Return 3 of the function	%MFr.m.c.24
<code>ACTION_CMD</code>	INT	RW	Action to be carried out	%MWr.m.c.26
<code>PARAM_CMD_1</code>	DINT	RW	Parameter 1	%MDr.m.c.27
<code>PARAM_CMD_2</code>	DINT	RW	Parameter 2	%MDr.m.c.29
<code>PARAM_CMD_3</code>	REAL	RW	Parameter 3	%MFr.m.c.31
<code>PARAM_CMD_4</code>	REAL	RW	Parameter 4	%MFr.m.c.33
<code>PARAM_CMD_5</code>	REAL	RW	Parameter 5	%MFr.m.c.35
<code>PARAM_CMD_6</code>	REAL	RW	Parameter 6	%MFr.m.c.37
<code>PARAM_CMD_7</code>	REAL	RW	Parameter 7	%MFr.m.c.39
<code>PARAM_CMD_8</code>	REAL	RW	Parameter 8	%MFr.m.c.41
<code>PARAM_CMD_9</code>	REAL	RW	Parameter 9	%MFr.m.c.43
<code>PARAM_CMD_10</code>	REAL	RW	Parameter 10	%MFr.m.c.45
<code>PARAM_CMD_11</code>	REAL	RW	Parameter 11	%MFr.m.c.47
<code>PARAM_CMD_12</code>	REAL	RW	Parameter 12	%MFr.m.c.49
<code>PARAM_CMD_13</code>	REAL	RW	Parameter 13	%MFr.m.c.51
<code>PARAM_CMD_14</code>	REAL	RW	Parameter 14	%MFr.m.c.53
<code>PARAM_CMD_15</code>	REAL	RW	Parameter 15	%MFr.m.c.55
<code>PARAM_CMD_16</code>	REAL	RW	Parameter 16	%MFr.m.c.57
<code>PARAM_CMD_17</code>	REAL	RW	Parameter 17	%MFr.m.c.59
<code>PARAM_CMD_18</code>	REAL	RW	Parameter 18	%MFr.m.c.61

Details concerning T_CSY_CAM-type IODDT implicit exchange objects

At a Glance

The T_CSY_CAM IODDT possesses implicit exchange objects, which are described below. This type of IODDT is applied to the TSX_CSY_84 module for channels 25 to 31.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_CAM

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

List of implicit exchange input objects

The following table introduces the implicit exchange input objects for the T_CSY_CAM IODDT which apply to the TSX_CSY_84 module.

Standard symbol	Type	Access	Meaning	Address
CH_ERROR	EBOOL	R	Channel error bit.	%I.r.m.c.ERR

Details concerning T_CSY_CAM-type IODDT explicit exchange objects

At a Glance

This part introduces the explicit exchange objects of the T_CSY_CAM IODDT that apply to the TSX CSY 84 module for channels 25 to 31. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: IODDT_VAR1 of the type T_CSY_CAM

Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

Indicators of explicit exchange execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: EXCH_STS (%MWr.m.c.0). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Read channel status words in progress	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjust parameters exchange in progress	%MWr.m.c.0.2
TRF_IN_PROGR	BOOL	R	TRF_RECIPe function in progress	%MWr.m.c.0.3
RECONF_IN_PROGR	BOOL	R	Module reconfiguring	%MWr.m.c.0.15

Explicit exchange report: EXCH_RPT

The following table introduces the report bits: EXCH_RPT (%MWr.m.c.1). These variables are automatically updated by the system upon each explicit exchange.

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Channel status words read fault (1 = failure)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Fault during exchange of command parameters (1 = failure)	%MWr.m.c.1.1

Standard symbol	Type	Access	Meaning	Address
ADJ_ERR	BOOL	R	Fault during an exchange of adjustment parameters (1 = failure)	%MWr.m.c.1.2
TRF_ERR	BOOL	R	Fault while TRF_RECIPE function is in progress	%MWr.m.c.1.3
RECONF_ERR	BOOL	R	Fault during channel reconfiguration (1 = failure)	%MWr.m.c.1.15

Channel fault word

The following table introduces the channel fault bits: `CH_FLT`. These variables are updated by a `READ_STS (IODDT_VAR1)`.

Standard symbol	Type	Access	Meaning	Address
EXT_FLT0	BOOL	R	External fault 0: servodrive fault	%MWr.m.c.2.0
EXT_FLT1	BOOL	R	External fault 1: communication fault with the axis	%MWr.m.c.2.1
EXT_FLT2	BOOL	R	External fault 2	%MWr.m.c.2.3
INT_FLT	BOOL	R	Internal fault	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Configuration fault: different hardware and software configurations	%MWr.m.c.2.5
COM_FLT	BOOL	R	Communication fault	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Faulty project configuration, adjustment or command fault	%MWr.m.c.2.7
PROCESS_CONF	BOOL	R	Creation of move in progress object	%MWr.m.c.2.11
PROCESS_CONF_FAILED	BOOL	R	Configuration fault (except for channel 0)	%MWr.m.c.2.12

Objects of the TRF_RECIPE function

The following table introduces the objects associated with the `TRF_RECIPE` function. These objects are automatically updated by the system each time the `TRF_RECIPE` function is used.

Standard symbol	Type	Access	Meaning	Address
ERROR_TRF	INT	R	Read error for the <code>TRF_RECIPE</code> function	%MWr.m.c.3
RETURN_TRF_1	DINT	R	Return 1 of the <code>TRF_RECIPE</code> function	%MDr.m.c.4
RETURN_TRF_2	REAL	R	Return 2 of the <code>TRF_RECIPE</code> function	%MFr.m.c.6
RETURN_TRF_3	REAL	R	Return 3 of the <code>TRF_RECIPE</code> function	%MFr.m.c.8
ACTION_TRF	INT	R	Action to be carried out by the <code>TRF_RECIPE</code> function	%MWr.m.c.10

Standard symbol	Type	Access	Meaning	Address
PARAM_TRF_1	DINT	R	Parameter 1 of the TRF_RECIPE function	%MDr.m.c.11
PARAM_TRF_2	DINT	R	Parameter 2 of the TRF_RECIPE function	%MDr.m.c.13
PARAM_TRF_3	REAL	R	Parameter 3 of the TRF_RECIPE function	%MFr.m.c.15
PARAM_TRF_4	REAL	R	Parameter 4 of the TRF_RECIPE function	%MFr.m.c.17

WRITE_CMD interface words

The following table explains the meanings of the variables associated with the WRITE_CMD whose action was specified in the word ACTION_CMD. These variables are updated by a WRITE_CMD (IODDT_VAR1).

Standard symbol	Type	Access	Meaning	Address
ERROR_CMD	INT	RW	Error during WRITE_CMD	%MWr.m.c.19
RETURN_CMD_1	DINT	RW	Return 1 of the function	%MDr.m.c.20
RETURN_CMD_2	REAL	RW	Return 2 of the function	%MFr.m.c.22
RETURN_CMD_3	REAL	RW	Return 3 of the function	%MFr.m.c.24
ACTION_CMD	INT	RW	Action to be carried out	%MWr.m.c.26
PARAM_CMD_1	DINT	RW	Parameter 1	%MDr.m.c.27
PARAM_CMD_2	DINT	RW	Parameter 2	%MDr.m.c.29
PARAM_CMD_3	REAL	RW	Parameter 3	%MFr.m.c.31
PARAM_CMD_4	REAL	RW	Parameter 4	%MFr.m.c.33

16.3 The IODDT Type T_GEN_MOD Applicable to All Modules

Details of the Language Objects of the T_GEN_MOD-Type IODDT

At a Glance

All the modules of Premium PLCs have an associated IODDT of type T_GEN_MOD.

Observations

- In general, the meaning of the bits is given for bit status 1. In specific cases an explanation is given for each status of the bit.
- Not all bits are used.

List of Objects

The table below presents the objects of the IODDT:

Standard symbol	Type	Access	Meaning	Address
MOD_ERROR	BOOL	R	Module error bit	%I.r.m.MOD.ERR
EXCH_STS	INT	R	Module exchange control word.	%MWr.m.MOD.0
STS_IN_PROGR	BOOL	R	Reading of status words of the module in progress.	%MWr.m.MOD.0.0
EXCH_RPT	INT	R	Exchange report word.	%MWr.m.MOD.1
STS_ERR	BOOL	R	Fault when reading module status words.	%MWr.m.MOD.1.0
MOD_FLT	INT	R	Internal error word of the module.	%MWr.m.MOD.2
MOD_FAIL	BOOL	R	Internal error, module failure.	%MWr.m.MOD.2.0
CH_FLT	BOOL	R	Faulty channel(s).	%MWr.m.MOD.2.1
BLK	BOOL	R	Terminal block fault.	%MWr.m.MOD.2.2
CONF_FLT	BOOL	R	Hardware or software configuration fault.	%MWr.m.MOD.2.5
NO_MOD	BOOL	R	Module missing or inoperative.	%MWr.m.MOD.2.6
EXT_MOD_FLT	BOOL	R	Internal error word of the module (Fipio extension only).	%MWr.m.MOD.2.7
MOD_FAIL_EXT	BOOL	R	Internal fault, module unserviceable (Fipio extension only).	%MWr.m.MOD.2.8
CH_FLT_EXT	BOOL	R	Faulty channel(s) (Fipio extension only).	%MWr.m.MOD.2.9
BLK_EXT	BOOL	R	Terminal block fault (Fipio extension only).	%MWr.m.MOD.2.10

Standard symbol	Type	Access	Meaning	Address
CONF_FLT_EXT	BOOL	R	Hardware or software configuration fault (Fipio extension only).	%MWr.m.MOD.2.13
NO_MOD_EXT	BOOL	R	Module missing or inoperative (Fipio extension only).	%MWr.m.MOD.2.14

17.1 **List of Accessible Functions**

Aim of this Section

This section recalls the different accessible functions for the TSX CSY 84/164 modules.

What Is in This Section?

This section contains the following topics:

Topic	Page
Accessible Functions for SERCOS® Channel 0	454
Accessible Functions for a Real Axis	456
Accessible Functions for an Imaginary Axis	460
Accessible Functions for a Remote Axis	463
Accessible Functions for a Group of Co-ordinated Axes	465
Accessible Functions for a Group of Slave Axes	467
Accessible Functions for a Cam Profile	469

Accessible Functions for SERCOS® Channel 0

Adjustment Functions

The following table lists the adjustment functions:

Function	Code	Feedback	Parameters
GetOpticalPower	1547	Feedback 2: Percentage	None
SetOpticalPower	2547	None	Parameter 3: Percentage

Diagnostic Functions

The following table lists the diagnostic functions:

Function	Code	Feedback	Parameters
GetActualPhase	550	Feedback 1: Phase	None
GetAxisID	523	Feedback 1: Identifier	None
GetCommandedPhase	1545	Feedback 1: Phase	None
GetCombinedControl	1534	Feedback 1: Movement control bits	None
GetControl	1525	Feedback 1: Movement control bits	None
GetLoopDiagnosticMode	1546	Feedback 1: Diagnostic mode	None
GetMotionFault	5510	Feedback 1: List of faults	None
GetMotionWarning	5511	Feedback 1: List of warnings	None
GetNumberOfDrivesInRing	548	Return 1: No. of servodrives	None
GetNumberInSet	541	Feedback 1: No. of axes	None
GetSercosAddress	549	Feedback 1: Address	Parameter 1: Axis
GetStatus	5509	Feedback 1: Status bits	None
IsLoopUp	543	Feedback 1: 0 / 1	None
SetCommandedPhase	2545	None	Parameter 1: Phase
SetLoopDiagnosticMode	2546	None	Parameter 1: Diagnostic mode

Read/Write Functions for IDN Parameters

The following table lists the read/write functions for IDN parameters for SERCOS® servodrives:

Function	Code	Feedback	Parameters
GetIDN_S	1556	Feedback 1: SERCOS® parameter	Parameter 1: Identifier
GetIDN_P	1557	Feedback 1: SERCOS® parameter	Parameter 1: Identifier
GetIDN_US	1558	Feedback 1: SERCOS® parameter	Parameter 1: Identifier
GetIDN_UP	1559	Feedback 1: SERCOS® parameter	Parameter 1: Identifier
SetIDN_S	2556	None	Parameter 1: Identifier Parameter 2: SERCOS® parameter
SetIDN_P	2557	None	Parameter 1: Identifier Parameter 2: SERCOS® parameter
SetIDN_US	2558	None	Parameter 1: Identifier Parameter 3: SERCOS® parameter
SetIDN_UP	2559	None	Parameter 1: Identifier Parameter 3: SERCOS® parameter

Accessible Functions for a Real Axis

Movement Control Functions

The following table lists the movement control functions:

Function	Code	Feedback	Parameters
ForcedHome	6039	None	None
Home	6034	None	Parameter 3: Direction Parameter 4: Speed
Unhome	6038	None	None

Movement Functions

The following table lists the movement functions:

Function	Code	Feedback	Parameters
MoveImmed	513	None	Parameter 1: Type Parameter 3: Position Parameter 4: Speed
MoveImmed_G30	554	None	Parameter 1: Type Parameter 2: No. of points Parameter 3: Pathpoints Parameter 4: Approach speed
MoveQueue	520	None	Parameter 1: Type Parameter 3: Position Parameter 4: Speed
EnableRealTimeCtrlBit	2564	None	Parameter 1: Enable (1) Disable (0)

Adjustment Functions

The following table lists the adjustment functions:

Function	Code	Feedback	Parameters
DisableRollover	412	None	None
EnableRollover	411	None	None
GetAccel	1041	Feedback 2: Acceleration	None
GetAccelMax	1116	Feedback 2: Acceleration	None
GetActualSpeed	5065	Feedback 1: Speed	None
GetDecel	1042	Feedback 2: Deceleration	None

Function	Code	Feedback	Parameters
GetDecelMax	1117	Feedback 2: Deceleration	None
GetDefaultSpeed	1065	Feedback 1: Speed	None
GetChangeACCDECMove	1172	Feedback 1: 1: FCt enabled: PARAM_CMD_2 acknowledged in opcodes 513 and 520. 0: FCt disabled.	None
GetEnableMode	1524	Feedback 1: Mode	None
GetEnablePositionBand	1538	Feedback 2: Position	None
GetInPositionBand	1035	Feedback 2: Position	None
GetPositionLimit	1505	Feedback 2: Position	Parameter 1: Direction
GetRolloverLimit	1539	Feedback 2: Position	Parameter 1: Direction
GetSpeedLimit	1066	Feedback 2: Speed	None
GetSpeedOverride	1513	Feedback 2: Percentage	None
GetUnrolled CommandedPosition	547	Feedback 2: Position	None
GetUnrolledPosition	546	Feedback 2: Position	None
SetAccel	2041	None	Parameter 3: Acceleration
SetAccelMax	2116	None	Parameter 3: Acceleration
SetDecel	2042	None	Parameter 3: Deceleration
SetDecelMax	2117	None	Parameter 3: Deceleration
SetChangeACCDECMove	2172	None	Parameter 1: 1: FCt enabled: PARAM_CMD_2 acknowledged in opcodes 513 and 520. 0: FCt disabled.
SetDefaultSpeed	2065	None	Parameter 1: Speed
SetEnableMode	2524	None	Parameter 1: Mode
SetEnablePositionBand	2538	None	Parameter 3: Position
SetInPositionBand	2035	None	Parameter 3: Position
SetPosition	2053	None	Parameter 3: Position

Function	Code	Feedback	Parameters
SetPositionLimit	2505	None	Parameter 1: Direction Parameter 3: Position
SetRolloverLimit	2539	None	Parameter 1: Direction Parameter 3: Position
SetSpeedLimit	2066	None	Parameter 3: Speed
SetSpeedOverride	2513	None	Parameter 3: Percentage
EnableDriveWarning	5512	None	Parameter 1 = 13
DisableDriveWarning	5513	None	Parameter 1 = 13

Diagnostic Functions

The following table lists the diagnostic functions:

Function	Code	Feedback	Parameters
GetAxisId	523	Feedback 1: Identifier	None
GetCommandedPosition	1053	Feedback 2: Position	None
GetCombinedControl	1534	Feedback 1: Movement control bits	None
GetControl	1525	Feedback 1: Movement control bits	None
GetMotionFault	5510	Feedback 1: List of faults	None
GetMotionWarning	5511	Feedback 1: List of warnings	None
GetMoveQueueLength	9510	Return 1: Length	None
GetNumberInSet	541	Feedback 1: No. of axes	None
GetSercosAddress	549	Feedback 1: Address	None

Configuration Functions

The following table lists the configuration functions:

Function	Code	Feedback	Parameters
GetAccelType	1540	Feedback 1: Acceleration type	None
GetGearRatio	1500	Feedback 2: Numerator Feedback 3: Denominator	None
SetAccelType	2540	None	Parameter 1: Acceleration type
SetGearRatio	2500	None	Parameter 3: Numerator Parameter 4: Denominator

Read/Write Functions for IDN Parameters

The following table lists the read/write functions for IDN parameters for SERCOS® servodrives:

Function	Code	Feedback	Parameters
GetIDN_S	1556	Feedback 1: SERCOS® parameter	Parameter 1: Identifier
GetIDN_P	1557	Feedback 1: SERCOS® parameter	Parameter 1: Identifier
GetIDN_US	1558	Feedback 1: SERCOS® parameter	Parameter 1: Identifier
GetIDN_UP	1559	Feedback 1: SERCOS® parameter	Parameter 1: Identifier
SetIDN_S	2556	None	Parameter 1: Identifier Parameter 2: SERCOS® parameter
SetIDN_P	2557	None	Parameter 1: Identifier Parameter 2: SERCOS® parameter
SetIDN_US	2558	None	Parameter 1: Identifier Parameter 3: SERCOS® parameter
SetIDN_UP	2559	None	Parameter 1: Identifier Parameter 3: SERCOS® parameter

Accessible Functions for an Imaginary Axis

Movement Control Functions

The following table lists the movement control functions:

Function	Code	Feedback	Parameters
ForcedHome	6039	None	None
Home	6034	None	Parameter 3: Direction Parameter 4: Speed
Unhome	6038	None	None

Movement Functions

The following table lists the movement functions:

Function	Code	Feedback	Parameters
MoveImmed	513	None	Parameter 1: Type Parameter 3: Position Parameter 4: Speed
MoveImmed_G30	554	None	Parameter 1: Type Parameter 2: No. of points Parameter 3: Pathpoints Parameter 4: Approach speed
MoveQueue	520	None	Parameter 1: Type Parameter 3: Position Parameter 4: Speed

Adjustment Functions

The following table lists the adjustment functions:

Function	Code	Feedback	Parameters
DisableRollover	412	None	None
EnableRollover	411	None	None
GetAccel	1041	Feedback 2: Acceleration	None
GetAccelMax	1116	Feedback 2: Acceleration	None
GetActualSpeed	5065	Feedback 1: Speed	None
GetDecel	1042	Feedback 2: Deceleration	None
GetDecelMax	1117	Feedback 2: Deceleration	None

Function	Code	Feedback	Parameters
GetDefaultSpeed	1065	Feedback 1: Speed	None
GetChangeACCDECMove	1172	Feedback 1: 1: FCt enabled: PARAM_CMD_2 acknowledged in opcodes 513 and 520. 0: FCt disabled.	None
GetEnablePositionBand	1538	Feedback 2: Position	None
GetPositionLimit	1505	Feedback 2: Position	Parameter 1: Direction
GetRolloverLimit	1539	Feedback 2: Position	Parameter 1: Direction
GetSpeedLimit	1066	Feedback 2: Speed	None
GetSpeedOverride	1513	Feedback 2: Percentage	None
GetUnrolled CommandedPosition	547	Feedback 2: Position	None
GetUnrolledPosition	546	Feedback 2: Position	None
SetAccel	2041	None	Parameter 3: Acceleration
SetAccelMax	2116	None	Parameter 3: Acceleration
SetDecel	2042	None	Parameter 3: Deceleration
SetDecelMax	2117	None	Parameter 3: Deceleration
SetDefaultSpeed	2065	None	Parameter 1: Speed
SetChangeACCDECMove	2172	None	Parameter 1: 1: FCt enabled: PARAM_CMD_2 acknowledged in opcodes 513 and 520. 0: FCt disabled.
SetEnablePositionBand	2538	None	Parameter 3: Position
SetPosition	2053	None	Parameter 3: Position
SetPositionLimit	2505	None	Parameter 1: Direction Parameter 3: Position
SetRolloverLimit	2539	None	Parameter 1: Direction Parameter 3: Position
SetSpeedLimit	2066	None	Parameter 3: Speed
SetSpeedOverride	2513	None	Parameter 3: Percentage

Diagnostic Functions

The following table lists the diagnostic functions:

Function	Code	Feedback	Parameters
GetAxisId	523	Feedback 1: Axis identifier	None
GetCommandedPosition	1053	Feedback 2: Position	None
GetCombinedControl	1534	Feedback 1: Movement control bits	None
GetControl	1525	Return 1: Movement control bits	None
GetMotionFault	5510	Feedback 1: List of faults	None
GetMotionWarning	5511	Return 1: List of warnings	None
GetMoveQueueLength	9510	Feedback 1: Length	None
GetNumberInSet	541	Feedback 1: No. of axes	None

Configuration Functions

The following table lists the configuration functions:

Function	Code	Feedback	Parameters
GetAccelType	1540	Feedback 1: Acceleration type	None
GetGearRatio	1500	Feedback 2: Numerator Feedback 3: Denominator	None
SetAccelType	2540	None	Parameter 1: Acceleration type
SetGearRatio	2500	None	Parameter 3: Numerator Parameter 4: Denominator

Accessible Functions for a Remote Axis

Movement Control Functions

The following table lists the movement control functions:

Function	Code	Feedback	Parameters
ForcedHome	6039	None	None
Home	6034	None	Parameter 3: Direction Parameter 4: Speed
Unhome	6038	None	None

Adjustment Functions

The following table lists the adjustment functions:

Function	Code	Feedback	Parameters
DisableRollover	412	None	None
EnableRollover	411	None	None
GetRolloverLimit	1539	Feedback 2: Position	Parameter 1: Direction
GetUnrolledPosition	546	Feedback 2: Position	None
SetPosition	2053	None	Parameter 3: Position
SetRolloverLimit	2539	None	Parameter 1: Direction Parameter 3: Position

Diagnostic Functions

The following table lists the diagnostic functions:

Function	Code	Feedback	Parameters
GetAxisId	523	Feedback 1: Axis identifier	None
GetCombinedControl	1534	Return 1: Movement control bits	None
GetControl	1525	Return 1: Movement control bits	None
GetMotionFault	5510	Feedback 1: List of faults	None
GetMotionWarning	5511	Return 1: List of warnings	None
GetNumberInSet	541	Feedback 1: No. of axes	None
GetSercosAddress	549	Feedback 1: Address	Parameter 1: Axis

Configuration Functions

The following table lists the configuration functions:

Function	Code	Feedback	Parameters
GetGearRatio	1500	Feedback 2: Numerator Feedback 3: Denominator	None
SetGearRatio	2500	None	Parameter 3: Numerator Parameter 4: Denominator

Accessible Functions for a Group of Co-ordinated Axes

Movement Control Functions

The following table lists the movement control functions:

Function	Code	Feedback	Parameters
Home	6034	None	Parameter 3: Direction Parameter 4: Speed

Movement Functions

The following table lists the movement functions. In the MoveImmed and MoveQueue functions, use n registers (starting at Parameter 3, where n is the number of axes in the co-ordinated group) to specify the positions for all the axes in the co-ordinated group. Then use n additional registers (starting at Parameter n+3) to specify the speeds for all the axes.

Function	Code	Feedback	Parameters
MoveImmed	513	None	Parameter 1: Type Parameter (2 + n): Position Parameter (n + 3) -> (2n + 3): Speed
MoveQueue	520	None	Parameter 1: Type Parameter (2 + n): Position Parameter (n + 3) -> (2n + 3): Speed

Adjustment Functions

The following table lists the adjustment functions:

Function	Code	Feedback	Parameters
GetActualSpeed	5065	Feedback 1: Speed	None
GetDefaultSpeed	1065	Feedback 1: Speed	None
GetSpeedOverride	1513	Feedback 2: Percentage	None
SetDefaultSpeed	2065	None	Parameter 1: Speed
SetSpeedOverride	2513	None	Parameter 3: Percentage

Diagnostic Functions

The following table lists the diagnostic functions:

Function	Code	Feedback	Parameters
GetAxisId	523	Feedback 1: Axis identifier	None
GetCombinedControl	1534	Return 1: Movement control bits	None
GetControl	1525	Return 1: Movement control bits	None
GetMotionFault	5510	Feedback 1: List of faults	None
GetMotionWarning	5511	Return 1: List of warnings	None
GetMoveQueueLength	9510	Feedback 1: Length	None

Accessible Functions for a Group of Slave Axes

Adjustment Functions

The following table lists the adjustment functions:

Function	Code	Feedback	Parameters
DisableRollover	412	None	None
EnableRollover	411	None	None
GetAbsFollowerBias	1526	Feedback 1: Position	Parameter 1: Axis identifier
GetCamProfile	1530	Feedback 1: Profile identifier	Parameter 1: Axis identifier
GetFollowerBias	1527	Feedback 2: Position	Parameter 1: Axis identifier
GetFollowerMode	1529	Feedback 1: Follower mode	Parameter 1: Axis identifier
GetFollowerRatio	1114	Feedback 2: Numerator Feedback 3: Denominator	Parameter 1: Axis identifier
GetMasterOffset	1532	Feedback 2: Position	Parameter 1: Axis identifier
GetSpeedOverride	1513	Feedback 2: Percentage	None
GetMasterTriggerPosition	1531	Feedback 2: Position	Parameter 1: Axis identifier
SetFollowerRatio	2114	None	Parameter 1: Axis identifier Parameter 3: Numerator Parameter 4: Denominator
SetMasterOffset	2532	None	Parameter 1: Axis identifier Parameter 3: Position
SetMasterTriggerPosition	2531	None	Parameter 1: Axis identifier Parameter 3: Position
SetSpeedOverride	2513	None	Parameter 3: Percentage

Diagnostic Functions

The following table lists the diagnostic functions:

Function	Code	Feedback	Parameters
GetAxisId	523	Feedback 1: Axis identifier	None
GetCombinedControl	1534	Return 1: Movement control bits	None
GetControl	1525	Return 1: Movement control bits	None
GetMotionFault	5510	Feedback 1: List of faults	None
GetMotionWarning	5511	Return 1: List of warnings	None
GetNumberInSet	541	Feedback 1: No. of axes	None

Configuration Functions

The following table lists the configuration functions:

Function	Code	Feedback	Parameters
GetMaster	1528	Feedback 1: Axis identifier	None
SetFollowerConfig	420	None	Parameter 1: Axis identifier Parameter 2: Follower mode Parameter 3: Numerator or profile ID Parameter 4: Denominator
SetMaster	2528	None	Parameter 1: Axis identifier

Accessible Functions for a Cam Profile

Adjustment Functions

The following table lists the adjustment functions:

Function	Code	Feedback	Parameters
Length	534	Feedback 1: Table length	None
LookUpFollowerPosition	537	Feedback 2: Slave position	Parameter 3: Master position

Configuration Functions

The following table lists the configuration functions:

Function	Code	Feedback	Parameters
GetCoord	532	Feedback 1: Master position Feedback 2: Slave position	Parameter 1: Table index
GetInterpType	530	Feedback 1: Interp. type	None
SetCoord	533	None	Parameter 1: Table index Parameter 3: Master position Parameter 4: Slave position
SetInterpType	531	None	Parameter 1: Interp. type

Functions specific to the TSX CSY 164



Subject of this Part

This part describes the new functions specific to the TSX CSY 164 module.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
18	Compatibility	473
19	Configuration of the TSX CSY 164 module	475
20	Functions specific to the TSX CSY 164	481
21	Performances of the TSX CSY 164	489

Compatibility

Software Compatibility

- To develop a project using the TSX CSY 164 modules, the Unity Pro software must be minimum version SV = 2.1,
- For a TSX CSY 84 module, a program is compatible with a TSX CSY 164 module after reconfiguring the Unity Pro application.

Configuration of the TSX CSY 164 module

19

Aim of this chapter

This chapter describes the configuration screens of module TSX CSY 164 and some of the different functions it can perform: "SERCOS® functions," "real axis", "imaginary axis", etc.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Configuration of the TSX CSY 164 module	476
Configuring channels	477

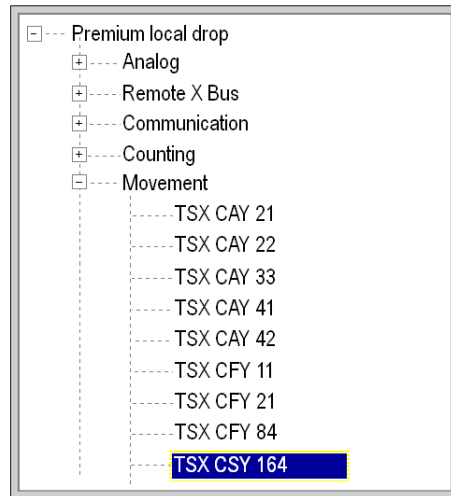
Configuration of the TSX CSY 164 module

Introduction

In Configuration mode, it is possible to define the operating characteristics for each TSX CSY 164 module path.

Addition of a Module

In the "Movement" family "Add module" dialog box, select the TSX CSY 164 module:



See the chapter "Configuration of the TSX CSY 84 module" (*see page 117*).

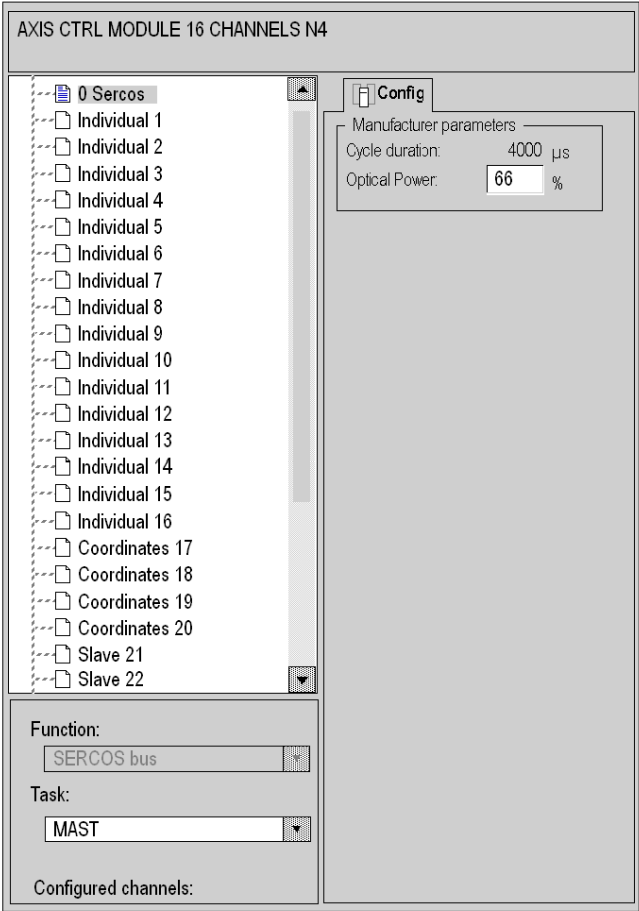
Configuring channels

Introduction

Configuring a channel involves defining the parameters for the application-specific function associated with this channel:

- Channel 0: SERCOS® function (*see page 122*),
- Channels 1 to 16: independent axis (real axis, imaginary axis, or remote axis),
- Channels 17 to 20: coordinated axes group (*see page 129*),
- Channels 21 to 24: slave axes group (*see page 130*),
- Channels 25 to 31: cam profile (*see page 133*).

Illustration



Configuration of an Independent Axis

An independent axis (channels 1 to 16) can support real axis functions, imaginary axis functions, and external setpoints:

- The configuration of a real axis enables control of a physical axis (which uses a servodrive). In this case, a certain consistency must be ensured between the parameters entered in the TSX CSY 164 module configuration screen and those defined during servodrive configuration.
- An imaginary axis can, for example, act as a master axis in a slave axes group. All of the parameters for this type of axis are defined in the configuration screens for the axis control modules.
- A remote axis can be used to upload an item of remote position data to the module. Configuring an external measurement involves configuring a real axis or an imaginary axis in which only the position information is valid.

Configuration of Coordinated Axes

Channels 17 to 20 (coordinated axes group) on the TSX CSY 84 allow the following services to be used:

- WRITE_CMD,
- READ_STS.

On the TSX CSY 164, a new command has been added:

- MOD_PARAM (*see page 482*).

Configuration of Slave Axes

Channels 21 to 24 (slave axes group) on the TSX CSY 84 allow the following services to be used:

- READ_PARAM
- WRITE_PARAM
- WRITE_CMD (except Move),
- SAVE_PARAM
- RESTORE_PARAM
- READ_STS.

On the TSX CSY 164, a new command has been added:

- MOD_PARAM (*see page 482*).

Functions specific to the TSX CSY 164



Aim of This Chapter

This chapter describes the new functions associated with the TSX CSY 164 module.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Dynamic reconfiguration of groups	482
Monitoring Function of the Position Deviations Between Axes	484

Dynamic reconfiguration of groups

Description

A new command is available for the **slave** and **coordinated axes groups** in order to change, by program, the member axes in the group.

This command will have the same effects as a reconfiguration in online mode (channel emergency stop).

It will allow:

- The master of a slave group to be changed on the fly
- The member axes in the group to be changed

- **Syntax of the MOD_PARAM instruction:**

MOD_PARAM %CHr.m,i: writing %Mx adjustment words of group i, of the module situated at the r.m address (rack number, position in the rack).

e.g.: writing slave mode parameters: writing of slave group 24 situated at slot address 4.

```
IF %M38 THEN
    MOD_PARAM (%CH4.24,0,0,0,0);
    RESET %M38;
END_IF;
```

- **Check exchanges:**

The two following bits can be used to check the writing in the configuration words:

%MWr.m,i,0.15: this bit is set to 1 when the exchange is in process. It is reset to 0 when the exchange has been completed.

%MWr.m,i,1.15: this bit is set to 1 when the exchange is erroneous (Parameters out of range, etc.)

If you are running Unity Pro, this function is written as follows:

- **Mod_Param (% CHr.m.c,0,0,0,0);**

The parameters within parentheses are always equal to 0.

This function will use the parameter words (%MW) as an interface.

Associated Language Objects

This function will use the parameter %MW as an interface. When this function is called, the %MW will be recopied into the constants (%KW) and the group will then carry out a reconfiguration.

For slave groups:

Source memory words	Source memory word mnemonics	Destination constant words	Comments
%MWx.y.36	SLAVE_CHANNEL_1	%KWx.y.3	Number of slave 1 axis of slave group
%MWx.y.37	FOLL_DESCRIPTION_1	%KWx.y.4	Definition of slave 1 axis of slave group
%MWx.y.44	SLAVE_CHANNEL_2	%KWx.y.11	Number of slave 2 axis of slave group
%MWx.y.45	FOLL_DESCRIPTION_2	%KWx.y.12	Definition of slave 2 axis of slave group
%MWx.y.52	SLAVE_CHANNEL_3	%KWx.y.19	Number of slave 3 axis of slave group
%MWx.y.53	FOLL_DESCRIPTION_3	%KWx.y.20	Definition of slave 3 axis of slave group
%MWx.y.60	SLAVE_CHANNEL_4	%KWx.y.27	Number of slave 4 axis of slave group
%MWx.y.61	FOLL_DESCRIPTION_4	%KWx.y.28	Definition of slave 4 axis of slave group
%MWx.y.68	SLAVE_CHANNEL_5	%KWx.y.35	Number of slave 5 axis of slave group
%MWx.y.69	FOLL_DESCRIPTION_5	%KWx.y.36	Definition of slave 5 axis of slave group
%MWx.y.76	SLAVE_CHANNEL_6	%KWx.y.43	Number of slave 6 axis of slave group
%MWx.y.77	FOLL_DESCRIPTION_6	%KWx.y.44	Definition of slave 6 axis of slave group

For coordinated axes groups: adding the Mod_Param function requires the addition of the following %mW to the interface language:

Source memory words	Destination constant words	Comments
%MWx.y.63	%KWx.y.4	Number of the axis member 0 in the coordinated axes group
%MWx.y.64	%KWx.y.5	Number of the axis member 1 in the coordinated axes group
%MWx.y.65	%KWx.y.6	Number of the axis member 2 in the coordinated axes group
%MWx.y.66	%KWx.y.7	Number of the axis member 3 in the coordinated axes group
%MWx.y.67	%KWx.y.8	Number of the axis member 4 in the coordinated axes group
%MWx.y.68	%KWx.y.9	Number of the axis member 5 in the coordinated axes group
%MWx.y.69	%KWx.y.10	Number of the axis member 6 in the coordinated axes group
%MWx.y.70	%KWx.y.11	Number of the axis member 7 in the coordinated axes group

When this function is called, the words (%MW) will be recopied into the constants (%KW) and the group will then carry out a reconfiguration.

The contents of these %MW gives the application-specific channel number of all the member axes in the coordinated axes group.

Monitoring Function of the Position Deviations Between Axes

Description

The monitoring function of the position deviations between axes allows any deviations between the independent axes (real, imaginary, or external setpoint) of the module to be monitored.

This function can be instantiated several times with lists of different axes.

The function will be ensured by 3 opcodes:

- Launch of monitoring
- Rereading of the function parameters
- Stop monitoring

Launch of Monitoring

Enables monitoring to start by calling a function **TRF_RECIPÉ (opcode 26200)** on channel 0.

When the function is called, the module takes a snapshot of the different positions of the axes to be monitored and then:

- If the 2 axes have a movement differing by more than the **alarm threshold**, the module triggers a warning on the two faulty axes.
- If the 2 axes have a movement differing by more than the **alarm threshold**, the module triggers a fault on the two faulty axes, stops all of the list axes and triggers a warning on the non-faulty axes.

Object	Type	Explanation	Comment
%MF r.m.0.15	Floating point	Instance number	Parameter of the TRF_RECIPÉ, converted to an integer by the module
%MF i	Floating point	Alarm threshold	In position units of the reference axis
%MF i+2	Floating point	Default threshold	In position units of the reference axis
%MW i+4	Word	Reference axis	
%MW i+5	Word	Axis 1	
%MF i+6	Floating point	Axis 1 numerator	
%MF i+8	Floating point	Axis 1 denominator	
.. ..	..	..	
%MW i+5+5(N-1)	Word	Axis N	

Object	Type	Explanation	Comment
%MF i+5+5(N-1)+1	Floating point	Axis N numerator	
%MF i+5+5(N-1)+3	Floating point	Axis N denominator	

Example:

Monitoring of axes 8, 12 and 7:

- Axis 8 is configured in mm
- Axis 12 is configured in °. It is set to 4° when axis 8 is set to 1 mm.
- Axis 7 is configured in rotations. It is set to 1 rotation when axis 8 is set to 10 mm.

The alarm threshold is 4 mm.

The default threshold is 10 mm.

```

IF %MW104.0.0: X3 = 0 THEN (* no TRF_RECIPE in progress *)
%MW104.0.10:= 26200;      (* Action 26200 = GAP_CONTROL *)
%MF104.0.15:= 1.0;       (* Instance number *)
%MF150:=4.0;             (* Alarm threshold *)
%MF152:= 10.0;           (* Default threshold *)
%MW154:= 8;              (* Reference axis *)
%MW155:= 12;             (* Second axis to monitor *)
%MF156:= 4.0.0;          (* Numerator of the ratio between
the 2nd axis and the reference axis *)
%MF158:= 1.0;            (* Denominator of the ratio between
the 2nd axis and the reference axis *)
%MW160:=7.0;             (* Third axis to monitor *)
%MF161:= 1.0;            (* Numerator of the ratio between
the 3rd axis and the reference axis *)
%MF163:= 10.0;           (* Denominator of the ratio
between the 3rd axis and the reference axis *)
TRF_RECIPE %CH 104.0 (15,150);
END_IF;

```

An alarm will be triggered if axis 8 moves 5 mm when axis 12 has only moved 4°.

A fault will be triggered if axis 8 moves 20 mm when axis 12 has only moved 4°.

Refusal of the TRF_RECIPÉ 26200: error on the word %MWr.m.0.3

- The size of the data (first parameter of TRF_RECIPÉ) is incorrect => error code 23
- Invalid instance number = > error code 10. Instance number must be between 0 and 7 or a maximum of 8 simultaneous instances.
- The instance is already active => error code
- The alarm threshold is over the default threshold => error code 65
- One of the thresholds is negative => error code 66
- One of the axes to be monitored corresponds to an axis that has not been configured as a real, imaginary or external setpoint axis => error code 37
- One of the denominators is zero => error code 67.

Rereading of the Function Parameters

Enables monitoring parameters to be reread for a given instance, by calling a function **TRF_RECIPÉ (opcode 16200)** on channel 0.

The result of the function can be read on the universal %M.

Object	Type	Explanation	Comment
%MF i	Floating point	Alarm threshold	In position units of the reference axis
%MF i+2	Floating point	Default threshold	In position units of the reference axis
%MW i+4	Word	Reference axis	
%MW i+5	Word	Axis 1	
%MF i+6	Floating point	Axis 1 numerator	
%MF i+8	Floating point	Axis 1 denominator	
.. .. .	..	..	
%MW i+5+5(N-1)	Word	Axis N	
%MF i+5+5(N-1)+1	Floating point	Axis N numerator	
%MF i+5+5(N-1)+3	Floating point	Axis N denominator	

Example:

```

IF %MW104.0.0: X3 = 0 THEN (* No TRF_RECIPE in progress *)
%MW104.0.10:= 16200;      (* Action 26200 = GAP_CONTROL *)
%MF104.0.15:= 1.0        (* Instance to be reread*)
TRF_RECIPE %CH 104.0 (15,150);
END_IF;

```

Refusal of the TRF_RECIPE 16200: error on the word %MWr.m.0.3

- The size of the data (first parameter of TRF_RECIPE) is incorrect => error code 23
- Invalid instance number => error code 10. Instance number must be between 0 and 7 or a maximum of 8 simultaneous instances.
- The instance is not active => error code 68.

Stop Monitoring

Enables monitoring of a given instance to be deleted, by calling a function **TRF_RECIPE (opcode 14200)** on channel 0.

Example:

```

IF %MW104.0.0: X3 = 0 THEN (* no TRF_RECIPE in progress *)
%MW104.0.10:= 14200;      (* Action 14200 = GAP_CONTROL *)
%MF104.0.15:= 1.0        (* Instance to be deleted*)
TRF_RECIPE %CH 104.0 (0,0);
END_IF;

```

Refusal of the TRF_RECIPE 14200: error on the word %MWr.m.0.3

- Invalid instance number => error code 10. Instance number must be between 0 and 7 or a maximum of 8 simultaneous instances.
- The instance is not active => error code 68.

Check of the Monitoring Activity

Allows a check to be made on whether an instance is active or inactive, by calling a function TRF_RECIPÉ (opcode 16201) on channel 0.

The result of the function can be read on a universal %MW.

TRF_RECIPÉ %CH4.0(1,150)

with 1= length and 150 = offset return.

Each bit in the first byte is the image of the activity of the corresponding instance (example: Bit 0 => Instance 0, etc....)

- 1=> instance active,
- 0=> instance inactive.

Example: Is the instance active?

```
IF %M9 AND NOT %MW4.0:X3 THEN
%MW4.0.10:=16201;
TRF_RECIPÉ %CH4.0(1,150);
RESET %M9;
END_IF;
```

The result can be accessed on the %MW150 word.

If %MW150 = 3 the 0 and 1 instances are active.

Refusal:error on the word %MW_{r.m,i},3:

- If the length to be read is not 1==> Error code 22,
- If the length is greater than the number of words (%MW) configured ==> Error code 19.

Performances of the TSX CSY 164 Module

At a Glance

The performances of the module are:

- SERCOS® bus cycle time:
 - SERCOS® tick equals 4 ms by default.
- Processing of an immediate command = 2 SERCOS® ticks
- Processing of a queued command = 2 SERCOS® ticks * Number of queued commands
- Processing of Remote measurement = 2 SERCOS® ticks + 1 SERCOS® tick for acknowledgment of the reference command on the slave
- Monitoring period of the deviations between axes: 20 ms independent of the Tick,
- Modification on the fly of the cycle time via the Write-cycle command to be chosen from the following:
 - SERCOS® Tick at 2 ms,
 - SERCOS® Tick at 3 ms,
 - SERCOS® Tick at 4 ms,

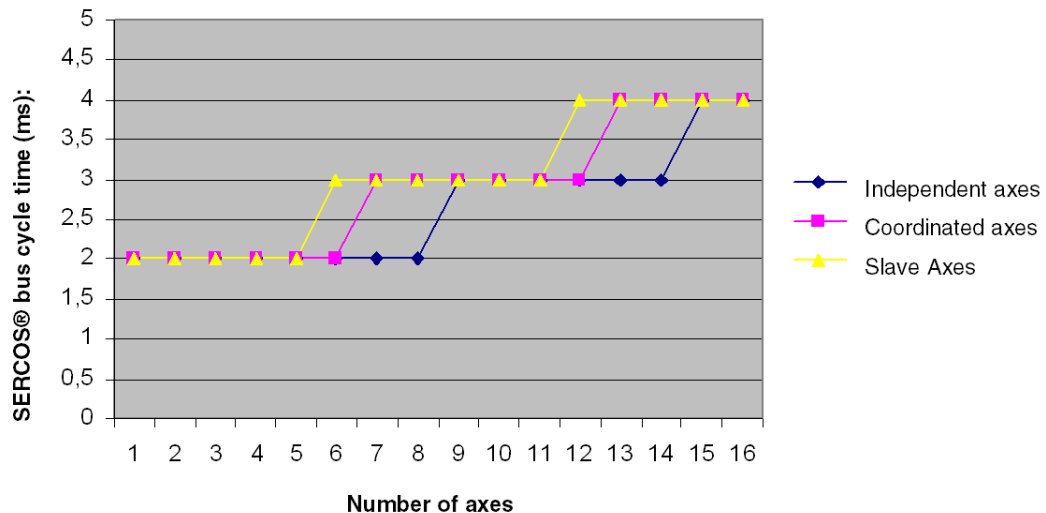
with acknowledgement during forcing in phase 0 of the SERCOS® ring.

The SERCOS® Tick is defined according to the number of axes configured and the type of applications (See the SERCOS® Cycle Time Selection Guide).

NOTE: The Lexium carries out an interpolation every 250 µs.

SERCOS® Cycle Time Selection Guide:

Performances of the TSX CSY 164

**Test conditions:**

NOTE: The performances shown were achieved under certain conditions guaranteeing a time buffer of approximately 25% of the SERCOS® CPU time. The Period of the MAST task is configured at 20ms. Transmission of the motion commands via MOVE_QUEUE is carried out once every 10 PLC cycles.

As a consequence,

- for the "independent axes" configuration: a MOVE_QUEUE command is sent to all axes every 200ms
- For the "slave axes" configuration, a MOVE_QUEUE command is sent to the master every 200ms
- For the "coordinated axes" configuration, a MOVE_QUEUE command is sent to all groups of coordinated axes every 200ms.

A WRITE_PARAM command is sent during this time period, in order to carry out a parameter adjustment.

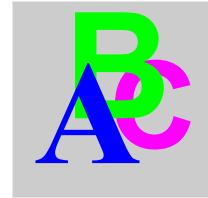
Modification of Cycle Time using a Program

```

(* change of the SERCOS® cycle time => %MW30 = 2000 or 3000 or
4000 *)
IF %M30 THEN      %MW101.0.26:=2565;
    %MD101.0.27:=%MW30;
    IF NOT %MW101.0:X1 THEN
        WRITE_CMD %CH101.0;
        RESET %M30;
        SET %M31;
    END_IF;
END_IF;
(* transition into SERCOS® phase 0 *)
IF %M31 THEN      %MW101.0.26:=2545;
    %MD101.0.27:=0;
    IF NOT %MW101.0:X1 THEN
        WRITE_CMD %CH101.0;
        RESET %M31;
        SET %M32;
    END_IF;
END_IF;
(* New transition into SERCOS® phase 4 *)
IF %M32 THEN      %MW101.0.26:=2545;
    %MD101.0.27:=4;
    IF NOT %MW101.0:X1 THEN
        WRITE_CMD %CH101.0;
        RESET %M32;
    END_IF;
END_IF;
(* Here, we can expect transition to phase 4 (OPCODE 550 result
in %MD101.0.20) then acknowledge the fault caused by the
transition to phase 0 *)

```

Index



0-9

990MCO000xxx, 88

A

ABS_MOVE, 189

adjusting, 231

ALLOW bits, 172

C

channel data structure for all modules

 T_GEN_MOD, 450

channel data structure for SERCOS modules

 T_CSY_CAM, 408

 T_CSY_CMD, 408

 T_CSY_COORD, 408

 T_CSY_FOLLOW, 408

 T_CSY_IND, 408

 T_CSY_RING, 408

 T_CSY_TRF, 408

configuring, 115

CONT_MOVE, 189

CONTROL bits, 172

cycle times (ticks)

 TSXCSY164, 489

 TSXCSY84, 382

D

debugging, 321

diagnosing, 74

diagnostics

 IsLoopUp, 346

E

error codes, 359

F

fault management, 359

function codes, 454

functions, 99

H

Home, 182

I

IDN, 312

INCR_MOVE, 189

installing, 69

M

manual mode, 179

MOD_PARAM, 482

motion status bits, 138

Movelmmed, 192, 226

movement control functions

 CONTROL bits, 172

 Home, 182

movement functions, *188*
 ABS_MOVE, *189*
 CONT_MOVE, *189*
 INCR_MOVE, *189*
 MoveImmed, *192, 226*
 MoveQueue, *194*
MoveQueue, *194*

O

opcodes, *454*

P

parameter settings, *386*
performances
 TSXCSY164, *489*
 TSXCSY84, *382*

Q

quick start, *33*

R

READ_PARAM, *400*
READ_STS, *406*
reference points, *184*
RESTORE_PARAM, *401*

S

SAVE_PARAM, *401*

T

T_GEN_MOD, *450*
TRF_RECIPE
 cam profiles, *304*
 general, *244*
TSXCSY164, *61*
TSXCSY84, *61*
TSXCSY85, *61*

W

wiring accessories, *87*
WRITE_CMD, *402*
WRITE_PARAM, *400*