PL7 Junior/Pro

Premium PLC Applications
Movement Control with TSX CSY 84 / 164
SERCOS® Module
Application Setup Manual

07/2008 eng

35009577.01

www.schneider-electric.com

Schneider Electric

Document Set

At a Glance

This manual is made up of 8 volumes:

- Vol. 1
  - Common Application-specific Functions
  - Discrete Application
  - AS-i Implementation
  - Operator Dialog Application
- Vol. 2
  - Counting Application
- Vol. 3
  - Axis Command Application
- Vol. 4
  - Step-by-step Axis Command Application
- Vol. 5
  - Electronic Cam Application
- Vol. 6
  - SERCOS® Application
- Vol. 7
  - Analog Application
  - PID Control Application
  - Weighing Application
- Vol. 8
  - Process Control Application

Table of Contents



About the Book 13

Part I The SERCOS® package in Premium PLCs 15
At a Glance 15

Chapter 1 SERCOS® Architecture 17
At a glance 17
Introduction 18
Introduction to SERCOS® architecture 20
Development of the TSX CSY •4 Modules 22
Limitations of PL7 Services According to Module Version 25

Chapter 2 Setup methodology 27
At a glance 27
Introduction 28
Implementing an independent axis 30
Implementation of a co-ordinated or slave axis group 32

Chapter 3 Initiation Example: TSX CSY 84 module 33
At a Glance 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
Configuration of the TSX CSY 84 module 40
Symbolization of I/O variables 43
Symbolization of Memory Variables 44
Programming the Preliminary Processing 45
Grafcet Programming 46
Transition Programming 47
Actions Programming 49
Programming the Post-Processing 53
Program transfer 57
Adjusting the Parameters 58

Introduction to the Debug Mode. 61

Part II Motion control common to TSX CSY 84 / 164 modules . 63
At a Glance 63

Chapter 4 Functional description of the TSX CSY 84 / 164 modules . . . 65
At a Glance 65
TSX CSY 84 / 164 module functions 66
"SERCOS®" function 68
"Real Axis" Function 69
"Imaginary Axis" Function 70
"Remote Axis" Function 71
Concept of the Group. 73
"Coordinated Axis Group" Function 75
"Slave Axis Group" Function 78
"Cam Profile" Function. 81

Chapter 5 Configuration of the TSX CSY 84 module 83
At a Glance 83
Presymbolization 84
Module Configuration. 85
Configuration of the Channels 88
Configuration of the SERCOS® Function (channel 0). 89
Configuring an Independent Axis (Channels 1 to 12) 90
Configuration of a Remote Axis (Channels 13 to 16). 95
Configuration of a set of coordinated axes (channels 17 à 20) 97
Configuration of a set of follower axes (channels 21 to 24). 99
CamProfile configuration 103

Chapter 6 Programming the TSX CSY 84 / 164 modules 107
At a Glance 107
6.1 Motion Status Bits 109
At a glance. 109
Bit RAMPING (%Ixy.i.0). 111
STEADY Bit (%Ixy.i.1) 112
STOPPING Bit (%Ixy.i.2) 113
PROFILE_END Bit (%Ixy.i.3). 114
IN_POSITION Bit (%Ixy.i.4) 115
AXIS_HOMING bit (%Ixy.i.5) 116
AXIS_HOMED bit (%Ixy.i.6). 117
AXIS_NOT_FOLLOWING bit (%Ixy.i.7). 118
HOLDING bit (%Ixy.i.8) 119
RESUMING bit (%Ixy.i.9). 120
DRIVE_ENABLED bit (%Ixy.i.10) 121
DRIVE_DIAG bit (%Ixy.i.11) 122
DRIVE_WARNING bit (%Ixy.i.12) 123

DRIVE_FAULT bit (%Ixy.i.13) 124
DRIVE_DISABLED bit (%Ixy.i.14) 125
AXIS_SUMMARY_FAULT bit (%Ixy.i.15) 126
AXIS_COMM_OK bit (%Ixy.i.16) 127
AXIS_IS_LINKED bit (%Ixy.i.17) 128
AXIS_AT_TARGET bit (%Ixy.i.20) 129
AXIS_POS_LIMIT bit (%Ixy.i.21) 130
AXIS_NEG_LIMIT bit (%Ixy.i.22) 131
AXIS_IN_COMMAND bit (%Ixy.i.18) 132
AXIS_WARNING bit (%Ixy.i.23) 133
BIAS_REMAIN Bit (%Ixy.i.24) 134
AXIS_MANUAL_MODE BIT (%Ixy.i.25) 135
DRIVE_REALTIME_BIT1 bit (%Ixy.i.26) 136
DRIVE_REALTIME_BIT2 bit (%Ixy.i.27) 137
AXIS_HOLD bit (%Ixy.i.28) 138
AXIS_HALT bit (%Ixy.i.29) 139
AXIS_FASTSTOP bit (%Ixy.i.30) 140
AXIS_READY bit (%Ixy.i.31) 141
CONF_OK Bit (%Ixy.i.32) 142
6.2 Motion control functions 143
At a glance 143
Movement Control Functions: General 144
Management of ALLOW and CONTROL bits 145
MovementControl Bits Modifiable on Edge 146
Control Bits Modifiable on Change of State 149
Status Bits and Words 150
Manual Mode 151
Set_Functional_Mode Function 153
Home Function 154
Reference Point with the Home function 156
ForcedHomed Function 157
Unhome Function 158
SetPosition Function 159
6.3 Move Functions 160
At a glance 160
About Movement Functions 161
MoveType Data Type 163
MoveImmed Function 164
MoveQueue Function 166
GetMoveQueueLength Function 168
GetSpeedOverride and SetSpeedOverride Function 169
EnableRealTimeCtrlBit Function 170
6.4 Motion Function Following Deactivation of the Sequence 171
At a Glance 171
Overview of the Function 172

"MoveImmed" Function . . . 173
"MoveQueue" Function . . . 176
6.5 Functions of actual speed/position. . . . 180
Introduction . . . 180
GetActualSpeedFunction. . . . 181
GetCommandSpeedFunction . . . 182
Modulo counter service . . . 183
GetUnrolledPosition Function . . . 184
GetUnrolledCommandedPosition Function . . . 185
GetCommandedPosition Function. . . . 186
6.6 Following Functions. . . . 187
Introduction . . . 187
About Following Functions. . . . 188
"Target sequence activation" function . . . 190
"Estimation of Remaining Time Before Activation of Sequence" Function 192
Sequence stop (Unlink) of the slave group set off by a trigger 194

Chapter 7 Adjustment of TSX CSY 84 / 164 modules. . . . 197

At a Glance . . . 197
7.1 SERCOS® Module Adjustment: General . . . 198
Introduction . . . 198
Description of the Adjustment Screens of SERCOS® Module: General 199
Adjustment Principles . . . 202
7.2 Adjustment of the Parameters of a Real, Imaginary or Remote Axis. . . . 203
Introduction . . . 203
Description of the Adjustment Screen of a SERCOS® Module: Real or imaginary axes . . . 204
Description of the Adjustment Screen of a SERCOS® Module: Remote Axes . . . 207
Description of the TRF_RECIPE Screen for Real Axes. . . . 209
TRF_RECIPE. . . . 212
TRF_RECIPE Examples . . . 214
Speed Parameter: DefaultSpeed. . . . 215
Speed Parameter: SpeedLimit. . . . 216
About acceleration/deceleration parameters . . . 217
Acceleration/Deceleration Parameter: Accel . . . 218
Acceleration/Deceleration Parameter: Decel. . . . 219
Acceleration/Deceleration Parameter: AccelMax. . . . 220
Acceleration/Deceleration Parameter: DecelMax . . . 221
Acceleration/Deceleration Parameter: AccelType . . . 222
AccelerationType Data type. . . . 224
Resolution Parameters: GearRatio . . . 225
Position Limit and Modulo Parameter: PositionLimit. . . . 227
About the Modulo. . . . 229
Position Limit and Modulo Parameter: RolloverLimit . . . 231
Position Limit and Modulo Parameter: EnableRollover . . . 232

About the In-Position Band 233
Position Limit Parameter: InPositionBand 234
About Enable Mode 235
Position Limit Parameter: EnableMode 236
Behavior Following Axis Disable 237
7.3 Parameters of a Group of Slave Axes 238
At a Glance 238
Description of the Adjustment Screen of a SERCOS® Module: Slave Axes 239
About a Group of Slave Axes 241
SlaveMode Data 244
Slave Group Parameters: FollowerMode 245
Slave Group Parameter: FollowerConfig 246
Slave Group Parameter: FollowerRatio 248
About the Master Offset 251
Slave Group Parameters: MasterOffset 252
Slave Group Parameters: MasterTriggerPosition 254
Slave Group Parameters: FollowerBias 256
Slave Group Parameters: AbsFollowerBias 257
Propagation of Stops in a Slave Group 259
Alignment Function of the Emergency Stop Inclines 260
7.4 Cam Profile Adjustment 262
Introduction 262
About Cam Profiles 263
Cam Profile Parameters: Length 264
Cam Profile Parameters: LookUpFollowerPosition Function 265
Cam Profile Parameters: Coord 266
Cam Profile Parameter: InterpType 268
Cam Profile Parameter : ProfileId 269
Description of the TRF_RECIPE Screen for Cam Profiles 270
TRF_RECIPE Instruction 272
7.5 Sercos® ring parameters 274
Introduction 274
Description of the Adjustment Screen of a SERCOS® Module: SERCOS® ring 275
Adjustment graph for optical power 276
Sercos® ring parameters: OpticalPower 277
7.6 SERCOS® IDN Read/Write Functions 278
At a glance 278
Drive Parameters: IDN_S and IDN_P 279
Drive Parameters: IDN_US and IDN_UP 282
Standard SERCOS Identification Numbers (IDN) 285
IDNs for customized telegrams 288

Chapter 8 Debugging the TSX CSY 84 / 164 modules 289
At a Glance 289

Description of the debug screen: General . . . 290
Debug Screen User Interface . . . 293
Debug: Enable . . . 294
Debug: Diagnostics . . . 295
Debug: Movement . . . 299
Debug: Sending commands (Auto mode) . . . 301
Debug: Manual command (Manual mode) . . . 302
Debug: Reference . . . 303
Debug: Following . . . 304
Debug: Drive . . . 305
Debug: Position . . . 306
Diagnostics of the Module . . . 307
Channel Diagnostics . . . 308

Chapter 9 Diagnostics and Maintenance . . . 309
At a glance . . . 309
9.1 SERCOS® Phase Diagnostics Introduction . . . 310
At a glance . . . 310
About SERCOS® Phases Standard . . . 311
GetActualPhase Function . . . 312
GetCommandedPhase Function . . . 313
9.2 Reading Actual Data for the Ring Introduction . . . 314
At a glance . . . 314
GetSercosAddress Function . . . 315
GetNumberOfDrivesInRing function . . . 316
IsLoopUp function . . . 317
9.3 Axis Identification . . . 318
GetNumberInSet Function . . . 318
9.4 Status and default information . . . 319
At a glance . . . 319
Implicit Exchange Objects . . . 320
Faults Accessible via the GetMotionFault Command . . . 321
GetMotionFault Function . . . 323
Faults Accessible via the GetMotionWarning Command . . . 324
GetMotionWarning Function . . . 327
9.5 Faults and Warnings . . . 328
At a Glance . . . 328
Module faults which can be accessed by explicit reading . . . 329
Channel faults accessible via explicit reading . . . 330
List of Configuration and Adjustment Errors . . . 331
Faulty Registers . . . 334
List of Error Codes for a WRITE_CMD Command . . . 339
WRITE_CMD Command: Programming Errors . . . 340
WRITE_CMD Command: Communication errors . . . 343
WRITE_CMD Command: System errors . . . 344
WRITE_CMD Command: System warnings . . . 347

TRF_RECIPE Command Explicit Writing Faults 348
Operating modes 349
Logic diagrams of faults 350
Drive faults 352
9.6 Performances of the TSX CSY 84 354
Performances of the TSX CSY 84 354

Chapter 10 Interface language 359
At a Glance 359
10.1 PL7 Interface 361
Introduction 361
Interface Language 362
Parameter Management 364
WRITE_PARAM and READ_PARAM 366
SAVE_PARAM and RESTORE_PARAM 367
WRITE_CMD 368
WRITE_CMD Examples 370
READ_STS 372
TRF_RECIPE and MOD_PARAM 373
10.2 List of Variables 374
At a Glance 374
Implicitly Exchanged Input Variables 375
Implicitly Exchanged Output Variables 378
Adjustment Parameters for the SERCOS® Function 381
Adjustment Parameters for an Individual Axis 382
Adjustment Parameters for a Set of Follower Axes 385
Constant Words 389

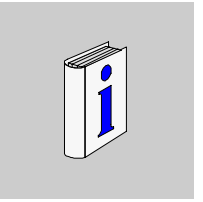
Chapter 11 Appendices 391
At a glance 391
11.1 List of Accessible Functions 392
At a Glance 392
Accessible Functions for SERCOS® Channel 0 393
Accessible Functions for a Real Axis 395
Accessible Functions for an Imaginary Axis 399
Accessible Functions for a Remote Axis 403
Accessible Functions for a Group of Co-ordinated Axes 405
Accessible Functions for a Group of Slave Axes 407
Accessible Functions for a Cam Profile 409

Part III Functions specific to the TSX CSY 164 411
At a Glance 411

Chapter 12 Compatibility 413
Compatibility 413

Chapter 13 Configuration of the TSX CSY 164 module 415
At a Glance 415
Configuration of the TSX CSY 164 module 416
Configuring channels 417
Chapter 14 Functions specific to the TSX CSY 164 419
At a Glance 419
Dynamic Reconfiguration of Groups 420
Monitoring Function of the Position Deviations Between Axes 422
Chapter 15 Performances of the TSX CSY 164 427
Performances of the TSX CSY 164 Module 427
Index 431

About the Book



At a Glance

Document Scope

This manual contains instructions on movement control software installation with TSX CSY 84 / 164 SERCOS® Module.

This manual is divided into three distinct parts:

- **Part I:** this part is **common to the two TSX CSY 84 and TSX CSY164 modules**, and deals with the SERCOS® range in Premium PLCs,
- **Part II:** this part is **common to the two TSX CSY 84 and TSX CSY164 modules**, and deals with movement commands in these modules,
- **Part III:** this part is **specific to the TSX CSY164 module**.

Note: All mounting procedures for the TSX CSY 84 are the same for the TSX CSY 164 module.

Validity Note

The update of this documentation includes the "Manual command" function in debug mode.

Related Documents

Title of Documentation	Reference Number
Hardware Implementation Manual	TSX DM 57 xx F

User Comments

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

The SERCOS® package in Premium PLCs

I

At a Glance

Subject of this Part

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

Note: In this part, all mounting procedures for the TSX CSY 84 are the same for the TSX CSY 164. Furthermore, the TSX CSY 84 is used as an example.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	SERCOS® Architecture	17
2	Setup methodology	27
3	Initiation Example: TSX CSY 84 module	33

SERCOS® Architecture

1

At a glance

Aim 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's 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
Limitations of PL7 Services According to Module Version	25

Introduction

Digital Link

The digital link between an axis command module (master) and intelligent drives (slaves) is defined by European standard EN 61491 concerning electrical equipment for industrial machines.

The use of a distributed architecture enables the input/output applications (position encoder, emergency stop etc.) to be connected directly to the drives, which reduces cabling costs.

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

Exchanged Data

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

- cyclic data exchanged from master to slaves (position command 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 and 2 reserved bytes in write. Several cycles are therefore required to exchange an object (eg. 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 drives are in repeater mode.
Phase 1	Determination of slaves on the bus.
Phase 2	Drive system configuration.
Phase 3	Cyclic exchange programming. Drive parametering.

In normal operation

Phase	Operating mode
Phase 4	Active cyclic exchanges.

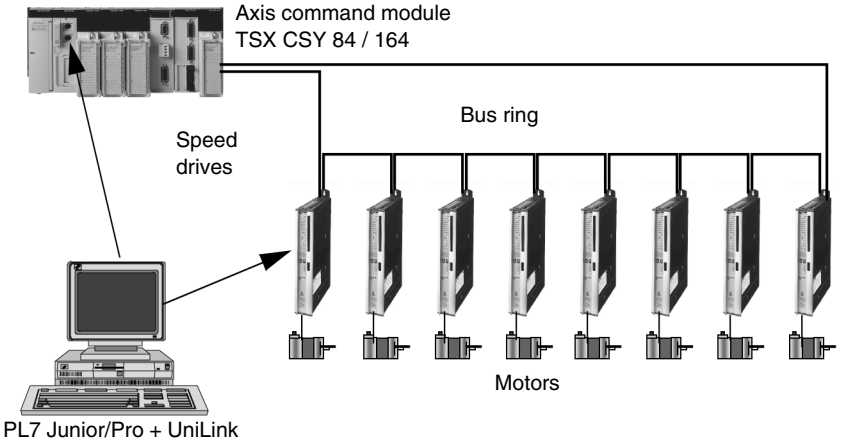
Each drive acting as a repeater on the bus, a power supply cut, a communication error, a fault on one of the drives or the disconnection of the power supply to the bus causes it to switch to phase 0.

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



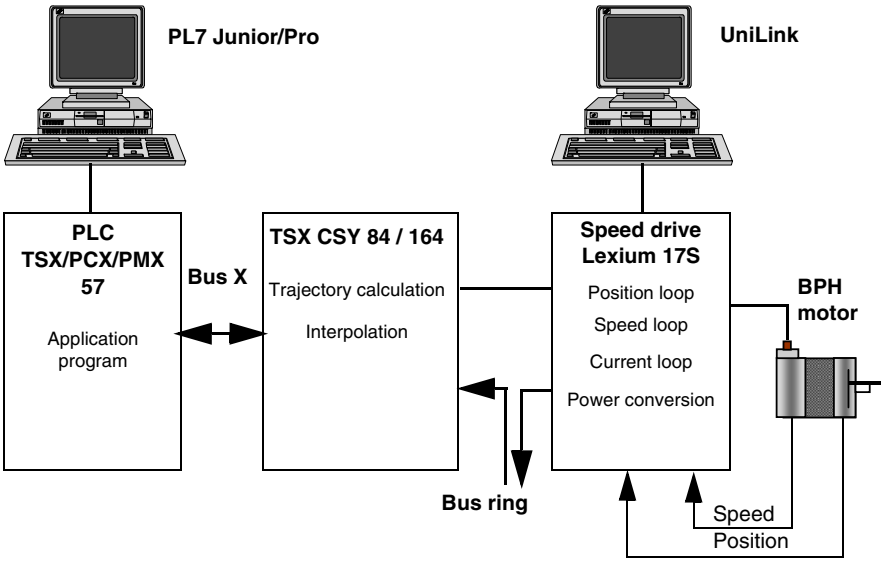
Introduction to the Premium package

The Premium package consists of:

- the range of Premium PLCs (rack, supply, processor etc.): TSX/PCX/PMX 57 software version ≥ V3.3 at a minimum,
- the axis command modules TSX CSY 84, which can control up to 8 drives via the bus ring,
- the range of Lexium 17S drives (4 A to 56 A): LXM 17S,
- the range of motors: BPH, etc.,
- UniLink software which can be used to enter parameters and adjust drives.
- PL7 Junior/Pro software minimum version SV 3.4 + C functional complement or SV > 3.4 for the TSX CSY 84,
- PL7 Junior/Pro software minimum version SV 4.3 + Motion TSX CSY 164 Add-on or SV ≥ 4.4 for the TSX CSY 164.

Functional Diagram

The following diagram shows the different functions performed by the axis command module and drives:



Note: It is possible to use a variable speed controller that is not in the Premium 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 command module TSX CSY 84 / 164 performs the trajectory calculation and interpolates several axes.

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

The exchanges between the PLC and the axis command module are made through the X bus at the bottom of the rack.
The exchanges between the axis command module and the drives use the SERCOS® bus ring.

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 164 V1.0**	**CSY 164 V1.1**	**Reference**
Possibility to close the position loop via an incremental encoder or SSI on Lexium 17S and 17HP drives **(1)** (Not documented)	X	X	X	X	X	
Possibility to switch the command position to torque **(1) and (2)** (Not documented)	X	X	X	X	X	
Possibility to switch the command position to speed command **(1) and (2)** (Not documented)	X	X	X	X	X	
Read or write the drive parameters	X	X	X	X	X	*TRF_RECIPE, p. 212*
Position loop with external encoder **(1)**	X	X	X	X	X	
Torque mode **(1)**	X	X	X	X	X	
Speed mode **(1)**	X	X	X	X	X	
Manuel mode **(4)**		X	X	X	X	*Manual mode, p. 151*
Behavior upon fault or independent axis disable (FREEWHEEL_STOP) **(3)**			X	X	X	*Adjustment Parameters for an Individual Axis, p. 382*
Propagation of stops in a slave group			X	X	X	*Adjustment Parameters for a Set of Follower Axes, p. 385*
Alignment function of the emergency stop inclines			X	X	X	*Alignment Function of the Emergency Stop Inclines, p. 260*
Selection of stop behavior following latching command			X	X	X	*Behavior Following Axis Disable, p. 237*
Stop of a member axis of a slave group in an absolute position following an Unlink			X		X	*"MoveQueue" Function, p. 176*
Stop of a member axis of a slave group in an absolute position			X		X	*"MoveImmed" Function, p. 173*

	CSY 84 V1.1	CSY 84 V1.2	CSY 84 V1.3	CSY 164 V1.0	CSY 164 V1.1	Reference
"Target sequence activation" function			X		X	*"Target sequence activation" function, p. 190*
Modulo counter service			X		X	*Modulo counter service, p. 183*
Switching ACC and DEC parameters in the MOVE			X		X	*Sequence stop (Unlink) of the slave group set off by a trigger, p. 194* and *"MoveImmed" Function, p. 173*
Sequence stop (Unlink) of the slave group set off by a trigger			X		X	*Sequence stop (Unlink) of the slave group set off by a trigger, p. 194*
Dynamic configuration of independent channels				X	X	*Configuring channels, p. 417*
Dynamic reconfiguration of groups				X	X	*Dynamic Reconfiguration of Groups, p. 420*
Monitoring function of the deviations between axes				X	X	*Monitoring Function of the Position Deviations Between Axes, p. 422*

(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 restriction for the Lexium HPs and Lexiums option AS.
(3) Requires a minimum version for the Lexium MHDx 5.51drives.
(4) Requires version PL7 SV4.3 at minimum to obtain the debug screens.

Limitations of PL7 Services According to Module Version

Illustration

	PL7 3.4 + C functional complement	PL7 4.0 + Motion Add-On	PL7 4.1 PL7 4.2	PL7 4.3	PL7 4.3 + Motion Add-On
TSX CSY 84 configuration	X	X	X	X	X
TSX CSY 84 debug screens:		X	X	X	X
TSX CSY 84 adjustment screens		X	X	X	X
TSX CSY 84 manual mode screens				X	X
TSX CSY 164 configuration					X

Setup methodology

2

At a glance

Aim of this chapter

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

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Introduction	28
Implementing an independent axis	30
Implementation of a co-ordinated or slave axis group	32

Introduction

TSX CSY 84 Module Functionalities

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

- 8 real axes connected to a drive (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 Functionalities

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

- 16 independent axes (channels 1 to 16),
  Channels 1 to 16 support real axis functions, imaginary axis functions, and external setpoints,
- 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 drive via the SERCOS® bus ring.

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 drive.

Group of coordinated 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 drive, an imaginary axis, or a remote axis.

The groups of slave or coordinated axes consist of a collection of independent 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: %Qxy.0.18 and %Qxy.0.26 to %Qxy.0.31 set to 1).

Implementation methodology for a real axis occurs in 3 phases:

- Phase 1: configuration of the drive using UniLink software,
- Phase 2: configuration of the TSX CSY 84 / 164 module, using the PL7 configuration editor (module declaration and parameter configuration for all axes used),
- Phase 3: program application writing, transferred from this program to the PLC and application debugging.

Note: Enabling the Unilink software drives inhibits module commands to the drive. It is therefore necessary to disable the drive before exiting Unilink software.

Implementation Methodology for an Imaginary Axis

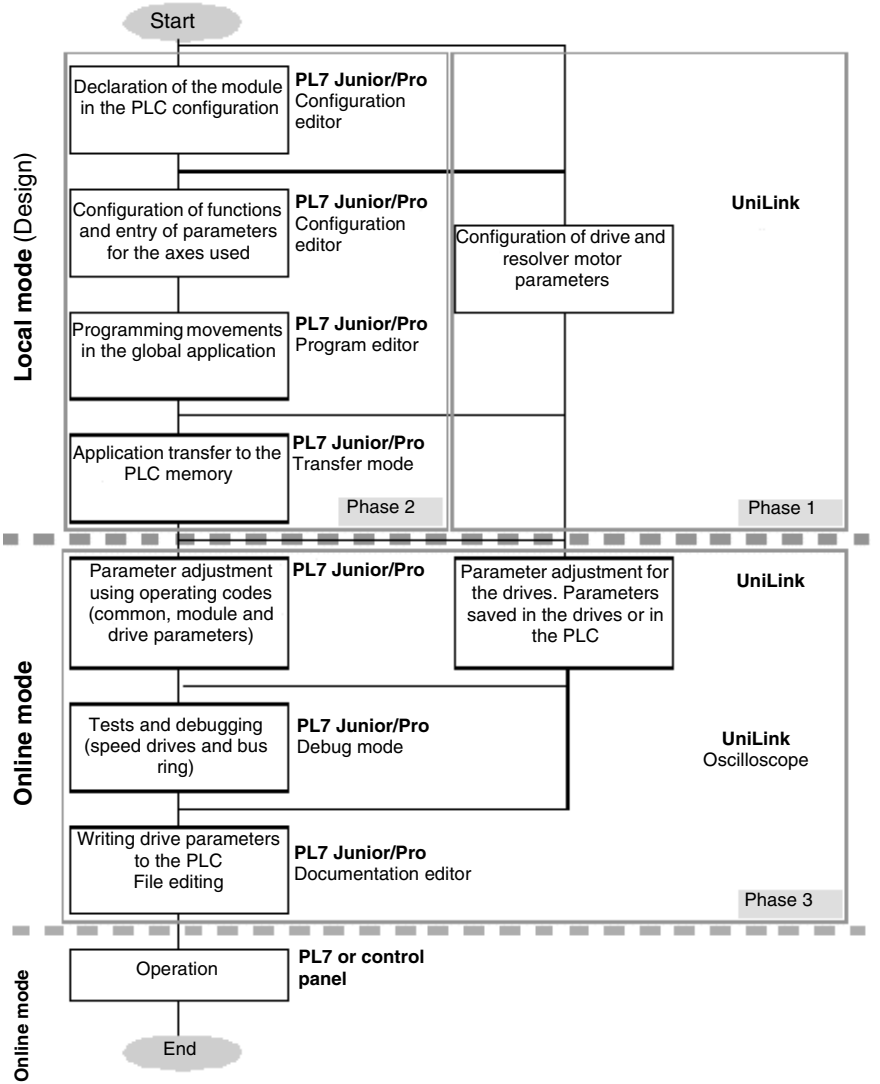
An imaginary axis is not connected to a drive (it is not a physical axis). Except for operations linked to a drive that does not exist, the implementation of an imaginary axis is the same as for a real axis.

Implementation Methodology for a Remote Input

A remote input has far fewer functions than other kinds of independent 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 drive 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.

Initiation Example: TSX CSY 84 module

3

At a Glance

Subject of this Chapter

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

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

What's 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
Configuration of the TSX CSY 84 module	40
Symbolization of I/O variables	43
Symbolization of Memory Variables	44
Programming the Preliminary Processing	45
Grafcet Programming	46
Transition Programming	47
Actions Programming	49
Programming the Post-Processing	53
Program transfer	57
Adjusting the Parameters	58
Introduction to the Debug Mode	61

Description of the example

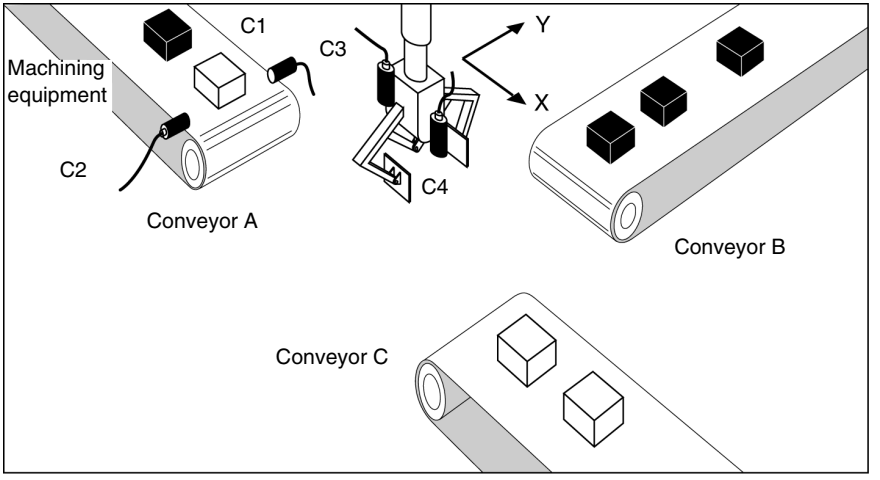
Introduction

This example is given for teaching purposes only. It will enable you to follow each phase of implementation for axis command TSX CSY 84 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 within the 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



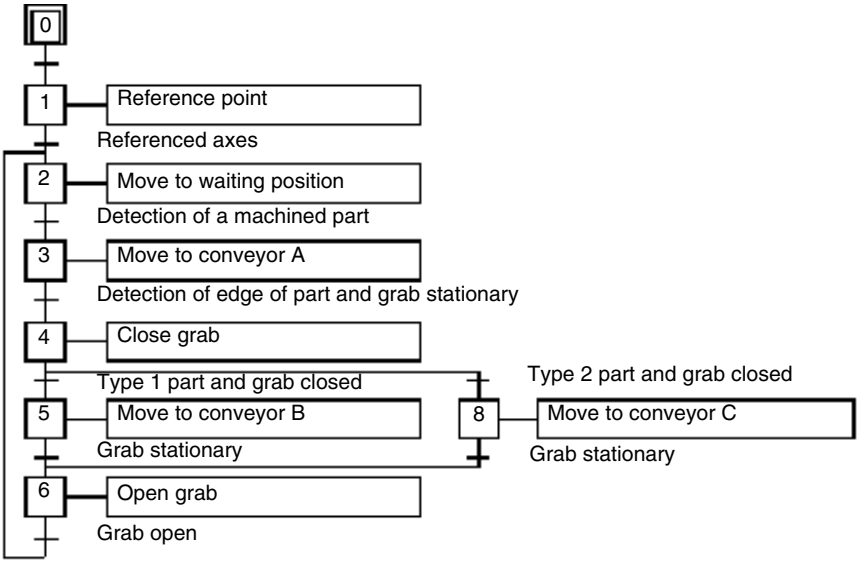
Inputs/Outputs

The inputs/outputs are as follows:

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

Application Grafcet

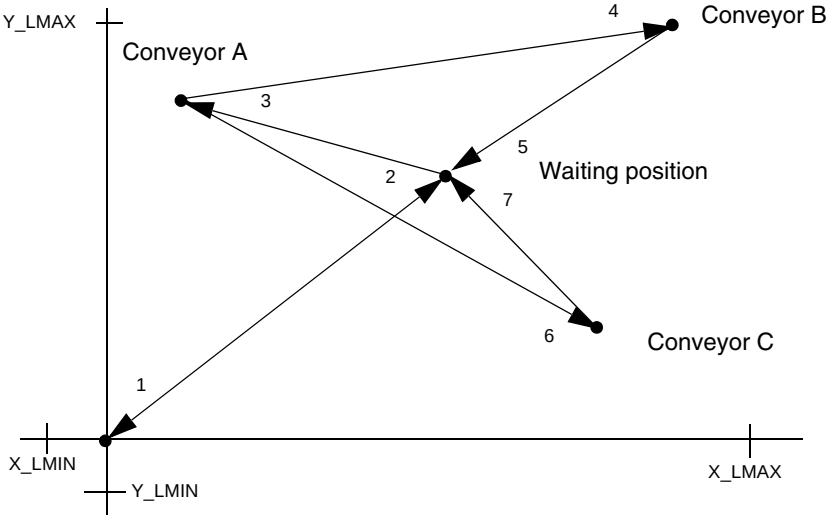
The application Grafcet is as follows:



Trajectory description

Trajectory illustration

The following diagram describes the trajectory:



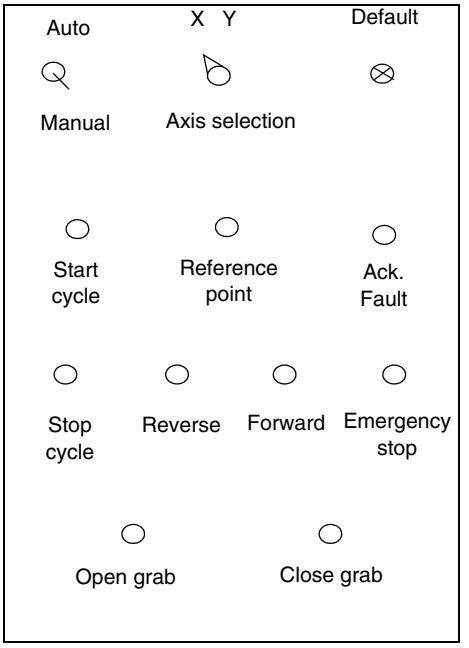
Description

1	Source port at speed Vp0
2	Switch at Vret speed to the ready position (Xatt, Yatt) with stop
3	Switch to the A conveyor belt (XA, YA) until the machined item is detected at speed VA
4	Switch to speed VB on belt B (XB, YB) with stop
6	Switch to speed VC on belt C (XC, YC) with stop
5, 7	Switch at Vret speed to the ready position (Xatt, Yatt) with stop

Description of Commands in Manual Mode

Front Panel of the Human-Machine Interface

The following commands appear on a front panel, and are used to control the moving part manually whilst there is an installation error. The commands and LEDs are run by internal bits.



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 source port on the selected axis.
- **Forward / Reverse**: positive or negative manual switch command for the selected axis.
- **Fault**: LED indicating all hardware and application 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.

- for TSX CSY 84: that PL7 software minimum version V4.3 is installed,
- Version TSX CSY 84: V1.2 minimum (integrating the "Manual mode" function),
- UniLink software is installed,
- hardware is installed.

Methodology

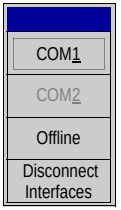
Installing the TSX CSY 84 modules requires:

- configuration of the Lexium 17S variable speed controller, using UniLink software,
- configuration of the TSX CSY 84 module, using PL7 software.

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.

Configuration of the TSX CSY 84 module

Software Declaration of the PLC Configuration

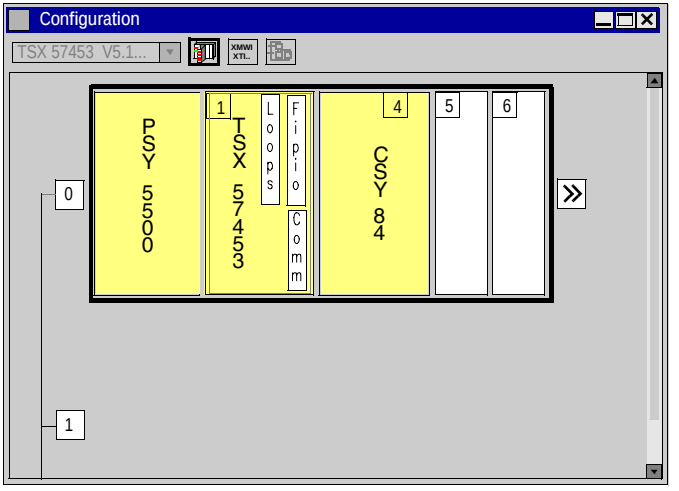
Launch the PL7 software, select the New File command, select a Premium processor and check the Grafcet option.
Access the configuration editor from the application browser. In order to do this:

- open the Station folder (double click on the icon or click on its link),
- open the Configuration folder (double click on the icon or click on its link),
- double click on the Hardware configuration icon.

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

- rack 0 and rack 1: TSX RKY 8E,
- processor: TSX P57 453,
- supply modules: TSX PSY 5500 for rack 0,
- axis command module: TSX CSY 84 (minimum V1.2) in position 4 of rack 0.

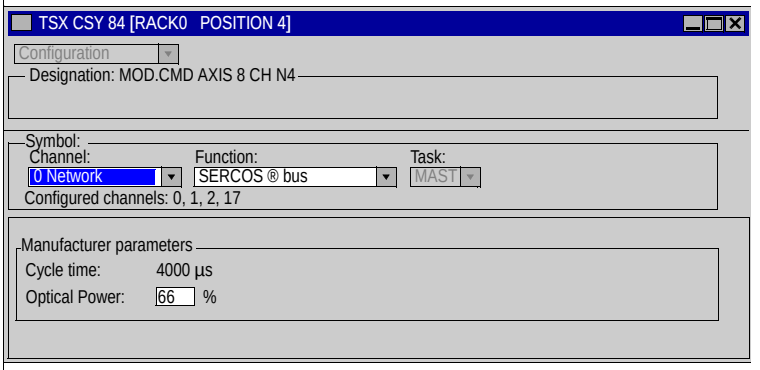
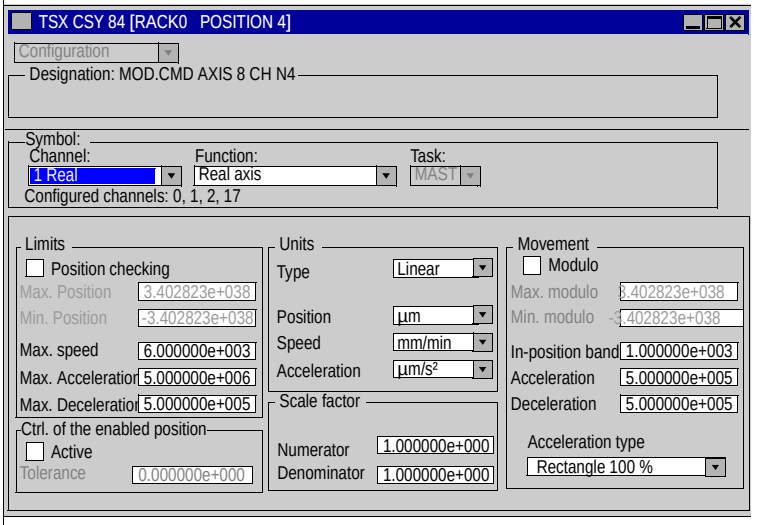
Illustration of the configuration screen:

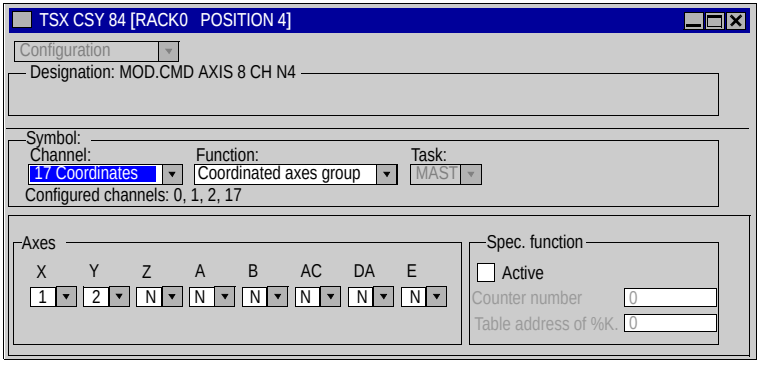


The processor cycle time should exceed 8ms (2 x the duration of the module cycle).

Axis configuration

The table below shows the procedure for configuring the axes:

Step	Action
1	Double click on the **TSX CSY 84 module** (position 4 in rack 0), in order to display the configuration screen.
2	In the channel level zone select: **0 Network**, then the **SERCOS ® bus** function.
3	Enter the parameter values: 
4	In the channel level zone select: **1 Real**, then the **Real axis** function.
5	Enter the parameter values: 
6	Repeat **step 3** to enter the parameters of the second axis (2 Real).
7	In the channel level zone select: **17 Coordinates**, then the **Group of co-ordinated axes** function.

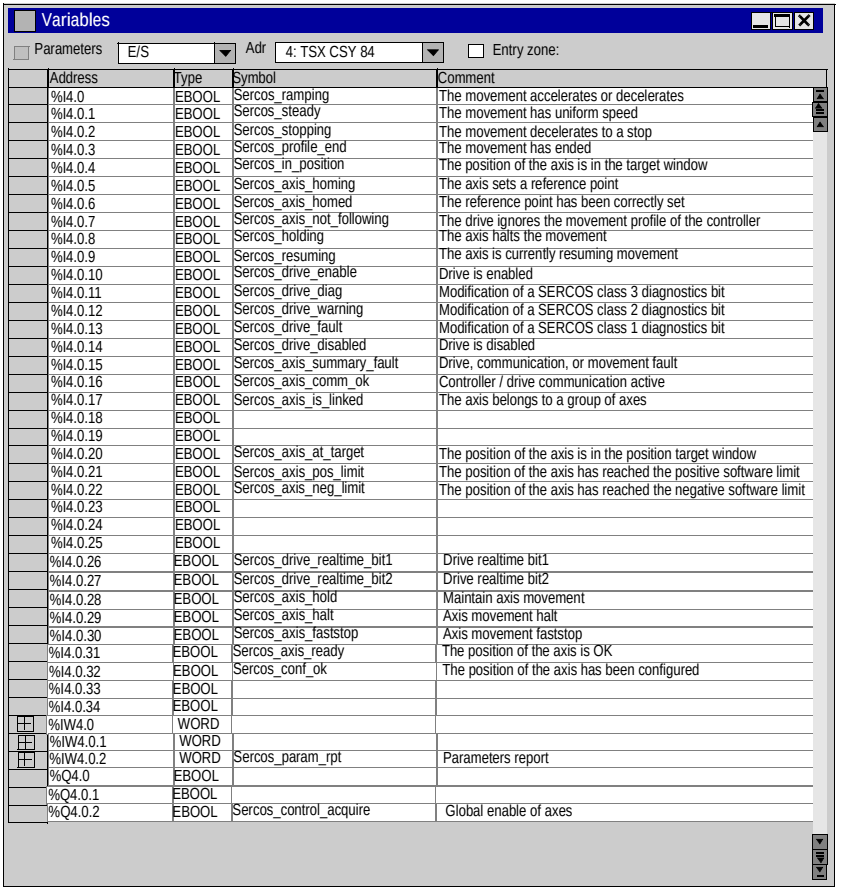
Step	Action
8	Enter the parameter values: 
9	Confirm the entry using the **Edit / Confirm** command or the icon
10	On the main screen of the configuration editor, confirm the configuration using the **Edit / Confirm** command or the icon: 

Note: Maximum speed. (VMax) is set for a 6000 revs/min motor, with movement of 1 mm per rev.

Symbolization of I/O variables

Entering Symbols

The symbol entry function can be accessed by double clicking in the application browser, then on the Variable and I/O icons



Address	Type	Symbol	Comment
%I4.0	EBOOL	Sercos_ramping	The movement accelerates or decelerates
%I4.0.1	EBOOL	Sercos_steady	The movement has uniform speed
%I4.0.2	EBOOL	Sercos_stopping	The movement decelerates to a stop
%I4.0.3	EBOOL	Sercos_profile_end	The movement has ended
%I4.0.4	EBOOL	Sercos_in_position	The position of the axis is in the target window
%I4.0.5	EBOOL	Sercos_axis_homing	The axis sets a reference point
%I4.0.6	EBOOL	Sercos_axis_homed	The reference point has been correctly set
%I4.0.7	EBOOL	Sercos_axis_not_following	The drive ignores the movement profile of the controller
%I4.0.8	EBOOL	Sercos_holding	The axis halts the movement
%I4.0.9	EBOOL	Sercos_resuming	The axis is currently resuming movement
%I4.0.10	EBOOL	Sercos_drive_enable	Drive is enabled
%I4.0.11	EBOOL	Sercos_drive_diag	Modification of a SERCOS class 3 diagnostics bit
%I4.0.12	EBOOL	Sercos_drive_warning	Modification of a SERCOS class 2 diagnostics bit
%I4.0.13	EBOOL	Sercos_drive_fault	Modification of a SERCOS class 1 diagnostics bit
%I4.0.14	EBOOL	Sercos_drive_disabled	Drive is disabled
%I4.0.15	EBOOL	Sercos_axis_summary_fault	Drive, communication, or movement fault
%I4.0.16	EBOOL	Sercos_axis_comm_ok	Controller / drive communication active
%I4.0.17	EBOOL	Sercos_axis_is_linked	The axis belongs to a group of axes
%I4.0.18	EBOOL		
%I4.0.19	EBOOL		
%I4.0.20	EBOOL	Sercos_axis_at_target	The position of the axis is in the position target window
%I4.0.21	EBOOL	Sercos_axis_pos_limit	The position of the axis has reached the positive software limit
%I4.0.22	EBOOL	Sercos_axis_neg_limit	The position of the axis has reached the negative software limit
%I4.0.23	EBOOL		
%I4.0.24	EBOOL		
%I4.0.25	EBOOL		
%I4.0.26	EBOOL	Sercos_drive_realtime_bit1	Drive realtime bit1
%I4.0.27	EBOOL	Sercos_drive_realtime_bit2	Drive realtime bit2
%I4.0.28	EBOOL	Sercos_axis_hold	Maintain axis movement
%I4.0.29	EBOOL	Sercos_axis_halt	Axis movement halt
%I4.0.30	EBOOL	Sercos_axis_faststop	Axis movement faststop
%I4.0.31	EBOOL	Sercos_axis_ready	The position of the axis is OK
%I4.0.32	EBOOL	Sercos_conf_ok	The position of the axis has been configured
%I4.0.33	EBOOL		
%I4.0.34	EBOOL		
%IW4.0	WORD		
%IW4.0.1	WORD		
%IW4.0.2	WORD	Sercos_param_rpt	Parameters report
%Q4.0	EBOOL		
%Q4.0.1	EBOOL		
%Q4.0.2	EBOOL	Sercos_control_acquire	Global enable of axes

Symbols for Internal Variables

The following internal variables are symbolized:

Address	Symbol	Comment
%MF102	Vit_atteindre	
%MF104	Vit_manu	

Symbolization of Memory Variables

Entering Symbols

The symbol entry function can be accessed by double clicking in the application browser, then on the Variable and Memory icons:

Address	Symbol	Comment
%M0	Presence_piece	
%M1	Capteur_2	Item identification sensor (0= type 2; 1 = type 1)
%M2	Capteur_3	Grab open / grab closed sensor
%M3	Auto_manu	Select mode switch (0 = auto; 1 = man)
%M4	Depart_cycle	
%M5	Arret_cycle	
%M6	Selection_x_y	Select axis manually to move or PO (1 = x; 0 = y)
%M7	Po_manu	
%M8	Avant	
%M9	Arriere	
%M10	Acq_defaut	
%M11	Type_def	
%M12	Valid_axes	Confirm axes
%M13	Ouv_pince	
%M14	Fermeture_pince	
%M20	Pince	Actuator command to open / close grab
%M21	Defaut	Fault indication

Programming the Preliminary Processing

MAST - Prl

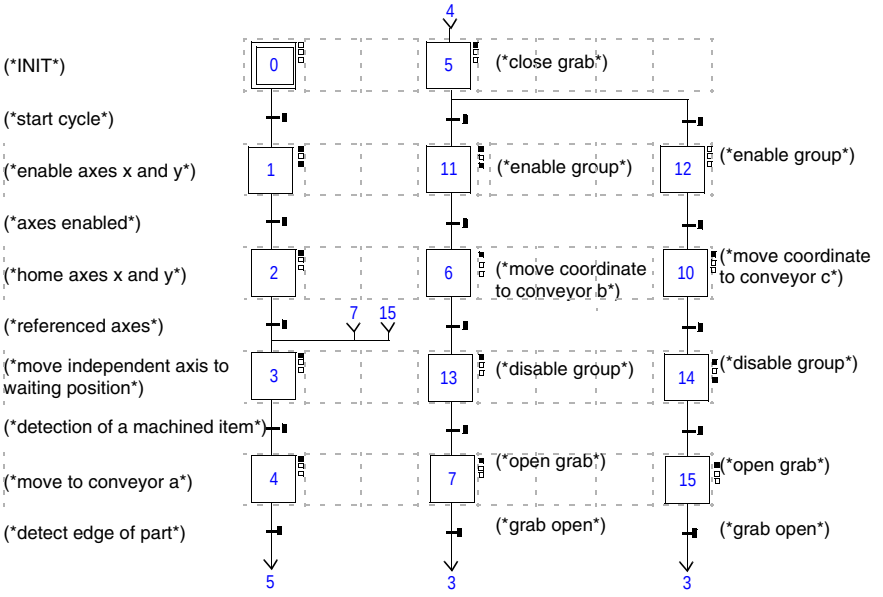
```
! (* chart initialization *)
IF NOT %M4 OR %S1 THEN SET %S21; END_IF;

!(* initialization of movement values *)
%MF50:=%MF52:= 1000.0; (* ready position *)
%MF54:=20000.0; (* move to belt b axis x *)
%MF56:=15000.0; (* move to belt b axis y *)
%MF58:=20000.0; (* move to belt c axis x *)
%MF60:=5000.0; (* move to belt c axis y *)
%MF102:=500.0; (* auto speed *)
%MF104 := 2000.0;(* manual speed *)
%MF106:= 1000.0;(* reference point speed *)

!(* initialization of channels 0_1_2 *)
%Q4.0.26 :=NOT %S0;
```

Grafcet Programming

CHART - PAGE 0 Grafcet



Transition Programming

Transition 0 -> 1

- **%X(0) -> %X(1)**

! (*dcy and auto mode and no error*)

```
%M4 AND NOT %M3 AND NOT %I4.1.15 AND NOT %I4.2.15
```

Transition 1 -> 2

- **%X(1) -> %X(2)**

! (*enabled and in_command*)

```
%I4.1.10 AND %I4.2.10 AND %I4.1.18 AND %I4.2.18
```

Transition 2 -> 3

- **%X(2) -> %X(3)**

! (*speed drives referenced*)

```
%I4.1.6 AND %I4.2.6
```

! (*check exchanges*)

```
AND(%M4.1.19=0)AND(%M4.2.19=0)AND NOT %MW4.1:X1 AND NOT
%MW4.2:X1
```

Transition 3 -> 4

- **%X(3) -> %X(4)**

! (*verification => end of instruction and in position*)

```
%I4.1.3 AND %I4.2.3 AND %I4.1.4 AND %I4.2.4
```

! (*presence of part*)

```
AND %M0
```

! (*check exchanges*)

```
AND(%M4.1.19=0)AND(%M4.2.19=0)AND NOT %MW4.1:X1 AND NOT
%MW4.2:X1
```

Transition 4 -> 5

- **%X(4) -> %X(5)**

! (*verification => end of instruction and in position*)

```
%I4.1.3 AND %I4.2.3 AND %I4.1.4 AND %I4.2.4
```

(*check exchanges*)

```
AND (%MW4.1.3=0) AND (%MW4.2.3=0) AND NOT %MW4.1:X1
AND NOT %MW4.2:X1
```

Transition 5 -> 11

- **%X(5) -> %X(11)**

! (*machined item identification*)

```
%M1 AND %M20
```

Transition 11 -> 6

- **%X(11) -> %X(6)**

! (*coordinated set enabled and in command?*)

```
%I4.17.10 AND %I4.17.18
```

Transition 6 -> 13

- **%X(6) -> %X(13)**
  ! (*verification => end of instruction and in position*)
  `%I4.1.3 AND %I4.2.3 AND %I4.1.4 AND %I4.2.4`
  (*check exchanges*)
  `AND (%MW4.17.19=0) AND NOT %MW4.17:X1`

Transition 13 -> 7

- **%X(13) -> %X(7)**
  ! (*verification => ch17 in command*)
  `NOT %I4.17.18`

Transition 7 -> 3

- **%X(7) -> %X(3)**
  ! (*grab open?*)
  `NOT %M2 AND %X7.T>100`

Transition 5 -> 12

- **%X(5) -> %X(12)**
  ! (*machined item identification*)
  `%M2 AND %M20`

Transition 12 -> 10

- **%X(12) -> %X(10)**
  ! (*coordinated set enabled and in command?*)
  `%I4.17.10 AND %I4.17.18`

Transition 10 -> 14

- **%X(10) -> %X(14)**
  ! (*verification => end of instruction and in position*)
  `%I4.1.3 AND %I4.2.3 AND %I4.1.4 AND %I4.2.4`
  (*check exchanges*)
  `AND (%MW4.17.19=0) AND NOT %MW4.17:X1`

Transition 14 -> 15

- **%X(14) -> %X(15)**
  ! (*verification => ch17 in command*)
  `NOT %I4.17.18`

Transition 15 -> 3

- **%X(15) -> %X(3)**
  ! (*grab open?*)
  `NOT %M2 AND %X15.T>100`

Actions Programming

Step 1

- **%X1 P1**

```
! (*enable channels and speed drives on axes 1 and 2*)
SET %Q4.1.10; SET %Q4.1.2; SET %Q4.2.10; SET %Q4.2.2;
```

Step 2

- **%X2 P1**

```
! (*channel 2 reference point*)
%MW4.2.26 := 6034; (***Reference point***)
%MD4.2.27 := 0; (***Param 1***)
%MD4.2.29 := 0; (***Param 2 ***)
%MF4.2.31 := 1.0; (***1 => positive direction, -1 negative direction***)
%MF4.2.33 := %MF106; (***speed***)
WRITE_CMD %CH4.2;

! (*channel 1 reference point*)
%MW4.1.26 := 6034; (***Reference point***)
%MD4.1.27 := 0; (***Param 1***)
%MD4.1.29 := 0; (***Param 2***)
%MF4.1.31 := 1.0; (***1 => positive direction, -1 => negative direction***)
%MF4.1.33 := %MF106; (***speed***)
WRITE_CMD %CH4.1;
```

Step 3

- **%X3 P1**

! (*channel 2 absolute movement*)

```
%MW4.2.26 := 513; (***Move Immed***)
(***Move type: -1 default, 0 ABS_MOVE, 1 INCR_MOVE, 2 CONT_MOVE, 3
ABS_MOVE_POS, 4 ABS_MOVE_NEG***)
%MD4.2.27 := 0;
%MD4.2.29 := 0; (***Param_2***)
%MF4.2.31 := %MF50; (Direction / Cont 1.0, Position / abs***)
%MF4.2.33 := 1000.0; (***Speed***)
WRITE_CMD %CH4.2;
```

! (*channel 1 absolute movement*)

```
%MW4.1.26 := 513; (***Move Immed***)
(***Move type: -1 default, 0 ABS_MOVE, 1 INCR_MOVE, 2 CONT_MOVE, 3
ABS_MOVE_POS, 4 ABS_MOVE_NEG***)
%MD4.1.27 := 0;
%MD4.1.29 := 0; (***Param_2***)
%MF4.1.31 := %MF52; (Direction / Cont 1.0, Position / abs***)
%MF4.1.33 := 1000.0; (***Speed***)
WRITE_CMD %CH4.1;
```

Step 4

- **%X4 P1**

! (*channel 1 absolute movement*)

```
%MW4.1.26 := 513; (***Move Immed***)
(***Move type: -1 default, 0 ABS_MOVE, 1 INCR_MOVE, 2 CONT_MOVE, 3
ABS_MOVE_POS, 4 ABS_MOVE_NEG***)
%MD4.1.27 := 0;
%MD4.1.29 := 0; (***Param_2***)
%MF4.1.31 := 0.0; (Direction / Cont 1.0, Position / abs***)
%MF.1.33 := %MF102; (***Speed***)
WRITE_CMD %CH.1;
```

! (*channel 2 absolute movement*)

```
%MW4.2.26 := 513; (***Move Immed***)
(***Move type: -1 default, 0 ABS_MOVE, 1 INCR_MOVE, 2 CONT_MOVE, 3
ABS_MOVE_POS, 4 ABS_MOVE_NEG***)
%MD4.2.27 := 0;
%MD4.2.29 := 0; (***Param_2***)
%MF4.2.31 := 1000.0; (Direction / Cont 1.0, Position / abs***)
%MF4.2.33 := %MF102; (***Speed***)
WRITE_CMD %CH4.2;
```

Step 5

- **%X5 P1**
  ! (*close grab*)
  `SET %M20;`

Step 6

- **%X6 P1**
  ! (*channel 17 absolute movement*)
  `%MW4.17.26 := 513;` (***Move Immed***)
  (*Move type: -1 default, 0 ABS_MOVE, 1 INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG*)
  `%MD4.17.27 := 0;`
  `%MD4.17.29 := 0;` (***Param_2***)
  `%MF4.17.31 := %MF54;` (*Position*)
  `%MF4.17.33 := %MF56;` (*Speed*)
  `%MF4.17.35 := %MF102;` (*Position*)
  `%MF4.17.37 := %MF102;` (*Speed*)
  `WRITE_CMD %CH4.17;`

Step 7

- **%X7 P1**
  (*open grab*)
  `RESET %M20;`

Step 10

- **%X10 P1**
  ! (*channel 17 absolute movement*)
  `%MW4.17.26 := 513;` (*Move Immed*)
  (*Move type: -1 default, 0 ABS_MOVE, 1 INCR_MOVE, 2 CONT_MOVE, 3 ABS_MOVE_POS, 4 ABS_MOVE_NEG*)
  `%MD4.17.27 := 0;`
  `%MD4.17.29 := 0;` (*Param_2*)
  `%MF4.17.31 := %MF58;` (*Position *)
  `%MF4.17.33 := %MF60;` (*Speed*)
  `%MF4.17.35 := %MF102;` (*Position *)
  `%MF4.17.37 := %MF102;` (*Speed*)
  `WRITE_CMD %CH4.17;`

Step 11

- **%X11 P1**
  (*coordinated set enabled and in command*)
  `SET %Q4.17.2;SET %Q4.17.10;`

Step 12

- **%X12 P1**
  (*coordinated set enabled and in command*)
  `SET %Q4.17.2;SET %Q4.17.10;`

Step 13	• **%X13 P1** **(*coordinated set not in command*)** `RESET %Q4.17.18;`
Step 14	• **%X14 P1** **(*coordinated set not in command*)** `RESET %Q4.17.18;`
Step15	• **%X15 P1** **(*open grab*)** `RESET %M20;`

Programming the Post-Processing

MAST - POST

```
! (*RESET error management *)
IF %M10 THEN SET %Q4.0.15;RESET %M10;
ELSE RESET %Q4.0.15;
END_IF;
%M21:=%I4.1.15 OR %I4.2.15;
IF %M21 THEN RESET %M4;END_IF;

! (* initialization of G7 AND management of MANU / AUTO mode available from
SV1.2 *)
IF RE %M3 AND NOT %MW4.1:X1 AND NOT %MW4.2:X2
THEN SET %S21;
%MW4.1.26:=2572;(*** Manual mode selection ***)
%MD4.1.27:=1;(*** Manual mode selection ***)
WRITE_CMD %CH4.1;

%MW4.2.26:=2572;(*** Manual mode selection ***)
%MD4.2.27:=1;(*** Manual mode selection ***)
WRITE_CMD %CH4.2;SET %M3;
END_IF;

IF FE %M3 AND NOT %MW4.1:X1 AND NOT %MW4.2:X2
THEN
%MW4.1.26:=2572;(*** Manual mode selection ***)
%MD4.1.27:=0;(*** AUTOMATIC mode selection ***)
WRITE_CMD %CH4.1;
%MW4.2.26:=2572;(*** AUTOMATIC mode selection ***)
%MD4.2.27:=0;(*** Manual mode selection ***)
WRITE_CMD %CH4.2;RESET %M3;
END_IF;

IF NOT %M3
THEN JUMP %L10;
END_IF;

! (* ----------------  Management of manual mode ----------------*)
(************ positive x channel manual mode  **************)
IF %M8 AND %M6 AND %I4.2.25 AND %I4.1.25
THEN SET %Q4.1.4;
ELSE RESET %Q4.1.4
;END_IF;
```

```
(************** negative x channel manual mode  *************)
IF %M9 AND %M6 AND %I4.2.25 AND %I4.1.25
THEN SET %Q4.1.5;
ELSE RESET %Q4.1.5;
END_IF;
(*************** positive y channel manual mode  ************)
IF %M8 AND NOT %M6 AND %I4.2.25 AND %I4.1.25
THEN SET %Q4.2.4;
ELSE RESET %Q4.2.4;
END_IF;
(*************** negative y channel manual mode  ************)
IF RE %M9 AND NOT %M6 AND %I4.2.25 AND %I4.1.25
THEN SET %Q4.2.5;
ELSE RESET %Q4.2.5;
END_IF;
(************* Manual homing of channel x  **************)
IF RE %M7 AND %M6 AND NOT %MW4.1:X1
THEN
%MW4.1.26:=6034;(*** Reference point ***)
%MD4.1.27:=0;(*** param 1 ***)
%MD4.1.29:=0;(*** Param 2 ***)
%MF4.1.31:=1.0;(*** Direction ***)
%MF4.1.33:=%MF106;(*** Speed ***)
WRITE_CMD %CH4.1;
END_IF;
(************** Manual homing of channel y  *************)
IF RE %M7 AND NOT %M6 AND NOT %MW4.2:X2
THEN
%MW4.2.26:=6034;(*** Reference point ***)
%MD4.2.27:=0;(*** Param 1 ***)
%MD4.2.29:=0;(*** Param 2 ***)
%MF4.2.31:=1.0;(*** Direction ***)
%MF4.2.33:=%MF106;(*** Speed ***)
WRITE_CMD %CH4.2;
END_IF;

! (* ---------- stop cycle in progress  ----------*)
%L10:IF %M5 THEN
RESET %Q4.1.29;
RESET %Q4.2.29;
ELSE
SET%Q4.1.29;
SET %Q4.2.29;
RESET %M5;
END_IF;
```

```
! (* ---------- error type read ----------*)
IF %M11 THEN
%MW4.1.26:=5510;
WRITE_CMD %CH4.1;
%MW4.2.26:=5510;
WRITE_CMD %CH4.2;
RESET %M11;
END_IF;

! (* ---------- Disable channels ----------*)
(* disable drive on CHANNEL 1 and 2 *)
IF %M100
THEN
RESET %Q4.1.26;
RESET %Q4.2.26;
RESET %M100;
ELSE SET %Q4.1.26;
SET %Q4.2.26;
END_IF;

!
%L100:
```

Program transfer

Procedure for transferring a program

After the application has been configured and the program has been entered, they must be transferred into the PLC processor memory, by following the following procedure :

Step	Action
1	Connect the terminal to the PLC using the **AP** → **Connect**command.
2	Activate the **AP** → **Transfer**command.
3	Select the **Console ->PLC** option.
4	Confirm transfer.

Adjusting the Parameters

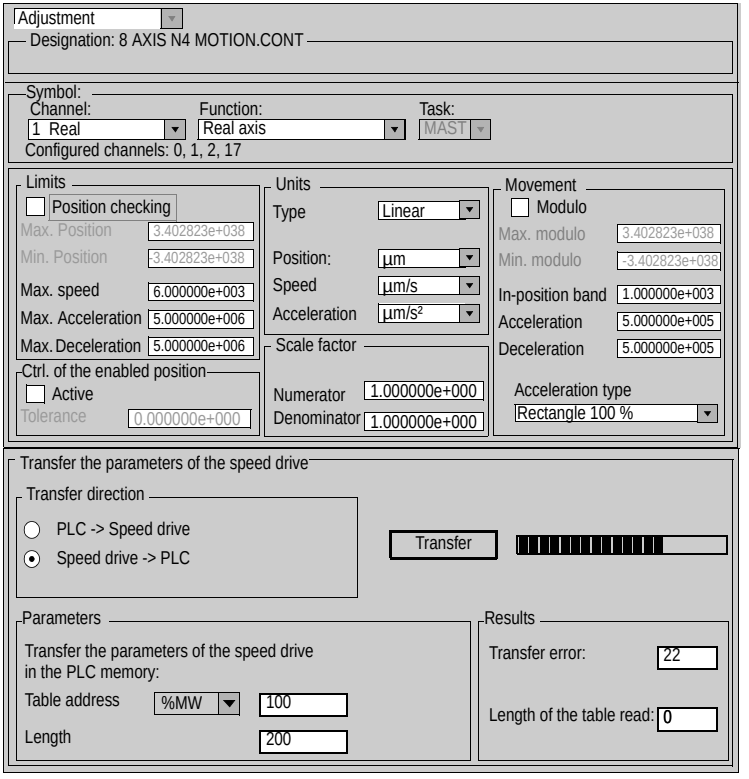
Introduction

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 local mode and modify the parameters directly inside the configuration screen,
- PL7 enables you to also modify the parameters in online mode by using either:
  - the adjustment screens (method explained below),
  - the functions associated with the adjustment parameters via the application.

Adjustment Procedure via the PL7 Screens

In order to carry out the parameter adjustment, you should perform the following steps:

Step	Action
1	Access the hardware configuration screen of the application.
2	Double click on the CSY 84 module. You can also select the module and execute the **Edit** → **Open Module** command.
3	Activate the **View** → **Adjust** command to access the adjustment screen (example: Channel 1) 
4	Enter the possible changes.
5	Confirm the entries using the **Edit** → **Confirm** command or click on the icon 

Restore/Save Parameters

This service is available for the Network channel, the real, imaginary and remote axes as well as for the slave axes. For more information on this service, refer to the Basic applications manual, Common applications section, RESTORE_PARAM and SAVE_PARAM functions.

Introduction to the Debug Mode

Introduction

Once the application has been performed then transferred, it is often necessary to debug it.
PL7 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 program.

Motion control common to TSX CSY 84 / 164 modules

II

At a Glance

Subject of this Part

This part introduces the TSX CSY 84 / 164 modules and describes the implementation of a movement command with these modules.

Note: In this part, all mounting procedures for the TSX CSY 84 are the same for the TSX CSY 164. Furthermore, the TSX CSY 84 is used as an example.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	Functional description of the TSX CSY 84 / 164 modules	65
5	Configuration of the TSX CSY 84 module	83
6	Programming the TSX CSY 84 / 164 modules	107
7	Adjustment of TSX CSY 84 / 164 modules	197
8	Debugging the TSX CSY 84 / 164 modules	289
9	Diagnostics and Maintenance	309
10	Interface language	359
11	Appendices	391

Functional description of the TSX CSY 84 / 164 modules

4

At a Glance

Subject of this Chapter

This chapter describes the functionalities of the TSX CSY 84 / 164 modules and the functions they can perform.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
TSX CSY 84 / 164 module functions	66
"SERCOS®" function	68
"Real Axis" Function	69
"Imaginary Axis" Function	70
"Remote Axis" Function	71
Concept of the Group	73
"Coordinated Axis Group" Function	75
"Slave Axis Group" Function	78
"Cam Profile" Function	81

TSX CSY 84 / 164 module functions

Introduction to the TSX CSY 84 Module

The TSX CSY 84 module has 5 types of application specific functions:
- SERCOS® function, installed by channel 0 (SERCOS® ring),
- Independent axis function installed by channels 1 – 16,
  - Real axis installed by channels 1 – 8,
  - Imaginary axis installed by channels 9 – 12,
  - Remote axis installed by channels 13 – 16,
- Coordinated 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.

Introduction to the TSX CSY 164 Module

The TSX CSY 164 module has 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,
- Coordinated 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.

Command Exchange

For each of the application-specific functions (SERCOS® ring, individual axis, coordinated 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 CONTROL bits, active on a rising edge (%Qxy.i.0 to %Qxy.i.15),
- 16 ALLOW bits, active on a falling edge (%Qxy.i.16 to %Qxy.i.31).

Optimization of Performance

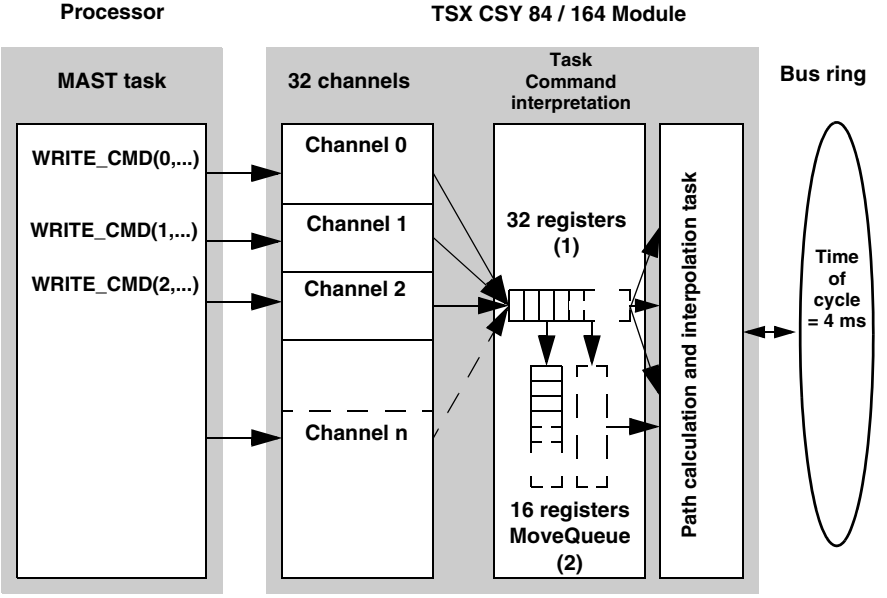
Implicit exchanges (%I, %Q) are performed for all channels, on each cycle of the MAST task. In order to optimize performance, it is recommended when possible **to favor implicit exchanges (%Q) over explicit exchanges (WRITE _CMD).**

Important

The TSX CSY 84 and TSX CSY 164 modules must be configured in MAST task.

Graphical Representation of Exchanges

On the TSX CSY 84, WRITE_CMD command exchanges are made according to the following principle:



(1) 32 registers (1 per channel)

(2) 16 registers of 32 commands (8 real, 4 imaginary, 4 coordinated axis groups)

"SERCOS®" function

At a Glance

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

Channel 0

Channel 0 is configured with default parameters. Parameters can be modified:

- either using configuration or adjustment screens,
- or through the application using explicit exchanges.

Accessible Services

Channel 0 enables the following services to be used (See *PL7 Interface, p. 361*):

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

"Real Axis" Function

At a Glance

The "imaginary axis" function is performed by channels 1 to 8 of the TSX CSY 84 module and by channels 1 to 16 of the TSX CSY 164 module.

This function can be used to control the drives via the bus ring to create physical axes (up to 8 possible physical axes).

Note: The parameters of an axis command module and variable speed controllers are run separately via PL7 and Unilink. So, if the user changes the value of a parameter used by the module and the variable speed controller, using the Unilink configurator, this will not be updated in the axis command module. After this type of operation, the user must therefore adjust the parameter in the module using PL7.

For more information on managing parameters, (see *Parameter Management, p. 364*).

Note: The drive corresponding to the real axis 1 should be located at address 1 of the SERCOS(r) ring.

Accessible Services

Channels 1 - 8 allow the following services to be used (See *PL7 Interface, p. 361*):

- 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 performed 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 coordinate 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 module TSX CSY 84) and 1 to 16 (of module TSX CSY 164) allow the following services to be used (See *PL7 Interface, p. 361*):

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

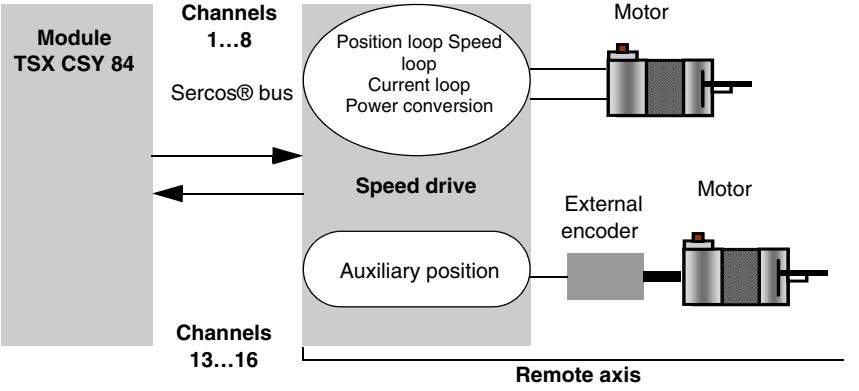
"Remote Axis" Function

At a Glance

The "remote axis" function is performed 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 is used to recognize position data from an external system.

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



Remote Value

The external value is accessible, according to its configuration, via:

- language object (implicit exchange) %QDxy.i.0 (REMOTE_POSITION),
- a drive's auxiliary input.

Note: For the **Remote value** channels, supported by a real axis, faults are indicated on the corresponding **real axis** channel.

Accessible Services

Channels 13 to 16 (of module TSX CSY 84) and channels 1 to 16 (of module TSX CSY 164) allow the following services to be used (See *PL7 Interface, p. 361*):

- 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.

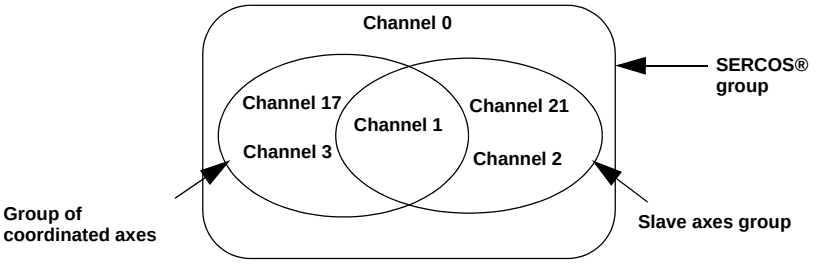
- the MovementControl interface for an individual axis always acts on that axis.
- the MovementControl interface for the SERCOS® ring acts on all of the MovementGroup axes on the string.
- the MovementControl interface for Coordinated Groups or Slave Groups are only in effect if the MovementGroup is 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 / 164 modules.

A "coordinated axis group" is a collection of physical axes (up to 8) whose movements are coordinated between them. 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 command. 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 their movement at the same time.

Note: All the axes used by a coordinated set must be referenced before it is confirmed and before completing coordinated moves

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



CAUTION

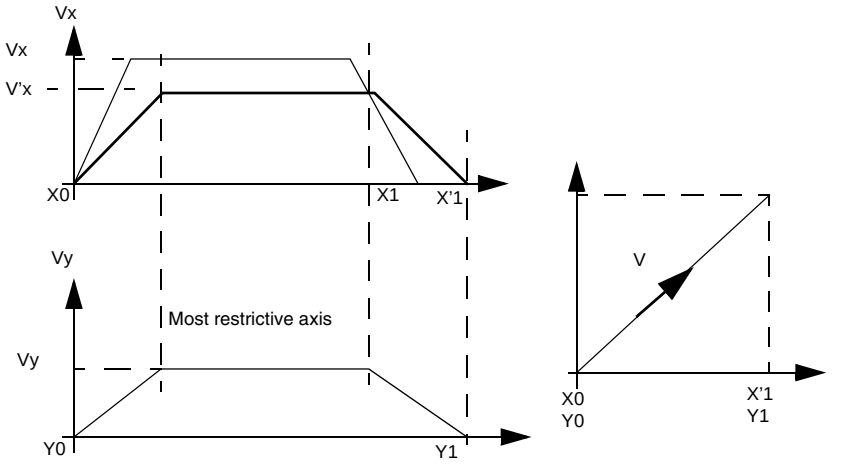
An axis member fault must be acknowledged at the level of the group (CLEAR_FAULT).

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

Reference Speed

Reference speed for the movement of a coordinated axis group must be calculated in relation to the most restricted axis. The algorithm reduces the speed of the highest performing axes.

For example, in the case of a 2 axis system

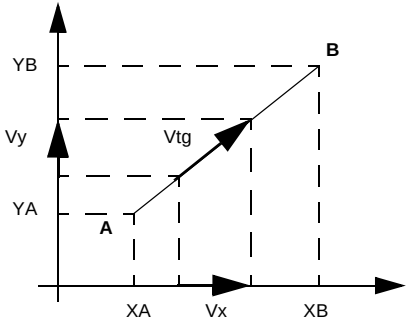


$$V = (V'x^{2} + Vy2)^{1/2}$$

Tangential Speed

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

Example in the case of a 2 axis system



$Vtg = (Vx^2 + Vy^2)^{1/2}$

The moving part must move 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.

From the value Vx (the most restricted axis) entered in the WRITE_CMD (MOVE) instruction, the interpolator calculates the Vy and then the Vtg speed.

Accessible Services

Channels 17 - 20 enable the following services to be used (See *PL7 Interface, p. 361*):

- WRITE_CMD,
- READ_STS,
- MOD_PARAM (see *Dynamic Reconfiguration of Groups, p. 420*), only on the TSX CSY 164 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 modules.

A group of slave axes consists of slave axes (a maximum of 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 ratio that has been defined in configuration, known as the Slave ratio (the slave position is equal to that of the master: setpoint or current value, 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 according 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 to a given axis.



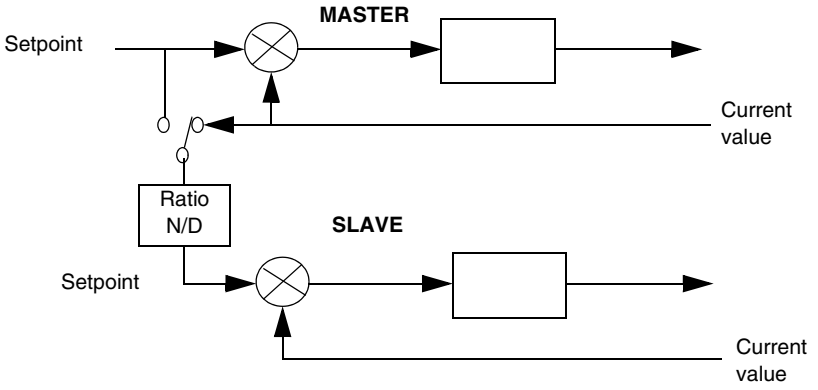
CAUTION

In a slave group, a master slave axis does not apply the controls defined in the maximum speed and acceleration parameters.
It follows the master speed law without constraint.
An axis member fault must be acknowledged at the level of the group (CLEAR_FAULT).

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

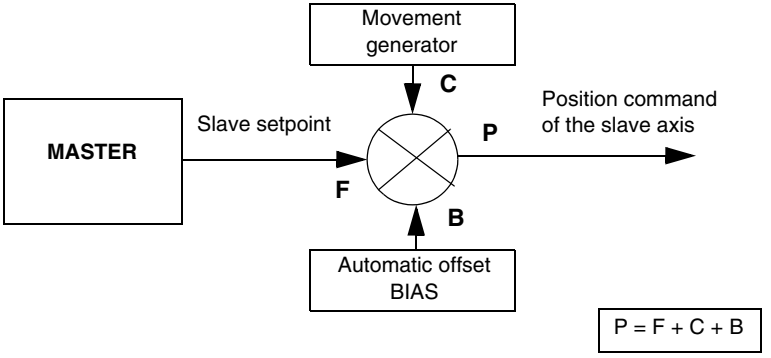
Ratio Mode

The ratio is defined by entering a numerator and a denominator. The ratio mode (gears) allows switches (current value, setpoint), offsets (Bias), start and slave stop conditions to be managed



Stop and Bias Sequence

These functions allow a movement or an automatic offset to be added to the slave setpoint. This allows the anticipation of a movement with the activation of the sequence or the continuation of a movement after the deactivation of the sequence.



Accessible Services

Channels 21 to 24 enable the following services to be used (See *PL7 Interface, p. 361*:

- READ_PARAM,
- WRITE_PARAM,
- WRITE_CMD (except Move),
- SAVE_PARAM,
- RESTORE_PARAM,
- READ_STS,
- MOD_PARAM (see *Dynamic Reconfiguration of Groups, p. 420*), only on the TSX CSY 164 module.

"Cam Profile" Function

At a Glance

The "cam profile" function is performed by channels 25 to 31 of the TSX CSY 84 / 164 modules.

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:

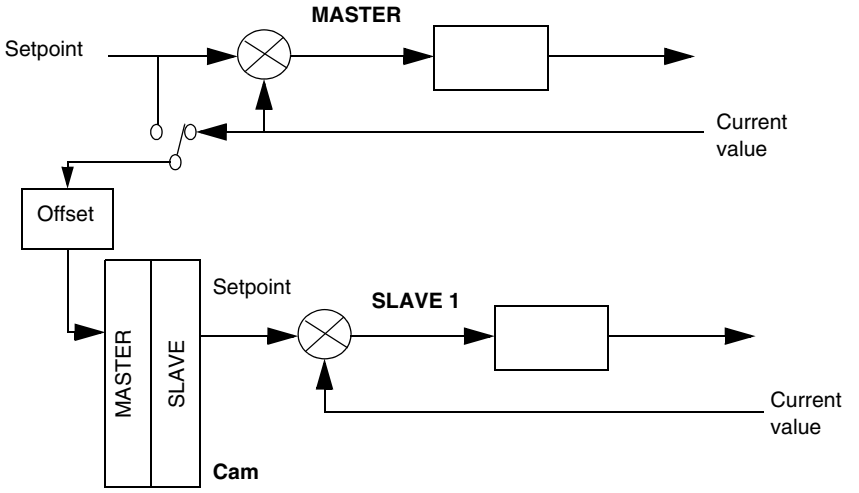
- a column that defines the position of the master in a slave axis group (usually an imaginary axis),
- a column that defines the position of the slave, associated with the master's position.

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:

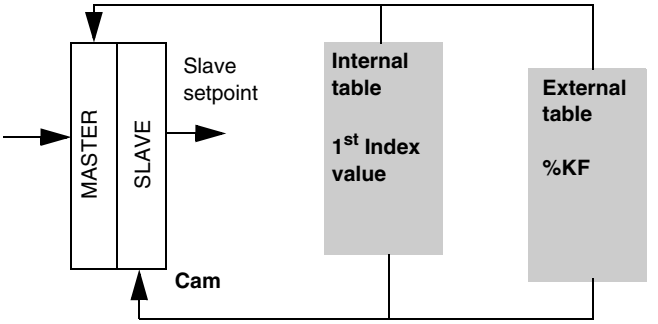


Cam

The cam can be implemented:

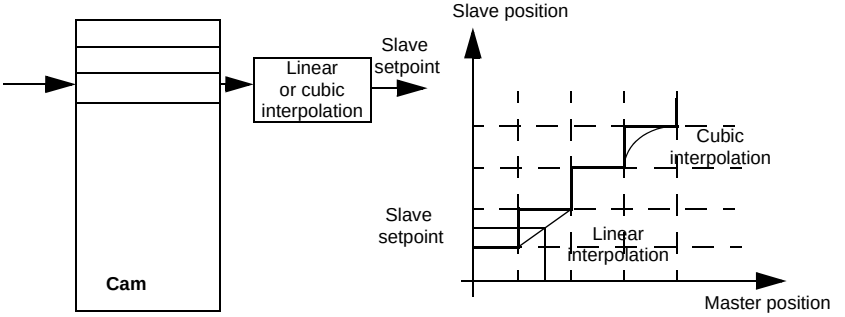
- either by an internal table in the axis command module, as defined by the initial table value and a fixed increment.
- or by an external table to the axis command module (contained in the PLC), defined as an address %KF.

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 *PL7 Interface, p. 361*):

- WRITE_CMD,
- READ_STS,
- TRF_RECIPE.

Configuration of the TSX CSY 84 module

5

At a Glance

Subject of this Chapter

This chapter describes the configuration screens of module TSX CSY 84 and some of the different functions it can perform: "SERCOS® functions," "real axis", "imaginary axis", etc.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Presymbolization	84
Module Configuration	85
Configuration of the Channels	88
Configuration of the SERCOS® Function (channel 0)	89
Configuring an Independent Axis (Channels 1 to 12)	90
Configuration of a Remote Axis (Channels 13 to 16)	95
Configuration of a set of coordinated axes (channels 17 à 20)	97
Configuration of a set of follower axes (channels 21 to 24)	99
CamProfile configuration	103

Presymbolization

Reminder

The Variables editor is used to symbolize the language objects associated with the TSX CSY 84 / 164 modules automatically. The user defines a generic symbol of the %CHxy.i channel, and all of the symbols of the objects linked to this channel are then automatically generated.

Syntax of Symbols

The syntax of the symbols which are generated automatically is as follows:

User_prefix_manufacturer_Suffix

- **User_prefix:** generic symbol given by the user to channel %CHxy.i (a maximum of 12 characters).
- **Manufacturer_suffix:** part of the symbol corresponding to the object language associated with the channel and given by the system (a maximum of 20 characters).

Comment

As well as the symbol, the system automatically generates a manufacturer comment providing a succinct reminder of the role of the object.

Module Configuration

Introduction

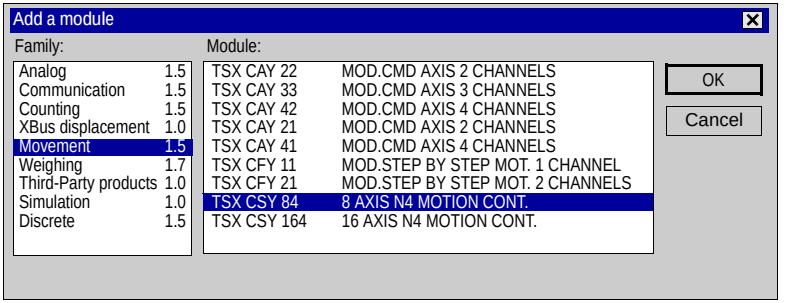
In Configuration mode, it is possible to define the operating characteristics for each TSX CSY 84 module path.

This mode is accessible in offline mode.

The TSX CSY 84 module occupies two positions in the rack. For this module to be offered, it is therefore necessary to select the furthermost right position of the two locations, reserved for receiving the module.

Adding the Module

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

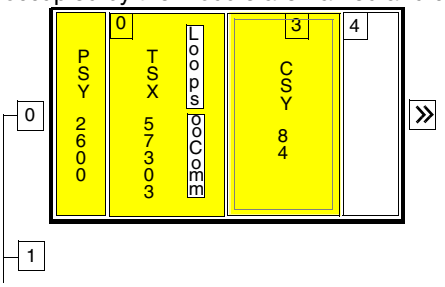


In order to do this:

- select **Movement** in the **Family** field,
- select the reference of the module (TSX CSY 84) in the **Module** field,
- confirm with **OK**.

Declaring the Module

To indicate that the module is declared in the configuration of the rack, the positions occupied by the module are framed and contain its reference.



Deleting a Module

To delete a module from its position:

- click on the latter to select it,
- press the <Del> key, a dialog box is then displayed,
- confirm the deletion of the module.

Application-Specific Channels

Application-specific channel encompasses all the channels of an intelligent module (counting module, axis command module, etc.). The 32 channels of module TSX CSY 84 are application-specific channels.

The maximum number of application-specific channels, which it is possible to implant in a configuration, is:

- TSX P57 103/153: 8 application-specific channels,
- TSX P57 203/2623/253/2823 / TPCX P57 203: 24 application-specific channels,
- TSX P57 303/3623/353 / TPCX 57 353: 32 application-specific channels,
- TSX P57 453/4823: 64 application-specific channels.

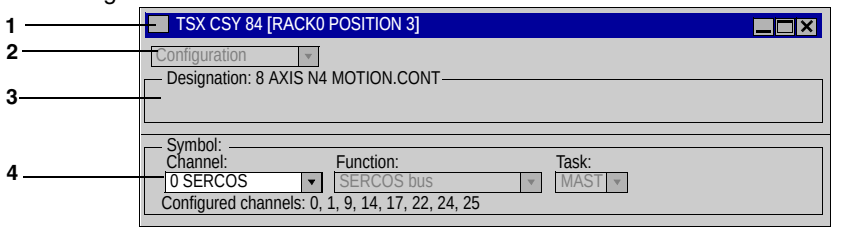
Accessing the Module Configuration Screen

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

- Select the module (by clicking on it),
- Activate the **Open Module** command from the **Services** pull-down menu.

Configuration Screen

The configuration screen of the module is as follows:



Description of the configuration screen

Specific	Description
1	This panel recalls the catalog reference of the module and its geographical address in the PLC (rack number and position in the rack).
2	This command zone indicates the current mode. Only Configuration mode can be accessed.
3	This module zone level contains the short title of the module.
4	This "channel" zone level is used to select the channel to be configured and recalls the associated function, as well as the task in which the implicit exchange objects are exchanged (always MAST). This zone recalls the numbers of the configured channels.

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.

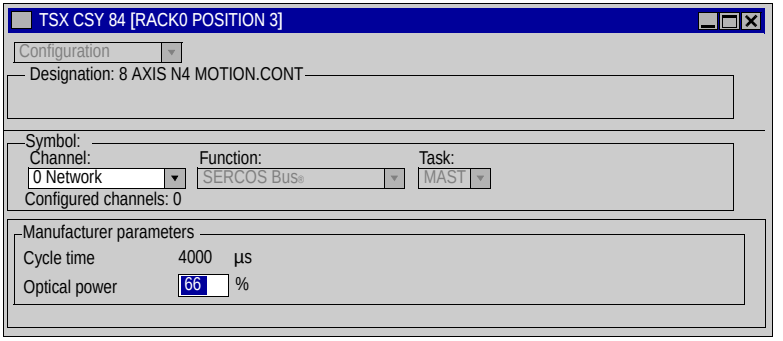
Configuration of the SERCOS® Function (channel 0)

Introduction

Channel 0, which controls the SERCOS® bus, is configured by default. The configuration screen is used to input the optical power and recalls the value of the bus parameters.

Configuration Screen

The configuration screen of the SERCOS® function of the TSX CSY 84 / 164 modules is as follows:



Description of the parameters

The SERCOS® bus parameters are defined in the **Manufacturing parameters** zone:

Parameters	Description
Cycle time	SERCOS® bus cycle time: 2 or 4 ms. 4 ms by default. The new value will be taken into account after a transition into SERCOS® phase 0 (opening of SERCOS® ring or SetCommandedPhase function) then another transition into phase 4.
Optical power	Adjustment of the optical power, which is necessary during first segment (included between the TSX CSY 84 or TSX CSY164 module and the first variable speed controller). This power (as a percentage) should be a function of the length of the segment (66% by default).

Configuring an Independent Axis (Channels 1 to 12)

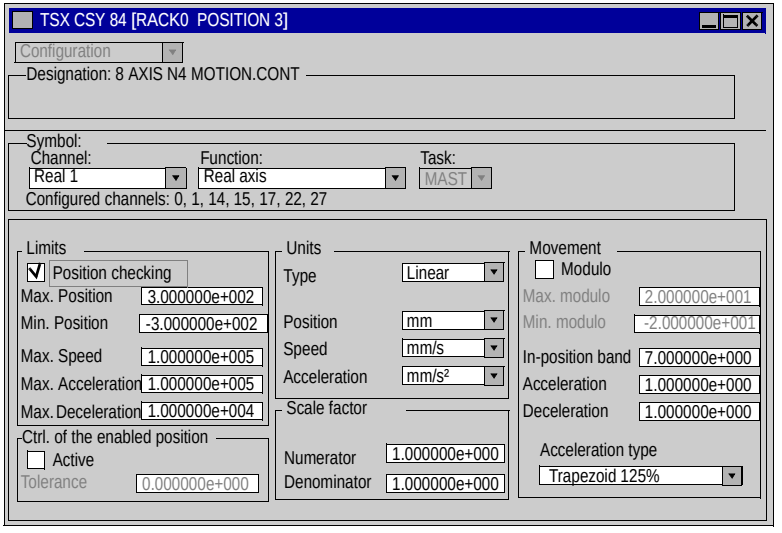
Introduction

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 variable speed controller). 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 variable speed controller configuration.

An imaginary axis can, for example, act as a master axis in a slave axis group. All of the parameters for this type of axis are defined in the configuration screens for the axis control modules.

Configuration Screen

The configuration screen for an individual axis is as follows. There are 5 parameter entry zones: **Limits, Enable position checking, Units, Scale factor, and Movement**.



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 permitted speed. This value is independent of the value defined in the variable speed controller (real axis). This value is entered as a floating point. When the maximum speed is configured as 0, speed monitoring is no longer enabled.
Max. Acceleration	Maximum permitted acceleration. This value is independent of the value defined in the variable speed controller (real axis). This value is entered as a floating point.
Max. Deceleration	Maximum permitted deceleration. This value is independent of the value defined in the variable speed controller (real axis). This value is entered as a floating point.

Enable position checking zone parameters

Description

Parameters	Description
Active	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.

Parameters of the Units Zone

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, mil ● Encoder points: points.
Speed	Speed unit. ● Angular: mrad/s, rad/s, rad/mim, deg/s, deg/min, arcmin/s, revs/s, revs/min ● Linear: μm/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: μm/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, mile/s^2 ● Encoder points: points/ms^2, points/s^2

Parameters of the Scale Factor Zone

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 rollover 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 drives which have the position defined in angular units: degrees, the module performs a rescaling operation, according to its reference which is measured in revolutions and its speed in revolutions/second.

For example, if the axis is configured as angular with a scale factor of 1/1:
- position units in revolutions and speed units in rev/s: an incremental movement of position 1 and speed 1 will execute 1 revolution in 1 second,
- position units in degrees and speed units 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 drive that remains in revolutions, the reference unit of the module is mm (for linear type) and inches (for linear English type). The transformation achieved by the module is equal to:
- 1 revolution for the drive = 1 mm for the module (linear type),
- 1 revolution for the drive = 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 units in mm and speed units in mm/s: an incremental movement of position 1 and speed 1 will execute 1 revolution in 1 second,
- position units in mm and speed units 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 an application 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 units 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 1m (or 10 axis revolutions) in 1 second (speed 1000 mm/s).

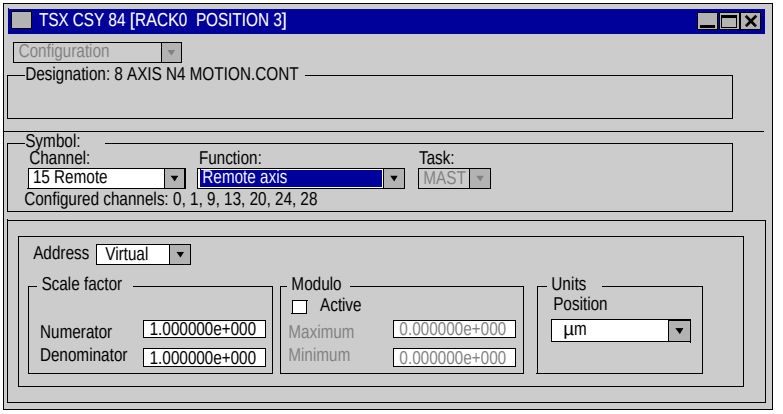
Configuration of a Remote Axis (Channels 13 to 16)

Introduction

A remote axis channel can be used to upload an item of external position data to the module. Configuring an external returns to configuring a real axis or an imaginary axis in which only the position data is valid.

Configuration Screen

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



Address

This field is used to define the origin of position data:

- Virtual: the position is written by the application program in the output word %QDxy.i.0,
- 1 to 8: the position is read on the output of the auxiliary position of the real axis defined by the choice of address field (1 to 8).

Parameters of the Scale Factor Zone

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.

Parameters of the Modulo Zone

Description

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

Parameters of the Units Zone

Description

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

Internal unit

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

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

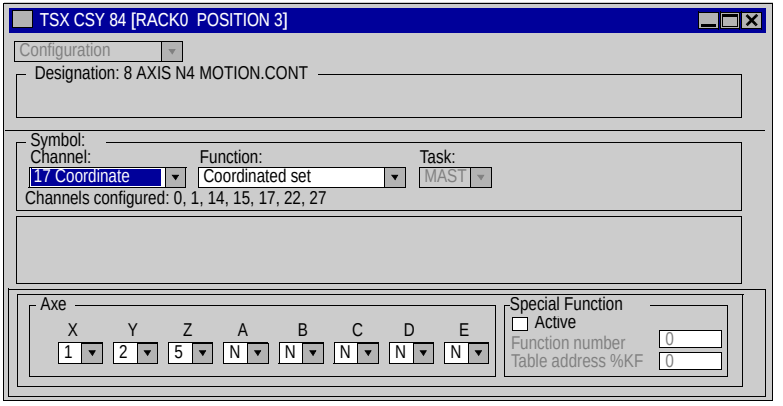
Configuration of a set of coordinated axes (channels 17 à 20)

Introduction

A coordinated axis group is made up of real axes or imaginary, whose movements are coordinated between themselves. Therefore, the configuration of a coordinated axis group requires the configuration of independent axes which belong to the group beforehand.

Configuration screen

The configuration screen of a coordinated axis group is as follows. It is used to define the list of axes which will be coordinated between themselves (a maximum of 8 axes).



Axis X to Axis E

These 8 fields are used to choose the axes which are coordinated between themselves (axes 1 to 12). The value N indicates that the axis does not belong to the coordination. The principal axis of the coordination is the one which is defined in the field Axis X.

Take care: the axes should be configured in order. For example, it is forbidden to have Axis X to N and Axis Y to 1.

The acceleration profile of the coordinated axis group is rectangular by default. Although each individual axis may have a defined profile, it is nevertheless possible to change the profile of the group using the SetAccelType command.

Parameters of the special Function zone

Description

Parameter	Description
Enable	This check box is used to confirm the activation of special functions
Function number	This field is used to enter the number of the special function to be activated.
Table address of %KF	This field is used to input the address from the beginning of the table containing the parameters associated with the special function.

Configuration of a set of follower axes (channels 21 to 24)

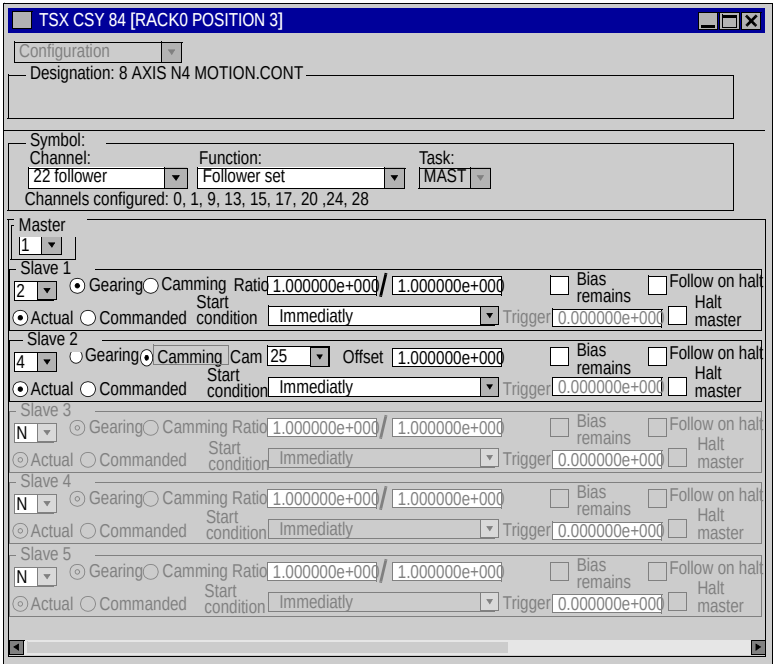
Introduction

A slave axis group is a collection of axes, composed of slave axes (a minimum of 6) which follow 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 is as follows. It offers 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 (axes 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.

Parameters of a Slave zone

Description

Parameter	Description
Slave 1 (to 6)	Is 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 the Ratio mode; that is to say following a ration 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 chosen in field No. 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 ration between the master axis and the slave. These values are entered as a floating point.
Start-up	Used to select the start-up condition: ● Immediately ● when the master position, increased by the offset value, reaches the value of the defined threshold in the Trigger field, in a positive direction, ● when the master position plus the offset value, reaches the value of the defined threshold in the Trigger field, in a positive direction, in a negative direction, ● when the master position plus the offset value is more than or equal to the value of the defined threshold in the Trigger field, ● when the master position plus the offset value is less than or equal to the value of the defined threshold in the Trigger field,
No.	In Cam mode, this field is used to select the cam profile number (between 25 and 31 inclusive).

Parameter	Description
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 master's table of the cam profile. 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 begin the pursuit of 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.
Bias remains (residual)	When this box is checked: ● a dynamics 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-up 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 <, >, <) to the value of the threshold (Trigger).
Follow on halt	When this box is checked, the confirmation of the link between the master and the slave provokes the halt of a possible additional movement of the slave axis, following a deceleration profile determined automatically.
Stop master/déf.	When this box is checked, the master stops when there is a pursuit discrepancy between the master and the slave.

Position of the 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,…) are ignored. To guarantee the security of the application, configure these parameters (of security) in the variable speed controller.

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, …). For example, if the master axe is configured in m and the slave axe in cm (different units but the same Linear type): and if one uses a scale factor of 1/1, this shows that if the master cover 1 mm, the slave will also move 1 mm.

If the units of master and slave 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) and if one wishes the slave to carry out 1 revolution when the master moves 1 m, it will be necessary to define the scale factor in the following way:

- 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 does 1000 mm, the slave makes 1 revolution).

CamProfile configuration

Introduction

A CamProfile can be used to define the position of a slave axis according to the position of the master axis using a table of points.

TSX CSY 84 / 164 modules

On the TSX CSY 84 / 164 modules, CamProfiles are configured on **channels 25 to 31**.

Configuration Example

The configuration screen for a CamProfile on the TSX CSY 84 module has 3 zones which can be used to define the table of points for the master and slave.

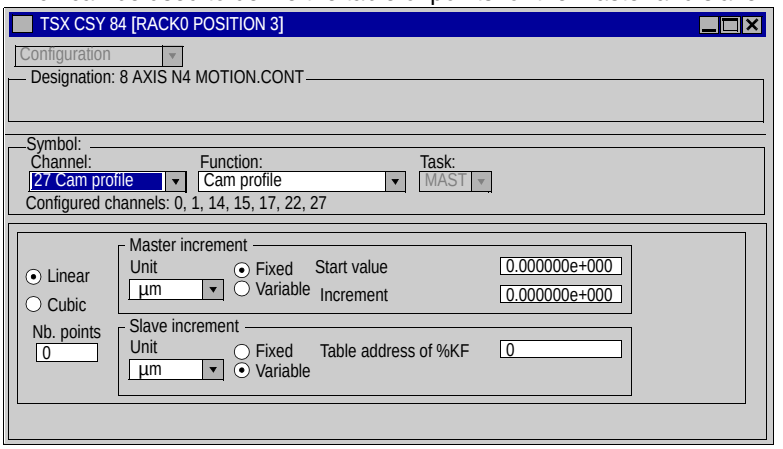


Table Parameters:

Parameters	Description
Linear	When this button is selected, interpolation between 2 consecutive CamProfile points is linear. This button operates alternately with the Cubic button.
Cubic	When this button is selected, interpolation between 2 consecutive CamProfile 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 CamProfile.

Master Increment zone parameters:

Parameters	Description
Unit	Used to define the unit which is 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 CamProfile 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 CamProfile. 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 CamProfile 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.
Table address 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 CamProfile 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 completely describes the rollover and an additional point is added (rollover + 1) for which the slave value is the same as the first value in the table.
For example, for a rollover 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 progressively tending towards the first value in the table).

Slave Increment zone parameters:

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 CamProfile points is always the same. This button operates alternately with the Variable button.

Parameters	Description
Start value	For a fixed increment, this field can be used to enter the start value for the CamProfile. 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 CamProfile 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.
Table address of %KF	For a variable increment, this field can be used to enter the start address for the table of points for the slave.

Programming the TSX CSY 84 / 164 modules

6

At a Glance

Subject of this Chapter

This chapter describes the different functions associated with axis movement.

Note: A TSX CSY 84 program written in PL7 will be compatible with the TSX CSY 164 module. However, porting an application written for a TSX CSY 84 to an application for the TSX CSY 164 is carried out by a "reconfiguration" operation in the PL7 application.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Motion Status Bits	109
6.2	Motion control functions	143
6.3	Move Functions	160
6.4	Motion Function Following Deactivation of the Sequence	171
6.5	Functions of actual speed/position	180
6.6	Following Functions	187

6.1 Motion Status Bits

At a glance

Aim of this sub-chapter

This section describes the motion status bits which can be accessed via implicit exchange (bits %Ixy.i.0 to %Ixy.i.31) or by the GetStatus function.

What's in this Section?

This section contains the following topics:

Topic	Page
Bit RAMPING (%Ixy.i.0)	111
STEADY Bit (%Ixy.i.1)	112
STOPPING Bit (%Ixy.i.2)	113
PROFILE_END Bit (%Ixy.i.3)	114
IN_POSITION Bit (%Ixy.i.4)	115
AXIS_HOMING bit (%Ixy.i.5)	116
AXIS_HOMED bit (%Ixy.i.6)	117
AXIS_NOT_FOLLOWING bit (%Ixy.i.7)	118
HOLDING bit (%Ixy.i.8)	119
RESUMING bit (%Ixy.i.9)	120
DRIVE_ENABLED bit (%Ixy.i.10)	121
DRIVE_DIAG bit (%Ixy.i.11)	122
DRIVE_WARNING bit (%Ixy.i.12)	123
DRIVE_FAULT bit (%Ixy.i.13)	124
DRIVE_DISABLED bit (%Ixy.i.14)	125
AXIS_SUMMARY_FAULT bit (%Ixy.i.15)	126
AXIS_COMM_OK bit (%Ixy.i.16)	127
AXIS_IS_LINKED bit (%Ixy.i.17)	128
AXIS_AT_TARGET bit (%Ixy.i.20)	129
AXIS_POS_LIMIT bit (%Ixy.i.21)	130
AXIS_NEG_LIMIT bit (%Ixy.i.22)	131
AXIS_IN_COMMAND bit (%Ixy.i.18)	132
AXIS_WARNING bit (%Ixy.i.23)	133
BIAS_REMAIN Bit (%Ixy.i.24)	134
AXIS_MANUAL_MODE BIT (%Ixy.i.25)	135
DRIVE_REALTIME_BIT1 bit (%Ixy.i.26)	136
DRIVE_REALTIME_BIT2 bit (%Ixy.i.27)	137
AXIS_HOLD bit (%Ixy.i.28)	138
AXIS_HALT bit (%Ixy.i.29)	139
AXIS_FASTSTOP bit (%Ixy.i.30)	140
AXIS_READY bit (%Ixy.i.31)	141
CONF_OK Bit (%Ixy.i.32)	142

Bit RAMPING (%Ixy.i.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 the Co-ordinated Axes Group Function	Acceleration or deceleration profile type. The AXIS_IS_LINKED member axes also have their RAMPING bits set to 1.
Applied to a Set 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 bits in an axis are set to 1.

STEADY Bit (%Ixy.i.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 the Co-ordinated Axes Group Function	Uniform speed profile type. The AXIS_IS_LINKED member axes also have their STEADY bits set to 1.
Applied to the Co-ordinated Slave Axes Function	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 (%Ixy.i.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 (%Ixy.i.4) is set to 1.

The STOPPING bit is set to 0 when the PROFILE_END (%Ixy.i.3) 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.

Applied to Real or Imaginary Axis Functions

Deceleration type movement profile with movement stop.

Applied to the Co-ordinated Axes Group Function

Deceleration type movement profile with movement stop.

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

Applied to the Co-ordinated Slave Axes Function

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 bits in an axis are set to 1.

PROFILE_END Bit (%Ixy.i.3)

Description	This bit (status 1) indicates that the movement demanded has ended. The axis may still be settling until IN_POSITION (%Ixy.i.4) is set to 1 or STOPPING (%Ixy.i.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 the Co-ordinated Axes Group Function	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 the Co-ordinated Slave Axes Function	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 (%Ixy.i.4)

Description

This bit (status 1) indicates that axis position is in the in-position band and that the STOPPING (%Ixy.i.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 (%Ixy.i.3) is set to 1.

Applied to the Co-ordinated Axes Group Function

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 the Co-ordinated Slave Axes Function

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 (%Ixy.i.5)

Description	This bit (status 1) indicates that the axis is performing a Reference Point (see *Home Function, p. 154*) 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 (%Ixy.i.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 (%Ixy.i.7)

Description

This bit (status 1) indicates that the drive is ignoring the controller's movement 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 (%Ixy.i.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 the Co-ordinated Axes Group Function	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 (%Ixy.i.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 the Co-ordinated Axes Group Function	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 (%Ixy.i.10)

Description

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

Note: If the AXIS_READY (%Ixy.i.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 (%Ixy.i.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 (%Ixy.i.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 (%Ixy.i.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_FAULT 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_FAULT bits (AXIS_IS_LINKED) are set to 1.

DRIVE_DISABLED bit (%Ixy.i.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_FAULT bit (%Ixy.i.15)

Description

This bit (status 1) indicates either:

- a faulty drive (DRIVE_FAULT),
- a SERCOS® communication fault (AXIS_COM_OK to 0),
- a faulty movement profile controller (AXIS_POS_LIMIT or AXIS_NEG_LIMIT).

But also, errors in groups of slaves or coordinated axes (a new function common to the TSX CSY 84 / 164 modules) such as:

- the enabled groups that have a faulty axis will be reported as faulty (%Ixy.i.15 at 1), with information given by the GET_MOTION_FAULT function: fault on group member (bit 02 (MF_MEMBER_FAULT) of the MOTION_FAULT word).

as well as the propagation of faults and warnings (new function common to TSX CSY 84 / 164 modules):

- a real fault on the axes,

Information concerning the fault can be accessed with the help of the GetMotionFault (see *GetMotionFault Function, p. 323*) function.

Note: The command bit ALLOW_NOT_FAULT set to 0 triggers an "AXIS_SUMMARY_FAULT" fault.

Applied to Real or Imaginary Axis Functions

Movement Fault.

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

This bit is set to 1 if at least one AXIS_SUMMARY_FAULT of the member axes (AXIS_IS_LINKED) is set to 1.

AXIS_COMM_OK bit (%Ixy.i.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 (%Ixy.i.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 the Co-ordinated Axes Group Function

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 the Co-ordinated Slave Axes Function

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_AT_TARGET bit (%Ixy.i.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 the Co-ordinated Slave Axes Function

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 (%Ixy.i.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_FAULT bit rising edge in the current value or the fault to be removed.
Applied to Real and Imaginary Axis Functions	Position of the commanded axis on or over the upper limit.
Applied to the Group of Co-ordinated Axes, Group of Slave Axes and SERCOS® Ring Functions	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 (%Ixy.i.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_FAULT bit rising edge in the current value or the fault to be removed.
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_IN_COMMAND bit (%Ixy.i.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 the Co-ordinated Axes Group Function

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_COMMAND 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 the Co-ordinated Slave Axes Function

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_COMMAND 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_WARNING bit (%Ixy.i.23)

Description

This bit (status 1) indicates either:

- detection of a movement warning (see *Faults Accessible via the GetMotionFault Command, p. 321*).

But also, errors in groups of slaves or coordinated axes (a new function common to the TSX CSY 84 / 164 modules) such as:

- the non-enabled groups that have a faulty axis will be reported as a Warning (%Ixy.i.23 at 1), with information given by the GET_MOTION_WARNING function: **fault on member** (bit 05 (MW_MEMBER_FAULT) of the MOTION_WARNING) word.
  The associated text on the PL7 screen is: "Fault on group member".

It also indicates propagation of faults and warnings (new function common to TSX CSY 84 / 164 modules):

- axes that were stopped following propagation of faults (by the groups or monitoring function) will be reported as Warnings (%Ixy.i.23 at 1) with information given by the GET_MOTION_WARNING function: **fault on axis linked** (bit 02 (MW_STOP_BY_SET) of the MOTION_WARNING) word.
  The associated text on the PL7 screen is: "Stop propagated by group".
- the groups, whether enabled or non-enabled, which have a Warning axis will be reported as a Warning (%Ixy.i.23 at 1), with information given by the GET_MOTION_WARNING function:**Warning on a member** (bit 04 (MW_MEMBER_WARNING) of the MOTION_WARNING) word.
  The associated text on the PL7 screen is: "Alarm on group member".

The GetMotionWarning function allows the various types of warning for present movements to be read.

The AXIS_WARNING bit is reset using a CONTROL_CLEAR_FAULT rising edge.

Applied to Real and Imaginary Axis Functions

Detection of an axis movement warning.

Applied to a Group of Coordinated Axes

This bit is set to 1 if one AXIS_WARNING bit of the member axes (AXIS_IS_LINKED) is set to 1.

BIAS_REMAIN Bit (%Ixy.i.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 (%Ixy.i.25)

Description

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

- %Ixy.i.25 = 0: auto mode (by default),
- %Ixy.i.25 = 1: manual mode.

DRIVE_REALTIME_BIT1 bit (%Ixy.i.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 (%Ixy.i.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 (%Ixy.i.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_FAULT.
- 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 (%Ixy.i.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_FAST_STOP,
  - AXIS_HOMING,
  - DRIVE_DISABLED,
  - AXIS_SUMMARY_FAULT.
- 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 (%Ixy.i.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_FAULT 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 (%Ixy.i.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_FAULT 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 (%Ixy.i.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.

6.2 Motion control functions

At a glance

Aim of this sub-chapter

This section describes the motion control functions.

What's in this Section?

This section contains the following topics:

Topic	Page
Movement Control Functions: General	144
Management of ALLOW and CONTROL bits	145
MovementControl Bits Modifiable on Edge	146
Control Bits Modifiable on Change of State	149
Status Bits and Words	150
Manual Mode	151
Set_Functional_Mode Function	153
Home Function	154
Reference Point with the Home function	156
ForcedHomed Function	157
Unhome Function	158
SetPosition Function	159

Movement Control Functions: General

Movement Control Functions

Movement control functions allow a MovementSet (Real Axis, Imaginary Axis, External Measurement Value Axis, Group of Slave Axes, Network or Group of Co-ordinated Axes) to perform the following operations:

- Reinitialize fault states on a MovementSet (ClearFault),
- Activate or deactivate a MovementSet (EnableDrive and DisableDrive),
- Stop a MovementSet quickly or normally (FastStop and Halt),
- Hold or resume a MovementSet (Hold and resume),
- Home a movement axis (Home),
- Move the motion axis off the home position: (Unhome),
- Release and acquire command control of the individual axis members of a Slave Group or Co-ordinated Group (Release and Acquire),
- Activate or deactivate following of the master for a group of slave axes (FollowOn and FollowOff),
- Acquire command control and activate or release and deactivate slave axes in a slave group (Link and Unlink).

The functions that perform all these actions initiate a movement control command and then return to the application program, possibly before the command has completed.

The application program must monitor the Status of the MovementSet to determine if the command has been completed.

The application program monitors the axis status bit (%I) to check on the movement status.

Movement Control Commands

Many commands are sent via movement control command bits.

Some of the movement control commands require a corresponding ALLOW bit to be configured before the command can be executed successfully.

ALLOW bits are set to 1 by default when the module is initialized (except for the network channel ALLOW_ENABLE bit) so that all corresponding motion control commands are authorized.

Management of ALLOW and CONTROL bits

Principle

Movement control bits (see *Implicitly Exchanged Output Variables, p. 378*) are managed in the following way:

- set the channel 0 ALLOW_ENABLE bit (%Qxy.i.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 *Interface language, p. 359*).

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 reasons 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.

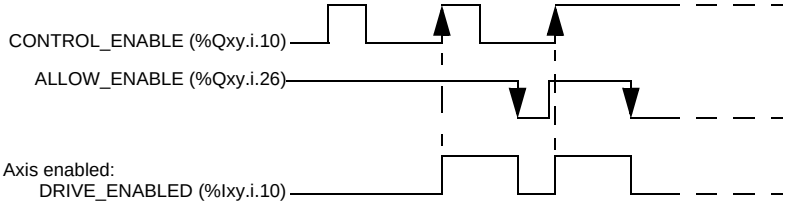
CONTROL bit names begin with CONTROL_ (see *Interface language, p. 359*).

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
%Qxy.i.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
%Qxy.i.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
%Qxy.i.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
%Qxy.i.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
%Qxy.i.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.
%Qxy.i.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.

Language object	Name	Description
%Qxy.i.31	ALLOW_NOT_FAULT	Falling edge: triggers a user error. The AXIS_SUMMARY_FAULT 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:

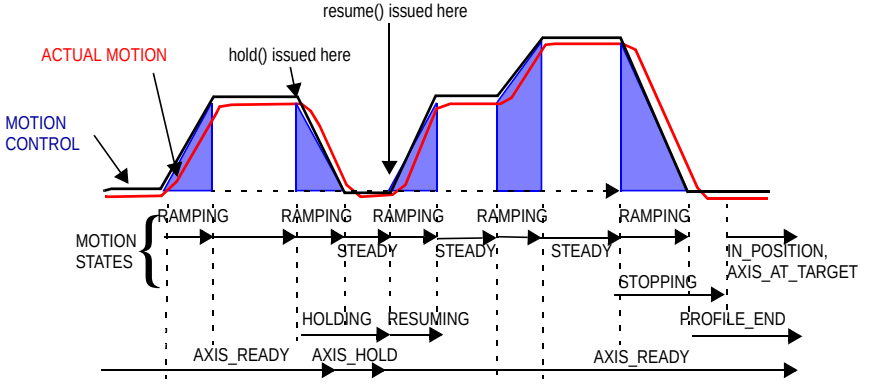
Language object	Name	Description
%Qxy.i.2	CONTROL_ ACQUIRE	Rising edge: command to acquire controlled axes and link them to the MovementGroup. The AXIS_IN_COMMAND bit is set if the operation is performed correctly. Associated ALLOW bit: ALLOW_ACQUIRE
%Qxy.i.6	REALTIME_CONTROL_BIT1	Rising edge: activates the IDN command when configured in the drive.
%Qxy.i.7	REALTIME_CONTROL_BIT2	Rising edge: activates the IDN command when configured in the drive.
%Qxy.i.10	CONTROL_ENABLE	Rising edge: activates the controlled axes. The DRIVE_ENABLED bit is set to 1 if the drive is enabled. Associated ALLOW bit: ALLOW_ENABLE
%Qxy.i.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
%Qxy.i.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
%Qxy.i.15	CONTROL_CLEAR_ FAULT	Rising edge: clears movement faults (MovementFault). The AXIS_SUMMARY_FAULT bit is set to 0 if the operation is performed correctly.
%Qxy.i.4	CONTROL_JOG_ POS	Rising edge: performs continuous movements in positive direction. Falling edge: stops the movement in progress
%Qxy.i.5	CONTROL_JOG_ NEG	Rising edge: performs continuous movements in negative direction. Falling edge: stops the movement in progress
%Qxy.i.13	CONTROL_INC_ POS	Rising edge: performs incremental movements in positive direction.
%Qxy.i.14	CONTROL_INC_NEG	Rising edge: performs incremental movements in 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 4.3 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 (Motion control for SERCOS - motion User manual - Speed Parameter: DefaultSpeed)),
- the movement speed using the Set_Speed_Override function (see (Motion control for SERCOS - motion User manual - GetSpeedOverride and SetSpeed-Override Function)),
- 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 (Motion control for SERCOS - motion User manual - Home Function) and (Motion control for SERCOS - motion User manual - ForcedHomed Function) and (Motion control for SERCOS - motion User manual - SetPosition Function)).

The following commands are not available in manual mode:

- MoveImmed (Motion control for SERCOS - motion User manual - MoveImmed Function) and MoveQueue (Motion control for SERCOS - motion User manual - MoveQueue Function),
- 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 (Motion control for SERCOS - motion User manual - Set_Functional_Mode Function)).

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, p. 368*).

Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**2572**	**Command**
%MDxy.i.20	PARAM_CMD_1	-	Value to write (1)
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.
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.

.



CAUTION

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.

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

Action of the Function on the Real Axis

The axis must be enabled and in AXIS_IN_COMMAND status. It must not be moving, holding (Hold) or faulted, and it must have the ALLOW_MOVE Movement-Control bit set to 1 in order for the command to be successful.

How to Use this Function

This function is implemented either:

- directly by bit position control (implicit exchanges),
- for using the WRITE_CMD instruction (see *WRITE_CMD, p. 368*).

Associated Language Objects

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

Object	Name	Effective	Description
%Qxy.i.29	ALLOW_MOVE	Status 1	Allow bit
%Qxy.i.228	ALLOW_RESUME	Status 1	Allow bit
%Qxy.i.30	ALLOW_NOT_FASTSTOP	Status 1	Allow bit
%Ixy.i.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
%MWxy.i.26	**ACTION_CMD**	**6034**	**Command**
%MFxy.i.31	PARAM_CMD_3	-	Direction of the reference point course: 1.0 (positive) or -1.0 (negative)
%MFxy.i.33	PARAM_CMD_4	-	Speed of the reference point course in SpeedUnits
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

Reference Point with the Home function

How to Get the Reference Point with SERCOS® Drives

The source 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. Positioning this transition within a band of +/- 45 degrees from the zero pulse of the resolver or encoder top results in a reference point fault (unless the reference point fault is deactivated by the drive's configuration software). The position of the switch can be determined by reading the reference point switch position error parameter, SERCOS® IDN S-0298 or IDN US-0298, if supported by the drive.

The reference point 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 the reference point to be abandoned with an AXIS_SUMMARY_FAULT and leave the axis in the non-source state. If the Hold function is issued during the first step, the reference point is aborted with an AXIS_SUMMARY_FAULT, leaving the axis in the non-source state. 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 reference point switch but not in the specified source direction. (See the reference point 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 quit 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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**6039**	**Command**
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

Unhome Function

Description

This function cancels the reference point of an axis.

CAUTION: 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. Use the end of travel (EOT) switches connected directly to the drive to protect the axis machinery from inadvertent motion when the axis is not referenced.

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD, p. 368*).

Associated Language Objects

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

Object	Name	Effective	Description
%Ixy.i.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
%MWxy.i.26	**ACTION_CMD**	**6038**	**Command**
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

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

In the case of a sequence with **BIAS_REMAIN**, the recognition on the slave of the resulting offset is signalled on the %Ixy.i.24 bit. As long as the bit is at 1, the **SET_POSITION** command is forbidden.

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

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**2053**	**Command**
%MFxy.i.31	PARAM_CMD_3	-	Value to write.
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

6.3 Move Functions

At a glance

Aim of this sub-chapter

This section describes move functions.

What's in this Section?

This section contains the following topics:

Topic	Page
About Movement Functions	161
MoveType Data Type	163
MoveImmed Function	164
MoveQueue Function	166
GetMoveQueueLength Function	168
GetSpeedOverride and SetSpeedOverride Function	169
EnableRealTimeCtrlBit Function	170

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 modulos are activated).
  ABS_MOVE_POS: Absolute movement in the positive direction for axes with modulos activated or co-ordinated axes.
  ABS_MOVE_NEG: Absolute movement in the negative direction for axes with modulos activated or co-ordinated axes.
  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 *Status Bits and Words, p. 150*).

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 (see *Overview of the Function, p. 172*).

**Function not dealt with in this chapter; see the following chapter (Motion control for SERCOS - motion User manual - Movement Function for coordinated group compatible with the MMS Quantum function).

MoveImmed 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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**513**	**Command**
%MDxy.i.27	PARAM_CMD_1	-	Movement Type (1)
%MDxy.i.29	PARAM_CMD_2	-	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 10{,}000$). see opcode 2172, real axis (see *Adjustment Functions, p. 396*) or imaginary axis (see *Adjustment Functions, p. 400*).
%MFxy.i.A (3)	PARAM_CMD_ B (3)	-	Target position of axis N (2)
%MFxy.i.C (4)	PARAM_CMD_D (4)	-	Approach speed to target of N' axis (5)
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.
Legend			
(1): INCR_MOVE, ABS_MOVE, ABS_MOVE_POS, ABS_MOVE_NEG 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 (%MDxy.i.27) = Movement type
- PARAM_CMD_3 (%MFxy.i.31) = Position of axis 1
- PARAM_CMD_4 (%MFxy.i.33) = Position of axis 2
- PARAM_CMD_5 (%MFxy.i.35) = Approach speed of axis 1
- PARAM_CMD_6 (%MFxy.i.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 movement command. 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

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
%MWxy.i.26	**ACTION_CMD**	**520**	**Command**
%MDxy.i.27	PARAM_CMD_1	-	Movement Type (1)
%MDxy.i.29	PARAM_CMD_2	-	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 10{,}000$). see opcode 2172, real axis (see *Adjustment Functions, p. 396*) or imaginary axis (see *Adjustment Functions, p. 400*).
%MFxy.i.A (3)	PARAM_CMD_ B (3)	-	Target position of axis N (2)
%MFxy.i.C (4)	PARAM_CMD_D (4)	-	Approach speed to target of N' axis (5)
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.
Legend			
(1): INCR_MOVE, ABS_MOVE, ABS_MOVE_POS, ABS_MOVE_NEG 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 (%MDxy.i.27) = Movement type
- PARAM_CMD_3 (%MFxy.i.31) = Position of axis 1
- PARAM_CMD_4 (%MFxy.i.33) = Position of axis 2
- PARAM_CMD_5 (%MFxy.i.35) = Approach speed of axis 1
- PARAM_CMD_6 (%MFxy.i.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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**9510**	**Command**
%MDxy.i.20	RETURN_CMD_1	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

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, p. 368*).

Associated Language Objects

Exchange via WRITE_CMD: The table below presents the different language objects linked to these functions.

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**1513**	**Read command**
		2513	**Write command**
%MFxy.i.22	RETURN_CMD_2	-	Result of the read command
%MFxy.i.31	PARAM_CMD_3	-	Value to write)
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

EnableRealTimeCtrlBit Function

Description

This function allows you to enable or disable the RealTime control bits 1 and 2 (%Qxy.i.6 and %Qxy.i.7 respectively).

Enabling the bits causes them to be sent to the drive over 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 WRITE_CMD instruction WRITE_CMD (see *WRITE_CMD, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**2562**	**Command**
%MDxy.i.27	PARAM_CMD_1	-	1 = Enable 0 = Disable
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

6.4 Motion Function Following Deactivation of the Sequence

At a Glance

Subject 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's in this Section?

This section contains the following topics:

Topic	Page
Overview of the Function	172
"MoveImmed" Function	173
"MoveQueue" Function	176

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 *"MoveImmed" Function, p. 173*)" and "MoveQueue (see *"MoveQueue" Function, p. 176*)" commands, with different types of motions.

Type of motion

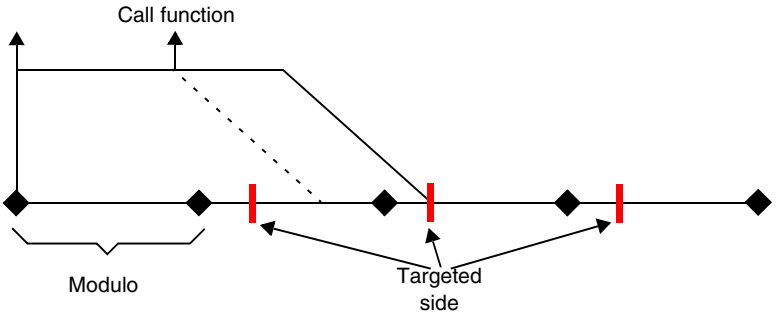
The following table lists the various values of the data type:

Type of motion	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)

"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 *MoveImmed Function, p. 164*) 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. This bit is set to 0 by the system after the motion is executed.

Object	Type	Symbol	Explanation	Comment
%MW xy.i.19	Word	ERROR_CMD		Error Code*
%MD xy.i.20	Double word	RETURN_CMD_1		not used
%MF xy.i.22	Floating point	RETURN_CMD_2		not used
%MF xy.i.24	Floating point	RETURN_CMD_3		not used
%MW xy.i.26	Word	ACTION_CMD	513	MoveImmed
%MD xy.i.27	Double word	PARAM_CMD_1	10 or 12 or 13 or 14	Type
%MD xy.i.29	Double word	PARAM_CMD_2	Acceleration / deceleration coefficient	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 10{,}000$). See opcode 2172, real axis (see *Adjustment Functions, p. 396*) or imaginary axis (see *Adjustment Functions, p. 400*).
%MF xy.i.31	Floating point	PARAM_CMD_3	Target position	
%MF xy.i.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 *Motion Warnings, p. 325*):

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):

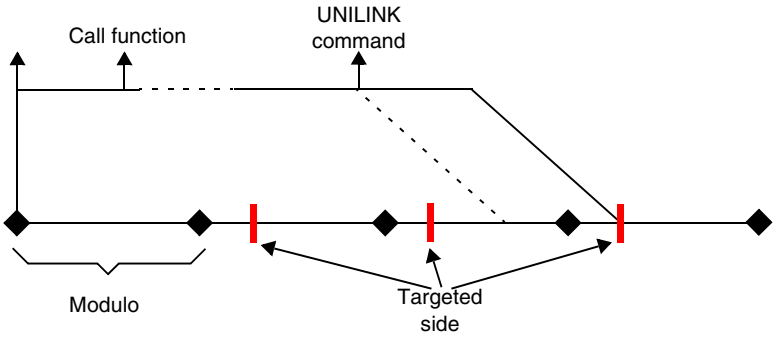
- if the axis is stopped, the motion will be "at most" one rotation,
- if the axis is previously in motion:
  - distance to be traveled and deceleration incline permitting, the axis will be able to switch to the requested target speed before stopping,
  - otherwise, the module will calculate the minimum number of revolutions to perform in order to stop at the absolute requested position (respecting the deceleration incline).

"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 *MoveQueue Function, p. 166*) 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
%MW xy.i.19	Word	ERROR_CMD		Error code
%MD xy.i.20	Double word	RETURN_CMD_1		not used
%MF xy.i.22	Floating point	RETURN_CMD_2		not used
%MF xy.i.24	Floating point	RETURN_CMD_3		not used
%MW xy.i.26	Word	ACTION_CMD	520	MoveQueue
%MD xy.i.27	Double word	PARAM_CMD_1	Type	10 or 12 or 13 or 14
%MD xy.i.29	Double word	PARAM_CMD_2	Acceleration / deceleration coefficient	Coefficient applied to the acceleration and deceleration configured, shown for one thousand ($1 \leq x \leq 10{,}000$). See opcode 2172, real axis (see *Adjustment Functions, p. 396*) or imaginary axis (see *Adjustment Functions, p. 400*).
%MF xy.i.31	Floating point	PARAM_CMD_3	Target position	
%MF xy.i.33	Floating point	PARAM_CMD_4	Target speed	

Command rejection:

- all existing cases of "MoveQueue" command rejection (see *WRITE_CMD Command: Programming Errors, p. 340*),
- 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 *Motion Warnings, p. 325*):

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 *AXIS_READY bit (%Ixy.i.31), p. 141*),
- 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).

6.5 Functions of actual speed/position

Introduction

Aim of this sub-chapter

This sub-chapter describes the functions of actual speed and position.

What's in this Section?

This section contains the following topics:

Topic	Page
GetActualSpeedFunction	181
GetCommandSpeedFunction	182
Modulo counter service	183
GetUnrolledPosition Function	184
GetUnrolledCommandedPosition Function	185
GetCommandedPosition Function	186

GetActualSpeedFunction

Description

The GetActualSpeed function returns the current movement speed value for the specified axis.

How to Use this Function

This function is implemented with the help of the WRITE_CMD instruction (see *WRITE_CMD, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**5065**	**Command**
%MFxy.i.22	RETURN_CMD_2	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

GetCommandSpeedFunction

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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**5066**	**Command**
%MFxy.i.22	RETURN_CMD_2	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

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 xy.i.19	Word	ERROR_CMD		
%MD xy.i.20	Double word	RETURN_CMD_1	NB_MODULO	Modulo number (positive or negative direction)
%MF xy.i.22	Floating point	RETURN_CMD_2		
%MF xy.i.24	Floating point	RETURN_CMD_3		
%MW xy.i.26	Word	ACTION_CMD	573	ReadNBModulo
%MD xy.i.27	Double word	PARAM_CMD_1	0	
%MD xy.i.29	Double word	PARAM_CMD_2	0	
%MF xy.i.31	Floating point	PARAM_CMD_3	0	
%MF xy.i.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 *Home Function, p. 154*),
- a position forcing 2053 (see *SetPosition Function, p. 159*) 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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**546**	**Command**
%MFxy.i.22	RETURN_CMD_2	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**547**	**Command**
%MFxy.i.22	RETURN_CMD_2	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**1053**	**Command**
%MFxy.i.22	RETURN_CMD_2	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

6.6 Following Functions

Introduction

Aim of this sub-chapter

This sub-chapter describes the following functions.

What's in this Section?

This section contains the following topics:

Topic	Page
About Following Functions	188
"Target sequence activation" function	190
"Estimation of Remaining Time Before Activation of Sequence" Function	192
Sequence stop (Unlink) of the slave group set off by a trigger	194

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 (%Ixy.i.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.

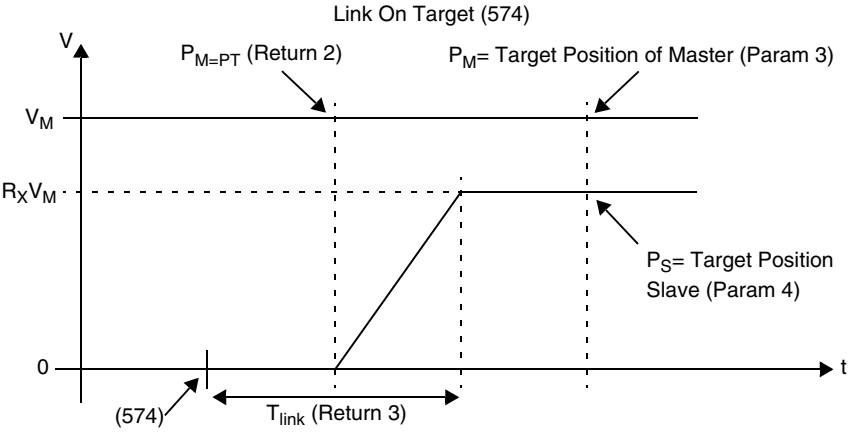
This synchronization is performed using the "**SetMasterTriggerPosition**" and "**FollowOn**" functions.

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 *AXIS_IS_LINKED bit (%Ixy.i.17), p. 128*)". 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
%MW xy.i.19	Word	ERROR_CMD		Command rejected *
%MD xy.i.20	Double word	RETURN_CMD_1		
%MF xy.i.22	Floating point	RETURN_CMD_2		Activation position of the master (equivalent to the "GetMasterTriggerPosition (see *Reading the Last Position Trigger Assigned by the SetTriggerUnlink Command, p. 195*)" function)
%MF xy.i.24	Floating point	RETURN_CMD_3		Time remaining before sequence activation of the slave (in seconds)
%MW xy.i.26	Word	ACTION_CMD	574	LinkOnTarget
%MD xy.i.27	Double word	PARAM_CMD_1	SlaveNumber	SERCOS address of the slave to link
%MD xy.i.29	Double word	PARAM_CMD_2	0	not used
%MF xy.i.31	Floating point	PARAM_CMD_3	MasterPosition	Target position of master axis
%MF xy.i.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 *AXIS_IS_LINKED bit (%Ixy.i.17), p. 128*)").

"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 *AXIS_IS_LINKED bit (%Ixy.i.17), p. 128*)". 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
%MW xy.i.19	Word	ERROR_CMD		Command rejected *
%MD xy.i.20	Double word	RETURN_CMD_1		
%MF xy.i.22	Floating point	RETURN_CMD_2		Estimation of time remaining before sequence activation of the slave
%MF xy.i.24	Floating point	RETURN_CMD_3		
%MW xy.i.26	Word	ACTION_CMD	575	GetTimeToLink
%MD xy.i.27	Double word	PARAM_CMD_1	SlaveNumber	SERCOS address of the slave
%MD xy.i.29	Double word	PARAM_CMD_2		
%MF xy.i.31	Floating point	PARAM_CMD_3		
%MF xy.i.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 *AXIS_IS_LINKED bit (%Ixy.i.17), p. 128*)").

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
%MW xy.i.19	Word	ERROR_CMD		Error code*
%MD xy.i.20	Double word	RETURN_CMD_1		
%MF xy.i.22	Floating point	RETURN_CMD_2		
%MF xy.i.24	Floating point	RETURN_CMD_3		
%MW xy.i.26	Word	ACTION_CMD	2180	SetTriggerUnlink
%MD xy.i.27	Double word	PARAM_CMD_1	Slave number	
%MD xy.i.29	Double word	PARAM_CMD_2	Trigger type	Zero (0), ** > (1), < (2), ≥ (3), ≤ (4).
%MF xy.i.31	Floating point	PARAM_CMD_3	Trigger position	
%MF xy.i.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
%MW xy.i.19	Word	ERROR_CMD		Error code*
%MD xy.i.20	Double word	RETURN_CMD_1		
%MF xy.i.22	Floating point	RETURN_CMD_2	Trigger position	
%MF xy.i.24	Floating point	RETURN_CMD_3		
%MW xy.i.26	Word	ACTION_CMD	1180	GetTriggerUnlinkPosition
%MD xy.i.27	Double word	PARAM_CMD_1	Slave number	
%MD xy.i.29	Double word	PARAM_CMD_2		
%MF xy.i.31	Floating point	PARAM_CMD_3		
%MF xy.i.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
%MW xy.i.19	Word	ERROR_CMD		Error code*
%MD xy.i.20	Double word	RETURN_CMD_1	Trigger type	Zero (0), > (1), < (2), ≥ (3), ≤ (4).
%MF xy.i.22	Floating point	RETURN_CMD_2		
%MF xy.i.24	Floating point	RETURN_CMD_3		
%MW xy.i.26	Word	ACTION_CMD	1181	GetTriggerUnlinkType
%MD xy.i.27	Double word	PARAM_CMD_1	Slave number	
%MD xy.i.29	Double word	PARAM_CMD_2		
%MF xy.i.31	Floating point	PARAM_CMD_3		
%MF xy.i.33	Floating point	PARAM_CMD_4		

* Error code: command rejected if the slave number is invalid.

Adjustment of TSX CSY 84 / 164 modules

7

At a Glance

Subject 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's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	SERCOS® Module Adjustment: General	198
7.2	Adjustment of the Parameters of a Real, Imaginary or Remote Axis	203
7.3	Parameters of a Group of Slave Axes	238
7.4	Cam Profile Adjustment	262
7.5	Sercos® ring parameters	274
7.6	SERCOS® IDN Read/Write Functions	278

7.1 SERCOS® Module Adjustment: General

Introduction

Aim of this sub-chapter

This sub-chapter introduces the adjustment screens of the SERCOS® module and gives a reminder of some general principles of explicit exchange.

What's in this Section?

This section contains the following topics:

Topic	Page
Description of the Adjustment Screens of SERCOS® Module: General	199
Adjustment Principles	202

Description of the Adjustment Screens of SERCOS® Module: General

Introduction

The adjustment screen displays the adjustment parameters of a SERCOS® channel.

These parameters can be modified in offline or online mode.

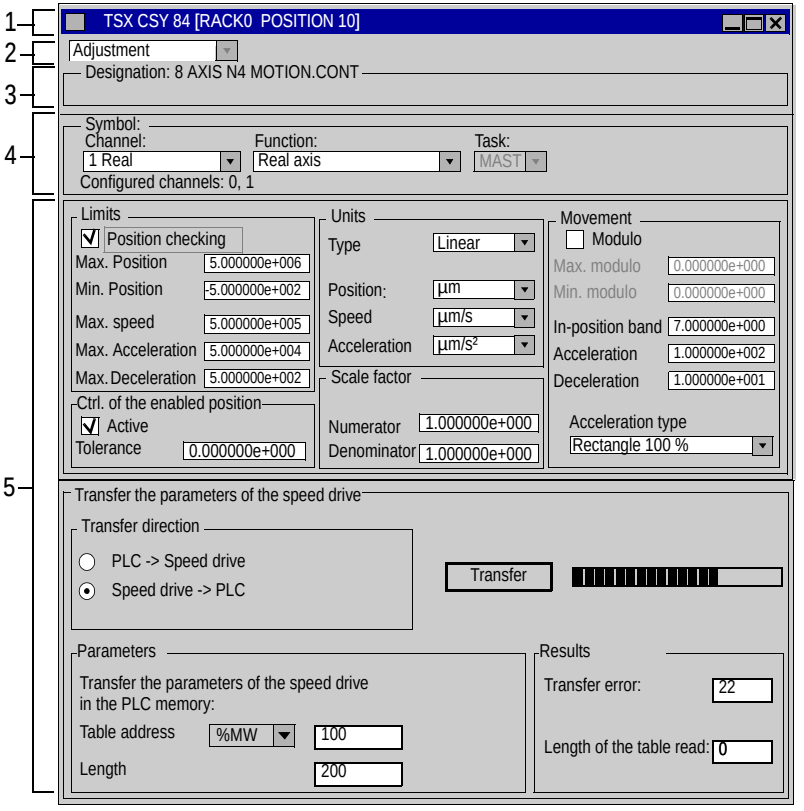
It also gives access to the configuration and debug screen.

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 an application, consult the section on Common application 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 adjustment screen and their functions.

Address	Identification	Function
1	Title bar	Recalls the reference of the module selected and its physical position in the rack.
2	Command zone	Indicates the screen in progress (**Adjustment**) and lets you go or return to other screens: ● **Configuration** ● **Debug** (or diagnostic), accessible only in online mode.
3	Module zone	Displays the designation of the module selected. Displays indicator lamp status **RUN**, **ERR**, **I/O** Creates direct access to the module diagnostics when it is faulty (indicated by the indicator lamp built into the diagnostics access button DIAG, which turns red).
4	Channel zone	Enables you to choose the channel to be controlled, Indicates the task in which the channel is configured, Provides direct access to the diagnostics of the channel when it is faulty
5	Adjustment zone	This zone is made up of different specific categories according to the type of channel.

Adjustment Principles

Introduction

The adjustment function is used to view and modify the operating parameters of the module.

the TSX CSY 84 / 164 module parameters are adjusted either:

- directly using the adjustment screens (offline or online),
- or using explicit exchanges (see *PL7 Interface, p. 361*) between the application and the axis command 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).

7.2 Adjustment of the Parameters of a Real, Imaginary or Remote Axis

Introduction

Aim of this sub-chapter

This sub-chapter introduces the adjustment screens of a real, imaginary or remote axis as well as the different accessible parameters.

What's in this Section?

This section contains the following topics:

Topic	Page
Description of the Adjustment Screen of a SERCOS® Module: Real or imaginary axes	204
Description of the Adjustment Screen of a SERCOS® Module: Remote Axes	207
Description of the TRF_RECIPE Screen for Real Axes	209
TRF_RECIPE	212
TRF_RECIPE Examples	214
Speed Parameter: DefaultSpeed	215
Speed Parameter: SpeedLimit	216
About acceleration/deceleration parameters	217
Acceleration/Deceleration Parameter: Accel	218
Acceleration/Deceleration Parameter: Decel	219
Acceleration/Deceleration Parameter: AccelMax	220
Acceleration/Deceleration Parameter: DecelMax	221
Acceleration/Deceleration Parameter: AccelType	222
AccelerationType Data type	224
Resolution Parameters: GearRatio	225
Position Limit and Modulo Parameter: PositionLimit	227
About the Modulo	229
Position Limit and Modulo Parameter: RolloverLimit	231
Position Limit and Modulo Parameter: EnableRollover	232
About the In-Position Band	233
Position Limit Parameter: InPositionBand	234
About Enable Mode	235
Position Limit Parameter: EnableMode	236
Behavior Following Axis Disable	237

Description of the Adjustment Screen of a SERCOS® Module: Real or imaginary axes

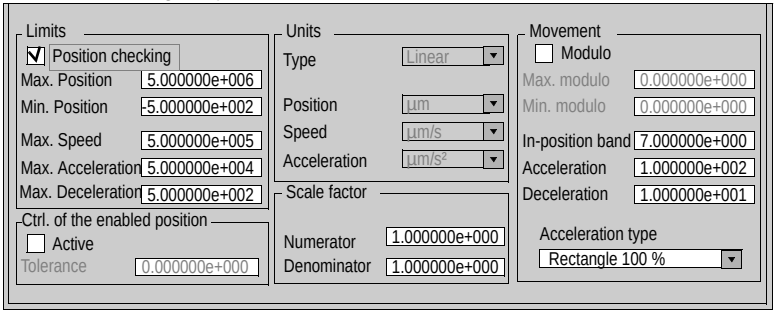
Introduction

This screen is used to directly modify the configuration parameters of a real or imaginary axis in online mode.

Note: For more information on the common parts of the different adjustment screens, please see *Description of the Adjustment Screens of SERCOS® Module: General, p. 199*.

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 **Limit** of the adjustment zone.

Element	Associated parameter	Bit/ word language associated	Symbol/description
Position checking	-	%MWxy.i.35:X4	This check box is used to activate the control of position limits.
Max. Position	PositionLimits	%MWxy.i.55	POSITION_MAX
Min. Position		%MWxy.i.57	POSITION_MIN
Max. speed	SpeedLimit	%MWxy.i.53	SPEED_MAX
Max. Acceleration	AccelMax	%MWxy.i.49	ACCEL_MAX
Min. Acceleration	DecelMax	%MWxy.i.	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	-	%MWxy.i.35:X1	This checkbox can be used to enable position checking.
Tolerance	-	%MWxy.i.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	-	%KWxy.i.34	-
Position	PositionUnits	%MWxy.i.65	-
Speed	VelocityUnits	%MWxy.i.64	-
Acceleration	AccelUnits	%MWxy.i.63	-

The table below shows the **Scale factor** of the adjustment zone.

Element	Associated parameter	Language word associated	Symbol
Numerator	GearRatio	%MWxy.i.59	SCALE_NUMERATOR
Denominator		%MWxy.i.61	SCALE_DENOMINATOR

The table below shows the **Movement** of the adjustment zone.

Element	Associated parameter	Bit/ word language associated	Symbol
Modulo	EnableRollover	%MWxy.i.35:X0	ENABLE_ROLLOVER
Max. modulo	RolloverLimit	%MWxy.i.45	ROLLOVER_MAX
Min. modulo		%MWxy.i.47	ROLLOVER_MIN
In-position band	InPositionBand	%MFxy.i.41	IN_POSITION_BAND
Acceleration	Accel	%MWxy.i.36	ACCEL
Deceleration	Decel	%MWxy.i.38	DECEL
Acceleration type	AccelType	%MWxy.i.40	ACCEL_TYPE

Description of the Adjustment Screen of a SERCOS® Module: Remote Axes

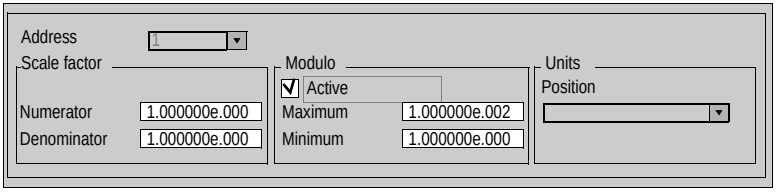
Introduction

This screen is used to directly modify the configuration parameters of a remote axis in online mode.

Note: For more information on the common parts of the different adjustment screens, please see *Description of the Adjustment Screens of SERCOS® Module: General, p. 199*.

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	%MFxy.i.Y (2)	NUMERATOR_X (1)
Denominator			DENOMINATOR_X (1)
Legend			
(1): X included between 1 and 8			
(2): Value of Y, See *Slave Group Parameter: FollowerRatio, p. 248*.			

The table below shows the **Modulo** of the adjustment zone.

Element	Associated parameter	Bit/ word language associated	Symbol/description
Active	EnableRollover	%MWxy.i.35:X0	ENABLE_ROLLOVER
Maximum	RolloverLimit	%MWxy.i.45	ROLLOVER_MAX
Minimum		%MWxy.i.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	%MWxy.i.65	-

Description of the TRF_RECIPE Screen for Real Axes

Introduction

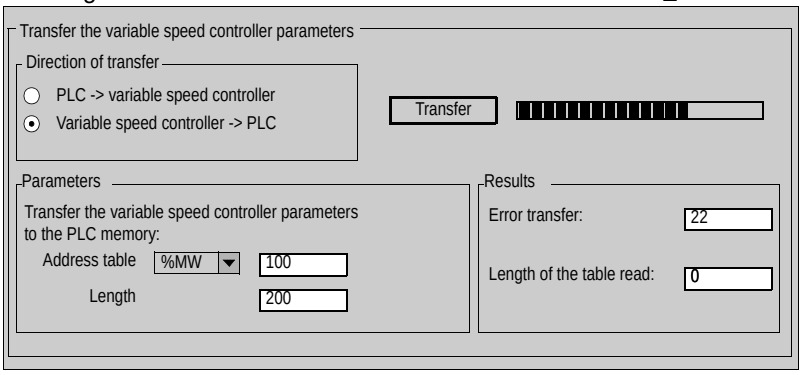
This screen, situated at the bottom of the adjustment screen of a real axis is used to transfer adjustment parameters between the PLC at the variable speed controller

This functionality has to be in online mode.

Note: For more information on the common parts of the different adjustment screens, please see *Description of the Adjustment Screens of SERCOS® Module: General, p. 199*.

Illustration

The diagram below shows the screen associated with the TRF_RECIPE function.



Description

The table below shows the **Direction of transfer** of the TRF_RECIPE zone.

Element	**Language word associated**	**Symbol**
PLC -> variable speed controller	%MWxy.i.10	ACTION_TRF
Variable speed controller ->PLC		

The table below shows the **Parameters** of the TRF_RECIPE zone.

Element	Language word associated	Description
Address table	%MDxy.i.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	%MWxy.i.3	This field provides the error code (see *TRF_RECIPE Command Explicit Writing Faults, p. 348*) in the case of an invalid transfer.
Length of the table read	%MDxy.i.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 %CHxy.i (length, address %MW): transfers parameters from the variable speed controller, the cam profile, or parameters of a special function to/from the table which begins with 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 %MWxy.i.10 (ACTION_TRF).

Example: TRF_RECIPE %CH104.3(10,100) (load variable speed controller parameters of real axis 3, of the module situated in position 4 of rack 1, to length table 10, which begins with address %MW100).

TRF_RECIPE Interface

The command to be carried out is defined in the word %MWxy.i.10 and the result of the command is available in the word %MWxy.i.3 to %MWxy.i.8.

Address	Type	Symbol	Meaning
%MWxy.i.3	Word	ERROR_TRF	Write error of the TRF_RECIPE command
%MDxy.i.4	Double Word	RETURN_TRF_1	Return 1 of the function
%MFxy.i.6	Floating point	RETURN_TRF_2	Return 2 of the function
%MFxy.i.8	Floating point	RETURN_TRF_3	Return 3 of the function
%MWxy.i.10	Word	ACTION_TRF	Action to be carried out
%MDxy.i.11	Double Word	PARAM_TRF_1	Parameter 1
%MDxy.i.13	Double Word	PARAM_TRF_2	Parameter 2
%MFxy.i.15	Floating point	PARAM_TRF_3	Parameter 3
%MFxy.i.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 (%MWxy.i.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 to the saved memory (Real axis, WRITE_CMD)
2	Unwrite parameters from the saved memory (Real Axis, WRITE_CMD)
3	Write parameters to 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.

```
!  (*If no exchange of parameters is in progress, then transfer the
   parameters table into the PLC*)
   %MW104.3.10 := 16001;
   TRF_RECIPE %CH104.3(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 half 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
%MWxy.i.26	**ACTION_CMD**	**1065**	**Read command**
		2065	**Write command**
%MFxy.i.22	RETURN_CMD_2	-	Result of the read.
%MFxy.i.31	PARAM_CMD_3	-	Value to write.
%MWxy.i.19	ERROR_CMD	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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
%MFxy.i.53	SPEED_MAX	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1066**	**WRITE_CMD**	**Read command**
		2066		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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
%MFxy.i.36	ACCEL	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1041**	**WRITE_CMD**	**Read command**
		2041		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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
%MFxy.i.38	DECEL	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1042**	**WRITE_CMD**	**Read command**
		2042		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

Acceleration/Deceleration Parameter: AccelMax

Description

This parameter is used to define the maximum acceleration applied to an axis.

The value of this acceleration is specified as AccelerationUnits.

Note: The maximum acceleration must be greater than or equal to the actual acceleration parameter.

Note: The servo drive's maximum acceleration 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 parameter.

Associated Functions

The associated functions of this parameter are:

- GetAccelMax which sends a read request for the actual maximum acceleration value,
- SetAccelMax 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 AccelMax parameter.

Object	Name	Function code	Instruction type	Description
%MFxy.i.49	ACCEL_MAX	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1116**	**WRITE_CMD**	**Read command**
		2116		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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 as 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
%MFxy.i.51	DECEL_MAX	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1117**	WRITE_CMD	**Read command**
		2117		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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 = Taccrec
1	TRIANGULAR_ACCEL	Tacc = 2Taccrec
2	ACCEL_TRAP_125	t2 = 3t1, Tacc = 1.25Taccrec
3	ACCEL_TRAP_150	t1 = t2, Tacc = 1.5Taccrec
4	ACCEL_TRAP_175	t1 = 3t2, Tacc = 1.75Taccrec

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
%MFxy.i.40	ACCEL_TYPE	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1540**	**WRITE_CMD**	**Read command**
		2540		**Write command**
%MFxy.i.20	RETURN_CMD_1	-	-	Result of the read.
%MFxy.i.27	PARAM_CMD_1	-	-	Value to write.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

AccelerationType Data type

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 jerk 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 than the triangular but less than the rectangular. There are three choices of trapezoidal acceleration profiles, 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 values of the data type:

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
%MFxy.i.59	SCALE_NUMERATOR	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MFxy.i.61	SCALE_DENOMINATOR	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1500**	**WRITE_CMD**	**Read command**
		2500		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read (Numerator).
%MFxy.i.24	RETURN_CMD_3	-	-	Result of the read (Denominator).
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write (Numerator).
%MFxy.i.33	PARAM_CMD_4	-	-	Value to write (Denominator).
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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 defined with PositionUnits units.

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,
- SePositionLimit 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, WRITE_CMD, READ_PARAM or WRITE_PARAM instructions.

	**CAUTION** When an incremental movement is required and the target position exceeds a **position limit** (%MFxy.i.55 and %MFxy.i.57 of the real or imaginary axis), the 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. **Failure to follow these instructions can result in injury or equipment damage.**

Language Objects

The table below presents the different language objects linked to the PositionLimit parameter.

Object	Name	Function code	Instruction type	Description
%MFxy.i.55	POSITION_MAX	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MFxy.i.57	POSITION_MIN	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1505**	**WRITE_CMD**	**Read command**
		2505		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MDxy.i.27	PARAM_CMD_1	-	-	Value to write: 1: positive limit -1: negative limit
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write: Position.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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 modulo 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 measurment 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 applications, you may want to set the position limits to values greater than the modulo limits. For example, suppose your application 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 an application, 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 application 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 its modulo to be enabled to use this function. This function does not activate the modulo. (Use the EnableRollover function). Modulo 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
%MFxy.i.45	ROLLOVER_MAX	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MFxy.i.47	ROLLOVER_MIN	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1539**	**WRITE_CMD**	**Read command**
		2539		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MDxy.i.27	PARAM_CMD_1	-	-	Value to write: 1: positive limit -1: negative limit
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write: modulo
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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
%MWxy.i.35:X0	ENABLE_ROLLOVER	1	WRITE_PARAM	Write command: Activation
		0		Write command: Deactivation
%MWxy.i.26	**ACTION_CMD**	**411**	**WRITE_CMD**	**Write command: Activation**
		412		**Write command: Deactivation**
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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
%MFxy.i.41	IN_POSITION_BAND	-	READ_PARAM	Read command
			WRITE_PARAM	Write command
%MWxy.i.26	**ACTION_CMD**	**1035**	**WRITE_CMD**	**Read command**
		2035		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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 diasabled 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 command 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 CURRENT VALUE mode and is the recommended mode.



WARNING

You should set the SETPOINT enable mode only after reading the enable mode description and clearly understanding that if the axis moves less than the EnablePositionBand while deactivated, the axis will "snap" back to the last setpoint position prior to deactivation. Adequate safeguards, such as axis brakes with no manual brake release and small, activated following error limits, should be used to prevent undesirable movement while activated.

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
%MWxy.i.26	**ACTION_CMD**	**1524**	**Read command**
		2524	**Write command**
%MDxy.i.20	RETURN_CMD_1	-	Result of the read.
%MDxy.i.27	PARAM_CMD_1	-	Value to write: Activation mode. 0: Current value 1: Setpoint
%MWxy.i.19	ERROR_CMD	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

Behavior Following 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 %MWxy.i.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.

7.3 Parameters of a Group of Slave Axes

At a Glance

Aim of this Section

This section describes the adjustment functions for a group of slave axes.

What's in this Section?

This section contains the following topics:

Topic	Page
Description of the Adjustment Screen of a SERCOS® Module: Slave Axes	239
About a Group of Slave Axes	241
SlaveMode Data	244
Slave Group Parameters: FollowerMode	245
Slave Group Parameter: FollowerConfig	246
Slave Group Parameter: FollowerRatio	248
About the Master Offset	251
Slave Group Parameters: MasterOffset	252
Slave Group Parameters: MasterTriggerPosition	254
Slave Group Parameters: FollowerBias	256
Slave Group Parameters: AbsFollowerBias	257
Propagation of Stops in a Slave Group	259
Alignment Function of the Emergency Stop Inclines	260

Description of the Adjustment Screen of a SERCOS® Module: Slave Axes

Introduction

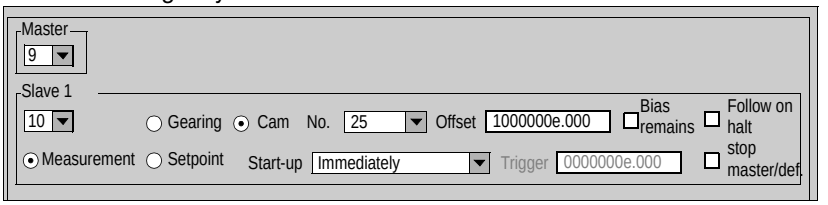
This screen is used to directly modify the configuration parameters of a slave axis in local or online mode.

Note: It is not possible to delete a master or a slave.

Note: For more information on the common parts of the different adjustment screens, please see *Description of the Adjustment Screens of SERCOS® Module: General, p. 199*.

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	%MWxy.i.35	Number of the master axis (1 to 16, N is not accessible)
Slave	SLAVE_CHANNEL_Z	%MWxy.i.36	Number of slave axis Z
Reading/ setpoint	FOLL_TYPE_Z	%MWxy.i.37:X2	Following type of slave Z
Gearing/Cam	FOLL_POSITION_Z)	%MWxy.i.37:X1	Slave Z ratio or cam mode)
N°	NUMERATOR_Z	%MFxy.i.38	Slave Z numerator
Offset	DENOMINATOR_Z	%MFxy.i.40	Slave Z denominator
Start-up	FOLL_START_Z	%MWxy.i.37:X8 to X10	Start conditions
Bias	FOLL_BIAS_REMAINS_Z	%MWxy.i.37:X7	Bias value
Trigger	TRIGGER_POSITIO_Z	%MFxy.i.42	Value of trigger for slave axis Z
Follow on halt	-	%MWxy.i.37:X3	Halt of slave axis Z if link deleted.
Stop master/def	-	%MWxy.i.37:X6	Stop the master in the event of a following 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_FOLLOW_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
%MWxy.i.26	**ACTION_CMD**	**1529**	**WRITE_CMD**	**Read command**
%MDxy.i.20	RETURN_CMD_1	-	-	Result of the read.
%MDx'y'.i.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)
Legend				
xy	Slave group address (channel between 21 and 24)			
x'y'	Slave axis address (channel between 1 and 12)			
(1)	This identifier is contained in the constant word %KWxy.i.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
%MWxy.i.26	**ACTION_CMD**	**420**	**WRITE_CMD**	**Write command**
%MDx'y'.i.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MDxy.i.29	PARAM_CMD_2	-	-	Value to write: Follower mode. See *SlaveMode Data, p. 244*
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write: Numerator for ratio mode, Movement profile for cam mode.
%MFxy.i.33	PARAM_CMD_4	-	-	Value to write: Denominator for ratio
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

Legend	
xy	Slave group address (channel between 21 and 24)
x'y'	Slave axis address (channel between 1 and 12)
(1)	This identifier is contained in the constant word %KWxy.i.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
%MFxy.i.38	NUMERATOR_1	-	READ_PARAM	Slave 1 read command
			WRITE_PARAM	Slave 1 write command
%MFxy.i.40	DENOMINATOR_1	-	READ_PARAM	Slave 1 read command
			WRITE_PARAM	Slave 1 write command
%MFxy.i.46	NUMERATOR_2	-	READ_PARAM	Slave 2 read command
			WRITE_PARAM	Slave 2 write command
%MFxy.i.48	DENOMINATOR_2	-	READ_PARAM	Slave 2 read command
			WRITE_PARAM	Slave 2 write command
%MFxy.i.54	NUMERATOR_3	-	READ_PARAM	Slave 3 read command
			WRITE_PARAM	Slave 3 write command
%MFxy.i.56	DENOMINATOR_3	-	READ_PARAM	Slave 3 read command
			WRITE_PARAM	Slave 3 write command
%MFxy.i.62	NUMERATOR_4	-	READ_PARAM	Slave 4 read command
			WRITE_PARAM	Slave 4 write command
%MFxy.i.64	DENOMINATOR_4	-	READ_PARAM	Slave 4 read command
			WRITE_PARAM	Slave 4 write command
%MFxy.i.70	NUMERATOR_5	-	READ_PARAM	Slave 5 read command
			WRITE_PARAM	Slave 5 write command
%MFxy.i.72	DENOMINATOR_5	-	READ_PARAM	Slave 5 read command
			WRITE_PARAM	Slave 5 write command
%MFxy.i.78	NUMERATOR_6	-	READ_PARAM	Slave 6 read command
			WRITE_PARAM	Slave 6 write command
%MFxy.i.80	DENOMINATOR_6	-	READ_PARAM	Slave 6 read command
			WRITE_PARAM	Slave 6 write command
%MWxy.i.26	**ACTION_CMD**	**1114**	**WRITE_CMD**	**Read command**
		2114		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of reading the Numerator variable.
%MFxy.i.24	RETURN_CMD_3	-	-	Result of reading the Denominator variable.
%MDxy.i.27	PARAM_CMD_1	-	-	Value to write: identifier of the axis of which the ratio is sent back.

Object	Name	Function code	Instruction type	Description
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write: ratio numerator.
%MFxy.i.33	PARAM_CMD_4	-	-	Value to write: ratio denominator.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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 Parameters: 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 Cam profile for several slave axes, but each slave's cam shape is rotated some amount with respect to another slave's cam.

Note: The MasterOffset may be modified while following is on, but be careful to avoid undesirable movement. Large changes should be blended in as a series of small increments to avoid rapid, immediate changes in the slave position.

Associated Functions

The associated functions of this parameter are:

- GetMasterOffset which sends a read request for the actual value of the master offset of the designated axis,
- SetMasterOffset which sends a write request for the actual 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
%MFxy.i.40	DENOMINATOR_1	-	READ_PARAM	Slave 1 read command
			WRITE_PARAM	Slave 1 write command
%MFxy.i.48	DENOMINATOR_2	-	READ_PARAM	Slave 2 read command
			WRITE_PARAM	Slave 2 write command
%MFxy.i.56	DENOMINATOR_3	-	READ_PARAM	Slave 3 read command
			WRITE_PARAM	Slave 3 write command
%MFxy.i.64	DENOMINATOR_4	-	READ_PARAM	Slave 4 read command
			WRITE_PARAM	Slave 4 write command
%MFxy.i.72	DENOMINATOR_5	-	READ_PARAM	Slave 5 read command
			WRITE_PARAM	Slave 5 write command
%MFxy.i.80	DENOMINATOR_6	-	READ_PARAM	Slave 6 read command
			WRITE_PARAM	Slave 6 write command
%MWxy.i.26	**ACTION_CMD**	**1532**	**WRITE_CMD**	**Read command**
		2532		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read: Position variable.
%MDx'y'.i.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write: Position.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

Legend	
xy	Slave group address (channel between 21 and 24)
x'y'	Slave axis address (channel between 1 and 12)
(1)	This identifier is contained in the constant word %KWxy.i.0

Slave Group Parameters: 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 MasterTrigger-Position parameter.

Object	Name	Function code	Instruction type	Description
%MFxy.i.42	TRIGGER_POSITION_1	-	READ_PARAM	Slave 1 read command
			WRITE_PARAM	Slave 1 write command
%MFxy.i.50	TRIGGER_POSITION_2	-	READ_PARAM	Slave 2 read command
			WRITE_PARAM	Slave 2 write command
%MFxy.i.58	TRIGGER_POSITION_3	-	READ_PARAM	Slave 3 read command
			WRITE_PARAM	Slave 3 write command
%MFxy.i.66	TRIGGER_POSITION_4	-	READ_PARAM	Slave 4 read command
			WRITE_PARAM	Slave 4 write command
%MFxy.i.74	TRIGGER_POSITION_5	-	READ_PARAM	Slave 5 read command
			WRITE_PARAM	Slave 5 write command
%MFxy.i.82	TRIGGER_POSITION_6	-	READ_PARAM	Slave 6 read command
			WRITE_PARAM	Slave 6 write command
%MWxy.i.26	**ACTION_CMD**	**1531**	**WRITE_CMD**	**Read command**
		2531		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read: Position variable.
%MDx'y'.i.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write: Position.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)
Legend				
xy	Slave group address (channel between 21 and 24)			
x'y'	Slave axis address (channel between 1 and 12)			
(1)	This identifier is contained in the constant word %KWxy.i.0			

Slave Group Parameters: 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
%MWxy.i.26	**ACTION_CMD**	**1527**	**WRITE_CMD**	**Read command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MDx'y'.i.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)
Legend				
xy	Slave group address (channel between 21 and 24)			
x'y'	Slave axis address (channel between 1 and 12)			
(1)	This identifier is contained in the constant word %KWxy.i.0			

Slave Group Parameters: 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 movement command (Move_Abs) to position the slave 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:

Command_pos = SlaveMethod (master_pos + master_offset) + Bias;

Where SlaveMethod is for:

- **Ratio**: multiply the master argument by the Ratio
- **Cam**: index into the Cam Profile given by the master argument.

Bias

The Bias is also composed of two parts:

Bias = additive_pos_command + abs_bias;

where:

- additive_pos_command is the position command being generated by any move commands directed to the slave 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.

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 AbsFollowerBias parameter.

Object	Name	Function code	Instruction type	Description
%MWxy.i.26	**ACTION_CMD**	**1526**	**WRITE_CMD**	**Read command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MDx'y'.i.27	PARAM_CMD_1	-	-	Value to write: Identifier (1) of the slave axis in the slave group.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)
Legend				
xy	Slave group address (channel between 21 and 24)			
x'y'	Slave axis address (channel between 1 and 12)			
(1)	This identifier is contained in the constant word %KWxy.i.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: %MWxy.i.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 a there is a master fault:

- Stop on master sequence (default value)
- Slave stop by latching command

Description:

Management bit: %MWxy.i.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 DECSTOP 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/s^2.

Object	Type	Symbol	Explanation	Comment
%MW xy.i.19	Word	ERROR_CMD		Error code
%MD xy.i.20	Double word	RETURN_CMD_1		not used
%MF xy.i.22	Floating point	RETURN_CMD_2		not used
%MF xy.i.24	Floating point	RETURN_CMD_3		not used
%MW xy.i.26	Word	ACTION_CMD	570	ADJUST_3022
%MD xy.i.27	Double word	PARAM_CMD_1	0	not used
%MD xy.i.29	Double word	PARAM_CMD_2	0	not used
%MF xy.i.31	Floating point	PARAM_CMD_3	Deceleration	
%MF xy.i.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 Lexiums ≥ 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.

7.4 Cam Profile Adjustment

Introduction

Aim of this sub-chapter

This sub-chapter describes adjustment functions for a cam profile.

What's in this Section?

This section contains the following topics:

Topic	Page
About Cam Profiles	263
Cam Profile Parameters: Length	264
Cam Profile Parameters: LookUpFollowerPosition Function	265
Cam Profile Parameters: Coord	266
Cam Profile Parameter: InterpType	268
Cam Profile Parameter: ProfileId	269
Description of the TRF_RECIPE Screen for Cam Profiles	270
TRF_RECIPE Instruction	272

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 Parameters: Length

Description

This parameter contains the number of point 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 at 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
%MWxy.i.26	**ACTION_CMD**	**534**	**WRITE_CMD**	**Read command**
%MDxy.i.20	RETURN_CMD_2	-	-	Result of the read.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

Cam Profile Parameters: 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
%MWxy.i.26	**ACTION_CMD**	**537**	**WRITE_CMD**	**Read command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of reading: Slave position.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write: Master position
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

Cam Profile Parameters: 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

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
%MWxy.i.26	**ACTION_CMD**	**532**	**WRITE_CMD**	**Read command**
		533		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read: Master position point from the cam profile table.
%MFxy.i.24	RETURN_CMD_3	-	-	Result of the read: Slave position point from the cam profile table.
%MDxy.i.27	PARAM_CMD_1	-	-	Value to write: Required point index.
%MFxy.i.31	PARAM_CMD_3			Value to write: Master position
%MFxy.i.33	PARAM_CMD_4			Value to write: Slave position
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

Cam Profile Parameter: InterpType

Description

This parameter is used to define the interpolation style used to generate the positions which do no 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

Do not modify a profile, do not call the functions Configure, SetCoord, or SetInterpType while following is on in a Slave Group which uses the cam profile.

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
%MWxy.i.26	**ACTION_CMD**	**530**	**WRITE_CMD**	**Read command**
		531		**Write command**
%MDxy.i.20	RETURN_CMD_1	-	-	Result of the read.
%MDxy.i.27	PARAM_CMD_1	-	-	Value to write: Interpolation Type 0: linear, 1: cubic.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

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
%MWxy.i.26	**ACTION_CMD**	**528**	**WRITE_CMD**	**Read command**
%MDxy.i.27	RETURN_CMD_1	-	-	Result of the read
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

Description of the TRF_RECIPE Screen for Cam Profiles

Introduction

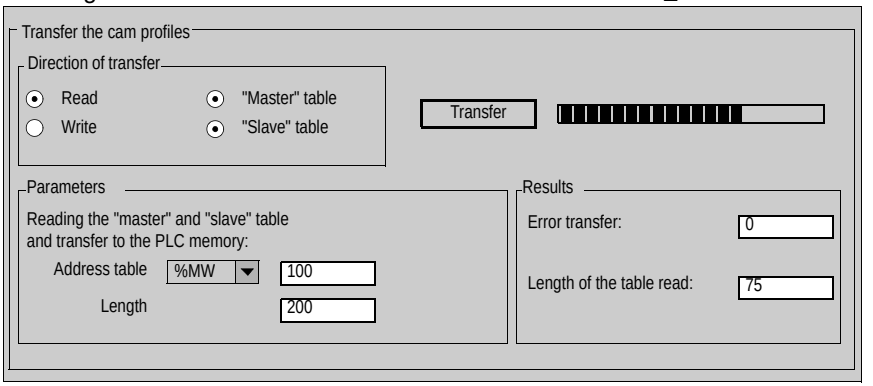
This screen, situated at the bottom of the adjustment screen of a cam profile is used to transfer adjustment parameters between the PLC at the variable speed controller

This function has to be in online mode.

Note: For more information on the common parts of the different adjustment screens, please see *Description of the Adjustment Screens of SERCOS® Module: General, p. 199*.

Illustration

The diagram below shows the screen associated with the TRF_RECIPE function.



Description

The table below shows the **Direction of transfer** of the TRF_RECIPE zone.

Element	Language word associated	Symbol/description
Read/Write	%MWxy.i.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	%MDxy.i.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	%MWxy.i.3	This field provides the error code (see *TRF_RECIPE Command Explicit Writing Faults, p. 348*) in the case of an invalid transfer.
Length of the table read	%MDxy.i.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 speed drive parameters with the "Real axis " function, (See *TRF_RECIPE, p. 212*).

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 %CHxy.i (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 %MWxy.i.10 (ACTION_TRF).

Example: TRF_RECIPE %CH104.25(100,200): "slave" table reading of cam profile 25 of module situated at position 4 of rack 1 and transfer to the PLC from address 200.

TRF_RECIPE Interface

The command to be carried out is defined in the word %MWxy.i.10 and the result of the command is available in the word %MWxy.i.3 to %MWxy.i.8.

Address	Type	Symbol	Meaning
%MWxy.i.10	**Word**	**ACTION_TRF**	**Action to be carried out**
%MWxy.i.3	Word	ERROR_TRF	Write error of the TRF_RECIPE command
%MDxy.i.4	Double Word	RETURN_TRF_1	Return 1 of the function
%MFxy.i.6	Floating point	RETURN_TRF_2	Return 2 of the function
%MFxy.i.8	Floating point	RETURN_TRF_3	Return 3 of the function
%MDxy.i.11	Double Word	PARAM_TRF_1	Parameter 1
%MDxy.i.13	Double Word	PARAM_TRF_2	Parameter 2
%MFxy.i.15	Floating point	PARAM_TRF_3	Parameter 3
%MFxy.i.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 (%MWxy.i.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 to the Module	PARAM_TRF_1 = 0. On MW
Writing the Module to 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(%MDxy.i.4)=length of the table read or written
	RETURN8TRF_2 rt RETURN_TRF_3 = 0

Exchange Check

The following 2 bits can be used to monitor the exchanges carried out with the help of TRF_RECIPE instruction:

Bit	Meaning
%MWxy.i.0:X3	Exchange in process.
%MWxy.i.1:X3	Incorrect exchange.

7.5 Sercos® ring parameters

Introduction

Aim of this sub-chapter

This sub-chapter introduces the adjustment screen of the SERCOS® ring and describes the associated adjustment parameters.

What's in this Section?

This section contains the following topics:

Topic	Page
Description of the Adjustment Screen of a SERCOS® Module: SERCOS® ring	275
Adjustment graph for optical power	276
Sercos® ring parameters: OpticalPower	277

Description of the Adjustment Screen of a SERCOS® Module: SERCOS® ring

Introduction

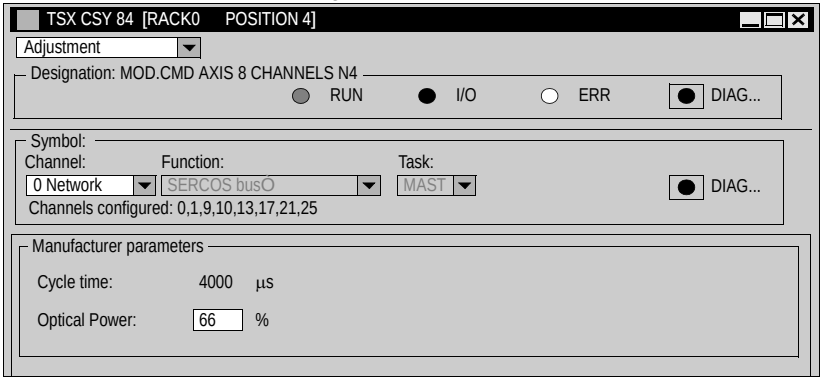
This screen is used to directly modify the value of the optic power in offline or online mode.

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 Adjustment Screens of SERCOS® Module: General, p. 199*.

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

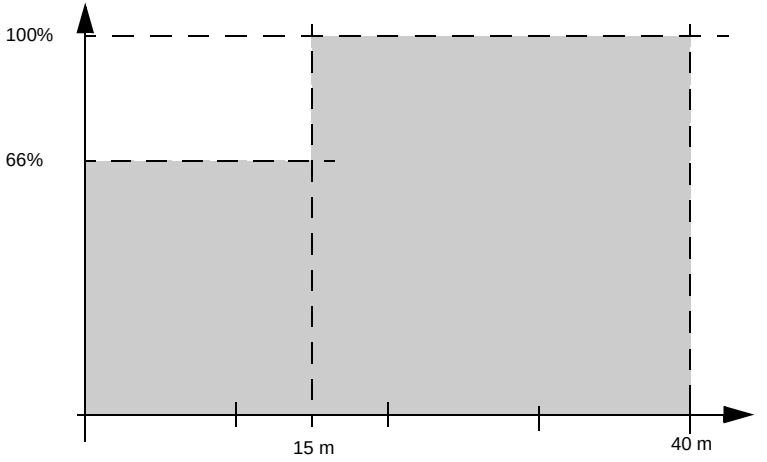
The table below shows the **Manufacturer parameters** of the adjustment zone.

Identification	Associated parameter	Bit/ word language associated	Symbol/Description
Cycle time	-	-	Value of the current ring cycle time.
Optical power	-	%MWxy.i.37	Adjusting the Optical Power

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 parameters: 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
%MWxy.i.26	**ACTION_CMD**	**1547**	**WRITE_CMD**	**Read command**
		2547		**Write command**
%MFxy.i.22	RETURN_CMD_2	-	-	Result of the read.
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

7.6 SERCOS® IDN Read/Write Functions

At a glance

Heading of this sub-chapter

This section describes the read/write functions for SERCOS® IDs and lists the standard IDs.

What's in this Section?

This section contains the following topics:

Topic	Page
Drive Parameters: IDN_S and IDN_P	279
Drive Parameters: IDN_US and IDN_UP	282
Standard SERCOS Identification Numbers (IDN)	285
IDNs for customized telegrams	288

Drive Parameters: IDN_S and IDN_P

Description

These parameters enable you to directly control the drives by the application (independently of the TSX CSY 84 / 164 modules).

They can be divided into two categories:

- IDN_S called "standard" (SERCOS® standard), See (*Standard SERCOS Identification Numbers (IDN), p. 285*),
- IDN_P called "owners" (specific to drive constructors, see information on drives).

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 command axis module.
- SetIDN_S and SetIDN_P which send a write request of 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® drive. 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 the16-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 IDN SERCOS® parameters are associated with unit and scale factors. 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
%MWxy.i.26	**ACTION_CMD**	**1556**	**WRITE_CMD**	**Read command IDN_S**
		2556		**Write command IDN_S**
%MWxy.i.26		**1557**		**Read command IDN_P**
		2557		**Write command IDN_P**
%MDxy.i.20	RETURN_CMD_1	-	-	Result of the read.
%MDxy.i.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 drive manual to find out what parameter sets are supported.)
%MDxy.i.29	PARAM_CMD_2	-	-	Value to write to the drive
%MFxy.i.31	PARAM_CMD_3	-	-	Value to write: AddressSercos. This parameter is required if you are using the GetIDN function on the BusSercos axisID 999.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)

Drive Parameters: IDN_US and IDN_UP

Description

These parameters are associated with units and scale factors.

They enable you to directly control the drives by the application (independently of the TSX CSY 84 / 164 modules).

They can be divided into two categories:

- IDN_US called "Standard Users" (SERCOS® standard), See (*Standard SERCOS Identification Numbers (IDN), p. 285*),
- IDN_UP called "User owners" (specific to drive 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 drive 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 drive.

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 drive.

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
%MWxy.i.26	ACTION_CMD	1558	WRITE_CMD	Read request IDN_US
		2558	WRITE_CMD	Write command IDN_US
%MWxy.i.26	ACTION_CMD	1559	WRITE_CMD	Read request IDN_UP
		2559	WRITE_CMD	Write command IDN_UP
%MFxy.i.22	RETURN_CMD_2	-	-	Read report: Value (1) of the floating point parameter.
%MDxy.i.27	PARAM_CMD_1	-	-	IDNSercos. Format type of 1 to 4095. However, use of bits 12 to 14 is allowed. These bits indicate which parameter sets are being selected. (See your drive manual to find out what parameter sets are supported.)
%MFxy.i.31	PARAM_CMD_3	-	-	AddressSercos. This parameter is required if you are using the GetIDN function on the BusSercos axisID 999.
%MFxy.i.33	PARAM_CMD_4	-	-	Contents of the variable: Value (1) of the floating point parameter.
%MWxy.i.19	ERROR_CMD	-	-	Error report. See (*WRITE_CMD Command: Programming Errors, p. 340*)
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

8

At a Glance

Subject of this Chapter

This chapter describes the debug functions of a SERCOS® channel.
All procedures for the TSX CSY 84 are the same for the TSX CSY 164 module.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Description of the debug screen: General	290
Debug Screen User Interface	293
Debug: Enable	294
Debug: Diagnostics	295
Debug: Movement	299
Debug: Sending commands (Auto mode)	301
Debug: Manual command (Manual mode)	302
Debug: Reference	303
Debug: Following	304
Debug: Drive	305
Debug: Position	306
Diagnostics of the Module	307
Channel Diagnostics	308

Description of the debug screen: General

Introduction

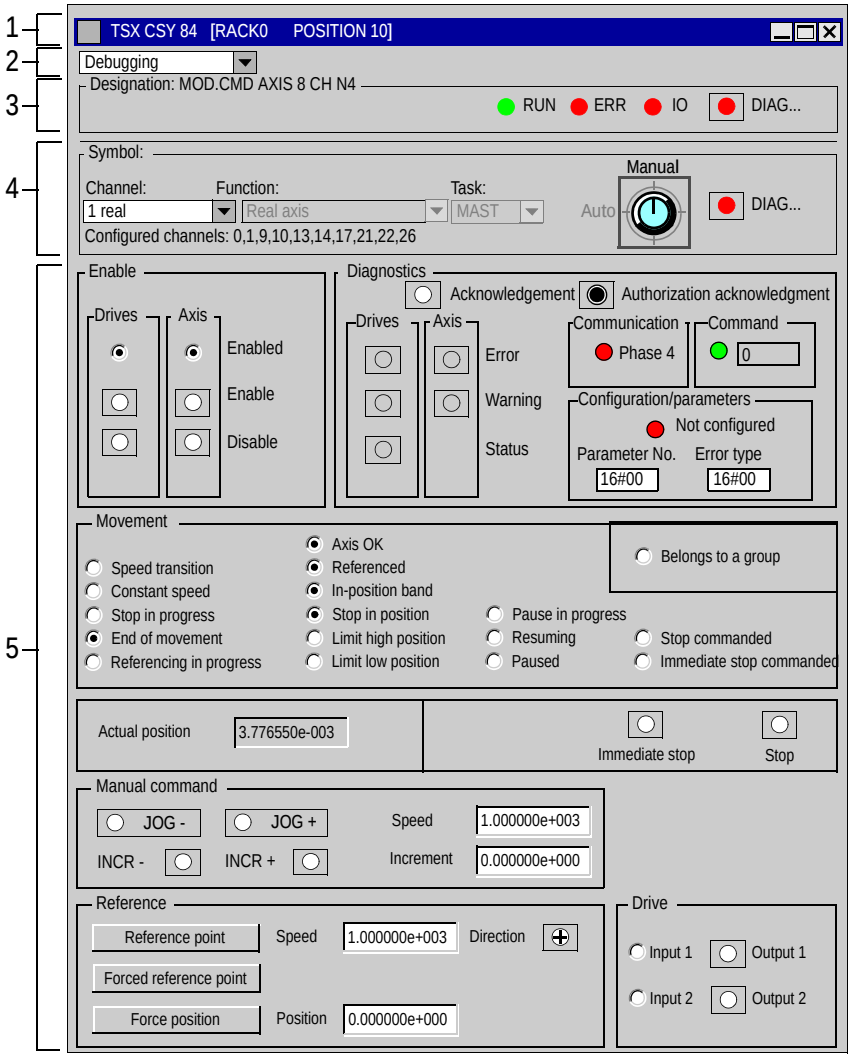
The different debug screens can be used, for all types of channels, to display the state of the inputs and outputs of a module channel as well as any faults.
It also enables you to command language objects (set to 1 or 0, forcing/unforcing a bit, etc.).
These parameters are accessible in online mode.

It also gives access to the configuration and adjustment screens.

Note: For an introduction to the processes of configuring, adjusting and debugging an application, consult the section on Common application 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 adjustment screen and their functions.

Address	Identification	Function
1	Title bar	Recalls the reference of the module selected and its physical position in the rack.
2	Command zone	Indicates the screen in progress (**Debug**) and allows access or return to other screens: ● **Configuration**, only accessible in offline mode ● **Adjustment**.
3	Module zone	Displays the designation of the module selected. Displays indicator lamp status **RUN**, **ERR**, **I/O** Creates direct access to the module diagnostics when it is faulty (indicated by the indicator lamp built into the diagnostics access button DIAG, which turns red).
4	Channel zone	Allows you to choose the channel to be adjusted and to modify (if several to choose from) the function associated with the channel, Indicates the task in which the channel is configured, Allows you to access manual mode (real and imaginary axes), Provides direct access to the diagnostics of the channel when it is faulty
5	Debug zone	This zone displays the state of the inputs, outputs and different parameters in progress. It can also be used to control axis movements in manual mode.

Debug Screen User Interface

Introduction

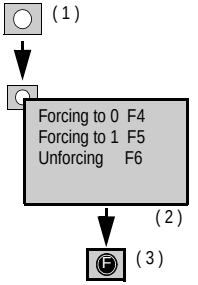
The debug screen provides access to the state of the input and output bits through LEDs.
It also enables you to command language objects (%Q) using command buttons

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 movement commands.

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: **Force to 0**, **Force 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

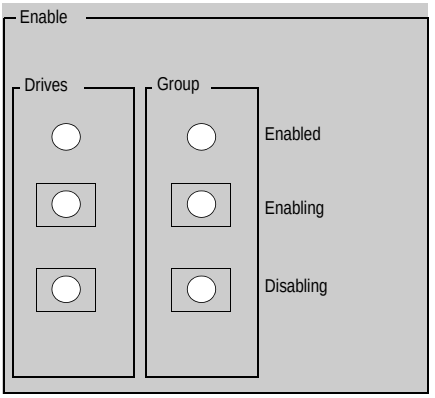
Introduction

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 *DRIVE_ENABLED bit (%Ixy.i.10), p. 121*.	
Button **Enable**	%Qxy.i.10	Enabling the axis at drive level.
Button **Disable**	%Qxy.i.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 *AXIS_IN_COMMAND bit (%Ixy.i.18), p. 132*.
Button **Enable**	%Qxy.i.2	Enabling the axis at axis level.
Button **Disable**	%Qxy.i.18	Disabling the axis at axis level.

Debug: Diagnostics

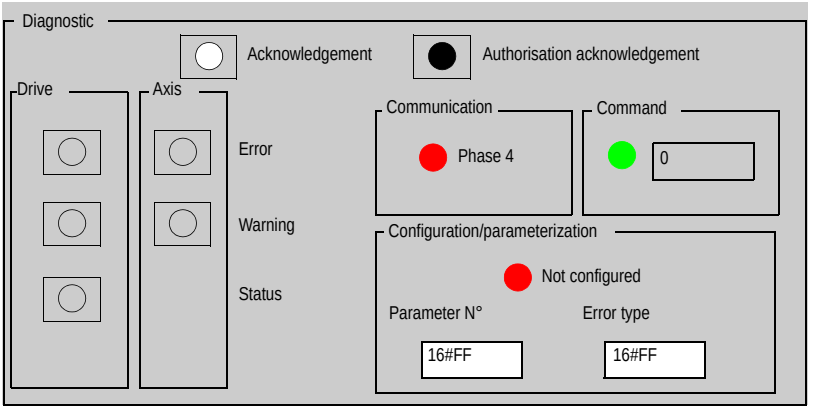
Introduction

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 Diagnostics field of the debug screen.



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**
Button **Acknowledgement**	%Qxy.i.15	Fault acknowledgement
Button **Authorisation acknowledgement**	%Qxy.i.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
LED **Error** Button **Error**	See *DRIVE_FAULT bit (%Ixy.i.13), p. 124*. (1)
LED **Warning** Button **Warning**	See *DRIVE_WARNING bit (%Ixy.i.12), p. 123*. (1)
LED **Status** Button **Status**	See *DRIVE_DIAG bit (%Ixy.i.11), p. 122*. (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
LED **Error** Button **Error**	See *AXIS_SUMMARY_FAULT bit (%Ixy.i.15), p. 126*. (1), **Note:** The drive fault data is not to be taken into account if there is a SERCOS® communication error.
LED **Warning** Button **Warning**	See *AXIS_WARNING bit (%Ixy.i.23), p. 133*. (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
LED **Phase 4**	%Ixy.i.16	See*AXIS_COMM_OK bit (%Ixy.i.16), p. 127*.

The table below describes the different **Command** zone elements of the Diagnostics field.

Element	Associated language object	Description
LED	%MWxy.i.1:X2	Faulty explicit command (Action_CMD).
Field	%MWxy.i.19	Possible values: 0 : Ok parameters Otherwise an error code Seewrite_cmd:programming errors command chapter (See *WRITE_CMD Command: Programming Errors, p. 340*)

The table below describes the different **Configuration/ Parameterization** zone elements of the Diagnostics field.

Element	Associated language object	Description
LED **Not configured**	%Ixy.i.32	The channel is configured.
Field **N° parameter**	%IW2xy.i.:X0 to X7	Indicates the faulty parameter. See faulty registers chapter (See *Faulty Registers, p. 334*)
Field **Error type**	%IW2xy.i.:X8 to X15	Indicates the type of error. See (See *List of Configuration and Adjustment Errors, p. 331*)

Debug: Movement

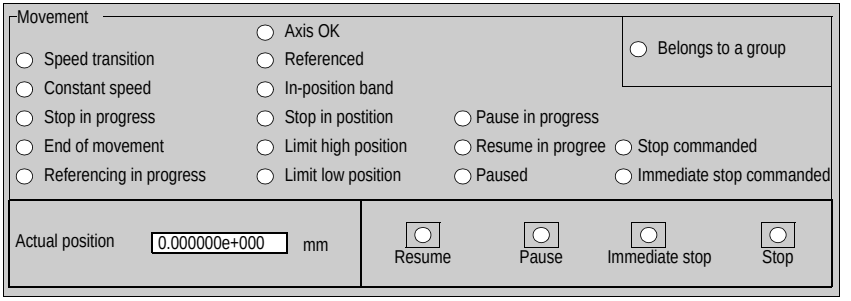
Introduction

This field is used to view module status bits and to control the movement of the selected axis.

Note: It is concerned with real or imaginary axes as well as with groups of co-ordinated axes.

Illustration

The illustration below shows the Movement field of the debug screen.



Description The table below describes the different elements of the Movement field.

Element	Associated language object	Description
LED **Speed transition**	See *Bit RAMPING (%Ixy.i.0), p. 111*.	
LED **Constant speed**	See *STEADY Bit (%Ixy.i.1), p. 112*.	
LED **Stop in progress**	See *STOPPING Bit (%Ixy.i.2), p. 113*.	
LED **End of movement**	See *PROFILE_END Bit (%Ixy.i.3), p. 114*.	
LED **Referencing in progress**	See *AXIS_HOMING bit (%Ixy.i.5), p. 116*.	
LED **Axe OK** (unitary axis only)	See *AXIS_READY bit (%Ixy.i.31), p. 141*.	
LED **Referenced (unitary axis only)**	See *AXIS_HOMED bit (%Ixy.i.6), p. 117*.	
LED **In-position band**	See *IN_POSITION Bit (%Ixy.i.4), p. 115*.	
LED **Stop in position**	See *AXIS_AT_TARGET bit (%Ixy.i.20), p. 129*.	
LED **High limit position**	See *AXIS_POS_LIMIT bit (%Ixy.i.21), p. 130*.	
LED **Low limit position**	See *AXIS_NEG_LIMIT bit (%Ixy.i.22), p. 131*.	
LED **Pause in progress**	See *HOLDING bit (%Ixy.i.8), p. 119*.	
LED **Resume in progress**	See *RESUMING bit (%Ixy.i.9), p. 120*.	
LED **Pause done**	See *AXIS_HOLD bit (%Ixy.i.28), p. 138*.	
LED **Stop commanded**	See *AXIS_HALT bit (%Ixy.i.29), p. 139*.	
LED **Immediate stop commanded**	See *AXIS_FASTSTOP bit (%Ixy.i.30), p. 140*.	
LED **Belongs to a group (unitary axis only)**	See *AXIS_IS_LINKED bit (%Ixy.i.17), p. 128*.	
Button **Resume**	%Qxy.i.12	Resumes control after a stop.
Button **Pause**	%Qxy.i.28	Temporary axis stop.
LED **Immediate stop**	%Qxy.i.30	Urgent stop command.
LED **Stop**	%Qxy.i.29	Stop axis.
Field **Current position (unitary axis only)**	%IFxy.i.0	Value of current position.

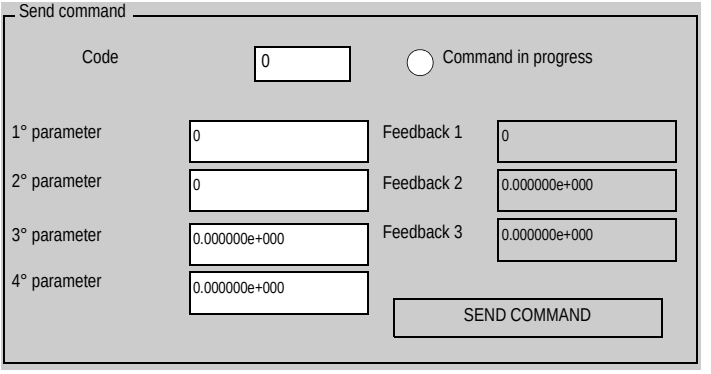
Debug: Sending commands (Auto mode)

Introduction

This field, which is specific to Auto mode, is used to launch a movement command using explicit exchanges.

Illustration

The illustration below shows the Send command field of the debug screen.



Description

The table below describes the different elements of the Send command field.

Element	Associated language object	Description
Code field	%MWxy.i.26	Function code
Command in progress LED	%MWxy.i.0:X1	(See *Exchange check, p. 369*)
Parameter fields	%MWxy.i.27 to 33 (1) %MWxy.i.27 to 44 (2)	Function parameters
Return N fields	%MDxy.i.20 %MFxy.i.22 %MFxy.i.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)

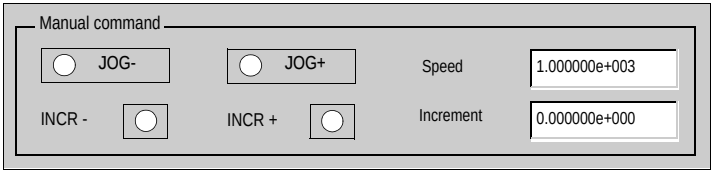
Introduction

This field, which is specific to manual mode, is used to launch a continuous (JOG) or incremental (INC) movement command.

Note: It is concerned with real and imaginary axes.

Illustration

The illustration below shows the Manual Command field of the debug screen.



Description

The table below describes the different elements of the Manual command field.

Element	Associated language object	Description
JOG+	%Qxy.i.4	See *Manual mode, p. 151*. The button associated with this command is a push button.
JOG-	%Qxy.i.5	See *Manual mode, p. 151*. The button associated with this command is a push button.
INC+	%Qxy.i.13	See *Manual mode, p. 151*.
INC-	%Qxy.i.14	See *Manual mode, p. 151*.
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	%QDxy.i	desired increment value.

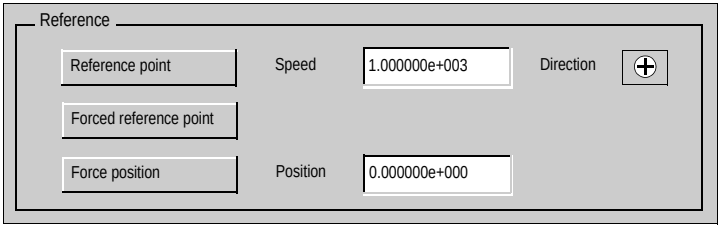
Debug: Reference

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.



Description

The table below describes the different elements of the Reference field.

Element	Description
Reference point	Sets a homing reference point (see *Home Function, p. 154*).
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 *ForcedHomed Function, p. 157*).
Force position	Modifies the homing point without referencing axis to that point (see *SetPosition Function, p. 159*).
Position	Home position forcing value.

Debug: Following

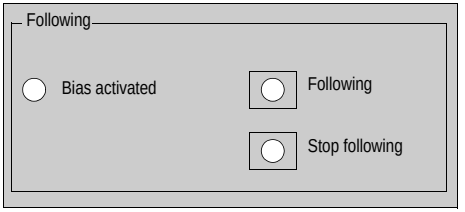
Introduction

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**	%Ixy.i.24	Offset added to the command position
Button **Follow**	%Qxy.i.11	*Control Bits Modifiable on Change of State, p. 149*
Button **Stop follow**	%Qxy.i.27	*MovementControl Bits Modifiable on Edge, p. 146*

Debug: Drive

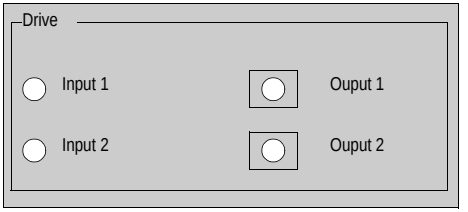
Introduction

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 *DRIVE_REALTIME_BIT1 bit (%Ixy.i.26), p. 136*
LED **Input 2**	%I27	See *DRIVE_REALTIME_BIT2 bit (%Ixy.i.27), p. 137*
Button **Output 1**	%Q6	Drive command
Button **Output2**	%Q7	

Debug: Position

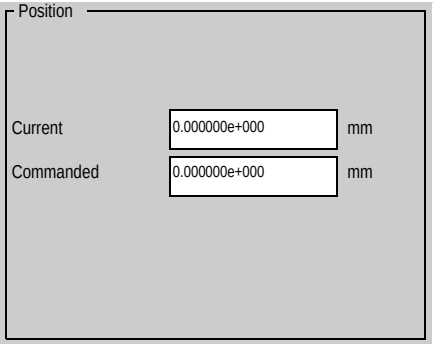
Introduction

This field is used to view the position of an external axis from the application.

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 application.

Diagnostics of the Module

Introduction

The Diagnostic function module displays the current faults, if any exist, classified according to their category:

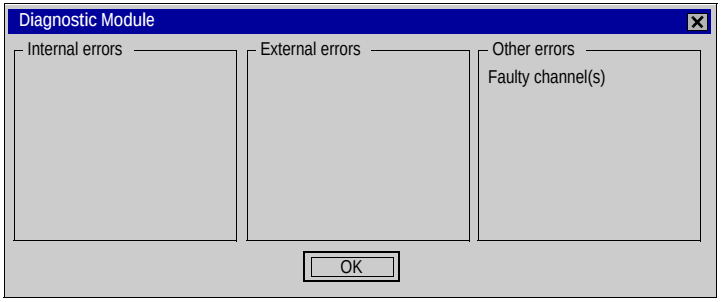
- internal errors (broken modules, self-test in progress),
- external errors (terminal block fault),
- other errors (configuration error, module missing or no power, faulty channel(s) (details in channel diagnostics).

A faulty module is shown by certain LEDs turning red, e.g.:

- in the configuration level rack editor:
  - the LED of the module position,
- in the configuration level module editor:
  - the **Err** and **I/O** LEDs according to the type of fault,
  - the **Diag** LED.

Procedure

The table below introduces the procedure for accessing the Diagnostic module screen.

Step	Action
1	Access the debug module screen.
2	Click on the **Diag** button located in the zone module. **Result:** The list of module faults appears.  **Note:** If there is a configuration fault, a major breakdown or a missing module, access to the diagnostic module screen is not possible. The following message will then appear on the screen: `The module is missing or different to the one configured in this position`.

Channel Diagnostics

Introduction

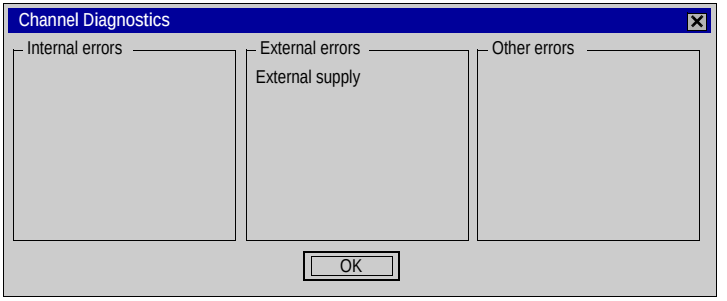
The Diagnostic function channel displays the current faults, if any exist, classified according to their category:

- internal faults (broken channels),
- external faults (faulty link or sensor supply),
- other faults (terminal block fault, 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.  **Note:** Diagnostic information about the channel can also be accessed by program (instruction READ_STS).

Diagnostics and Maintenance

9

At a glance

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's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
9.1	SERCOS® Phase Diagnostics Introduction	310
9.2	Reading Actual Data for the Ring Introduction	314
9.3	Axis Identification	318
9.4	Status and default information	319
9.5	Faults and Warnings	328
9.6	Performances of the TSX CSY 84	354

9.1 SERCOS® Phase Diagnostics Introduction

At a glance

Aim of this sub-chapter

This section describes the diagnostics parameter functions for the SERCOS® phase.

What's in this Section?

This section contains the following topics:

Topic	Page
About SERCOS® Phases Standard	311
GetActualPhase Function	312
GetCommandedPhase Function	313

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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**550**	**Command**
%MDxy.i.20	RETURN_CMD_1	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

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, p. 368*).

Associated Language Objects

Exchange via WRITE_CMD: The table below presents the different language objects linked to these functions.

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**1545**	**Read command**
		2545	**Write command**
%MDxy.i.20	RETURN_CMD_1	-	Result of the read command
%MDxy.i.27	PARAM_CMD_1	-	Value to write)
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

9.2 Reading Actual Data for the Ring Introduction

At a glance

Aim of this sub-chapter

This section describes the functions for reading actual data for the ring (SERCOS® address, number of drives, etc).

What's in this Section?

This section contains the following topics:

Topic	Page
GetSercosAddress Function	315
GetNumberOfDrivesInRing function	316
IsLoopUp function	317

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 %KWxy.i.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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**549**	**Command**
%MDxy.i.20	RETURN_CMD_1	-	Result of the command (1)
%MDxy.i.27	PARAM_CMD_1	-	Value to write (2)
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.
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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**548**	**Command**
%MDxy.i.20	RETURN_CMD_1	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**543**	**Command**
%MDxy.i.20	RETURN_CMD_1	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

9.3 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 application,
- 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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**541**	**Command**
%MDxy.i.20	PARAM_CMD_1	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

9.4 Status and default information

At a glance

Subject of this chapter

This section describes the read functions of the default and status information.

What's in this Section?

This section contains the following topics:

Topic	Page
Implicit Exchange Objects	320
Faults Accessible via the GetMotionFault Command	321
GetMotionFault Function	323
Faults Accessible via the GetMotionWarning Command	324
GetMotionWarning Function	327

Implicit Exchange Objects

Faulty Module	The %Ixy.MOD.ERR bit at status 1 indicates that the module situated in position xy is faulty. The causes of the fault are given in the %MWxy.MOD.2 internal word.
Faulty channel	The bit %Ixy.i.ERR at status 1 indicates that i channel of the module situated in position xy is faulty. The causes of the fault are given in the %MWxy.i.2 internal word.

Faults Accessible via the GetMotionFault Command

Description

The MotionFault data indicates the type of error that caused AXIS_SUMMARY_FAULT 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	Conflicting control with configuration drive 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 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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**5510**	**Command**
%MDxy.i.20	RETURN_CMD_1	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

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.
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 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, p. 368*).

Associated Language Objects

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

Object	Name	Function code	Description
%MWxy.i.26	**ACTION_CMD**	**5511**	**Command**
%MDxy.i.20	RETURN_CMD_1	-	Result of the command
%MWxy.i.19	ERROR_CMD	-	Error report. See *WRITE_CMD Command: Programming Errors, p. 340*.

9.5 Faults and Warnings

At a Glance

Title of this Section

This section shows the list of faults and warnings that can be accessed by explicit reading.

What's in this Section?

This section contains the following topics:

Topic	Page
Module faults which can be accessed by explicit reading	329
Channel faults accessible via explicit reading	330
List of Configuration and Adjustment Errors	331
Faulty Registers	334
List of Error Codes for a WRITE_CMD Command	339
WRITE_CMD Command: Programming Errors	340
WRITE_CMD Command: Communication errors	343
WRITE_CMD Command: System errors	344
WRITE_CMD Command: System warnings	347
TRF_RECIPE Command Explicit Writing Faults	348
Operating modes	349
Logic diagrams of faults	350
Drive faults	352

Module faults which can be accessed by explicit reading

Introduction

Word %MWxy.MOD.2 indicates any fault on the module in position xy.

List of faults

Bits %MWxy.MOD.2:X0 to %MWxy.MOD.2:X15 are used to diagnose the module faults:

Bit	Meaning
0	Internal fault: module inoperative
1	Functional fault: external fault, communication fault or application fault (see the channel status word %MWxy.i.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 accessible via explicit reading

Introduction

Word %MWxy.i.2 indicates any fault on channel i of the module in position xy.

List of faults

Bits %MWxy.i.2:X0 to %MWxy.i.2:X15 are used to diagnose the channel faults:

Bit	Meaning
0	External fault 0: drive 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	Application fault: 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 %IWxy.i.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	Description
1	Attempt to assign value out of range
2	Attempt to assign incompatible units
3	Unit not supported or unknown
4	Drive fault while uploading
5	Drive fault while downloading
6	Unexpected empty pointer to object
7	Failed to set units in drive
8	Units not set
9	String too large for MovementString
A	Invalid index in a group
B	Incorrect value in command
C	Incorrect value in a sequence
D	Invalid token in input
E	Invalid feedback channel for command
F	Invalid feedback peripheral for a command
10	Invalid feedback clocking frequency
11	Invalid feedback supply
12	Invalid feedback resolution
13	Invalid holding register address
14	Holding register database not configured
15	Holding register database empty
16	Holding register block too big
17	Holding register block does not correspond to the database
18	Cannot grant access to holding register block
19	Cannot release access to holding register block
1A	File open failed
1B	File write failed
1C	File read failed
1D	File closure failed
1E	File search failed
1F	Malformed input error
20	Clear fault function failed

Error code	Description
21	Missing tag in tags.cfg
22	No axis object available
23	Too many axes in configuration
24	Duplicate axes in configuration
25	Missing or invalid axis
26	Axis object or configuration file cannot be found
27	Value has different number of co-ordinates from axis
28	Movement axis is not actived
29	A movement fault has occurred on the movement controller
2A	Drive is not activated
2B	Command timed out
2C	Only one SERCOS® bus may be configured
2D	Axis rename failed
2E	Cannot perform command in current configuration
2F	Object is wrong type
30	Drive must be deactivated to perform command
31	Drive must be activated to perform command
32	Command not allowed at this time
33	Command cannot be executed due to drive 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
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_MAITRE
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
2F	REG_MODE_SUIVEUR_S6

Error code	Register
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 %MWxy.i.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 *WRITE_CMD Command: Programming Errors, p. 340*),
- the communication errors (see *WRITE_CMD Command: Communication errors, p. 343*),
- the system errors (see *WRITE_CMD Command: System errors, p. 344*).

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	Drive fault during download
5	Drive fault during upload
6	Unexpected empty pointer to object
7	Failed to set units in drive
8	Units not set
9	String too large for MovementString
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
33	Missing tag in tags.cfg
34	No axis object available
35	Too many axes in configuration

Error code	Description
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 move fault has occurred on the motion controller
42	Drive is 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	Drive must be deactivated to perform command
49	Drive must be activated to perform command
50	Command not allowed at this time
51	Command cannot be completed due to drive fault
60	Manual mode refused for axis linked to a coordinated or slave set
61	Auto control refused for an axis in manual mode
62	Slave group that has cam profile slaves
63	The deceleration for a slave or a master is above the maximum configured deceleration
64	Refusal of the TRF_ RECIPE 26200: the instance is already active
65	Refusal of the TRF_ RECIPE 26200: the alarm threshold is above the default threshold
66	Refusal of the TRF_ RECIPE 26200: one of the thresholds is negative
67	Refusal of the TRF_ RECIPE 26200: one of the denominators is zero
68	Refusal of the TRF_ RECIPE 16200: the instance is not active

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
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

Error code	Description
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
7777	A call was made to the Sample_User subroutine

TRF_RECIPE Command Explicit Writing Faults

Introduction

The %MWxy.i.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 TRF_RECIPE command specific faults.

List of Errors Specific to the TRF_RECIPE Command

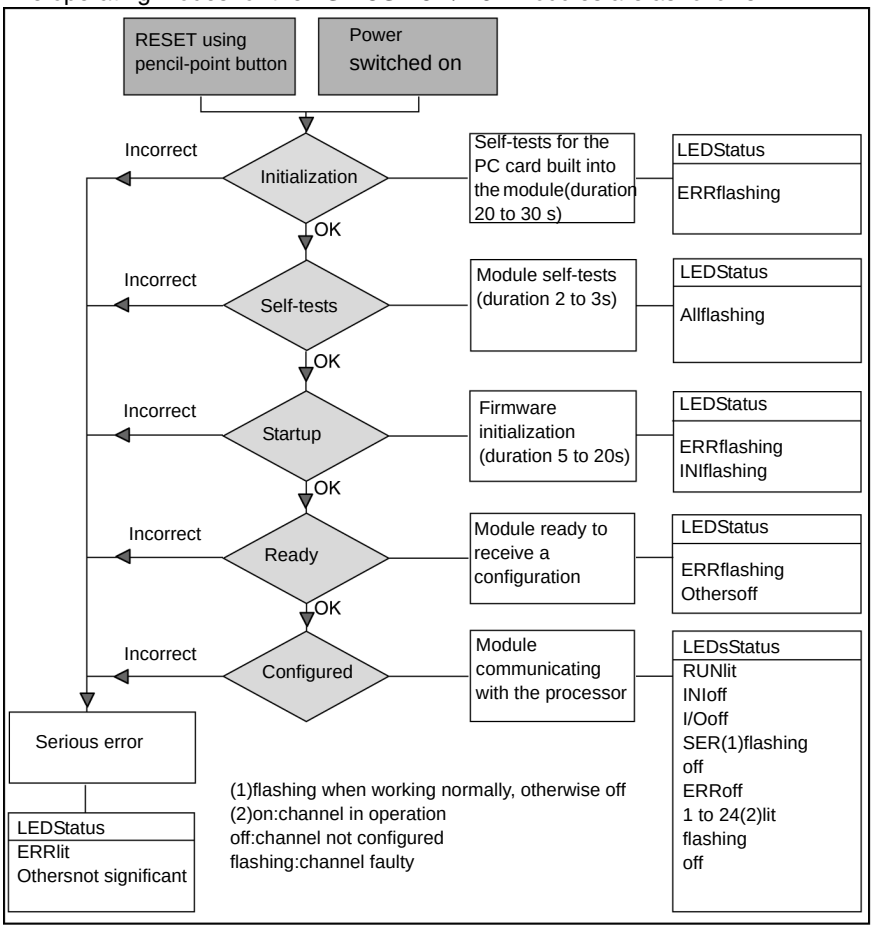
The following table gives the error code, the name and the description of the errors specific to the TRF_RECIPE command:

Error code	Description
6	The CamProfile is configured to be fixed, not variable. A CamProfile table cannot be read or written.
19	The transfer requested exceeds the capacity of the %MW or %KW of the PLC.
22	The table associated with the TRF_RECIPE is not large enough for the data to be exchanged.

Operating modes

Overview of the Operating Modes

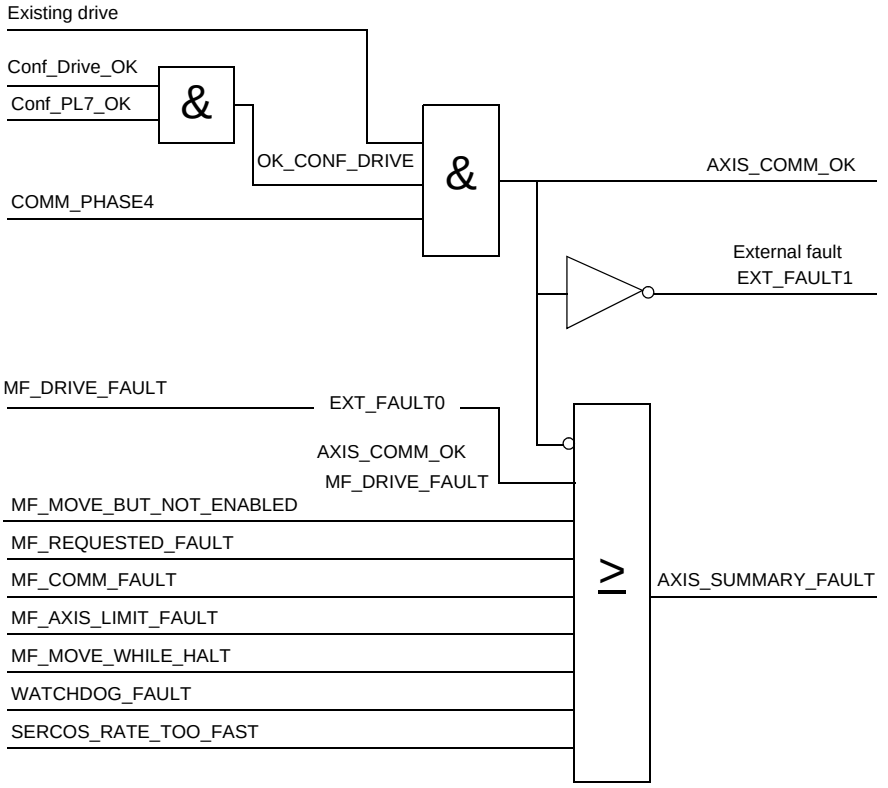
The operating modes for the TSX CSY 84 / 164 modules are as follows:



Logic diagrams of faults

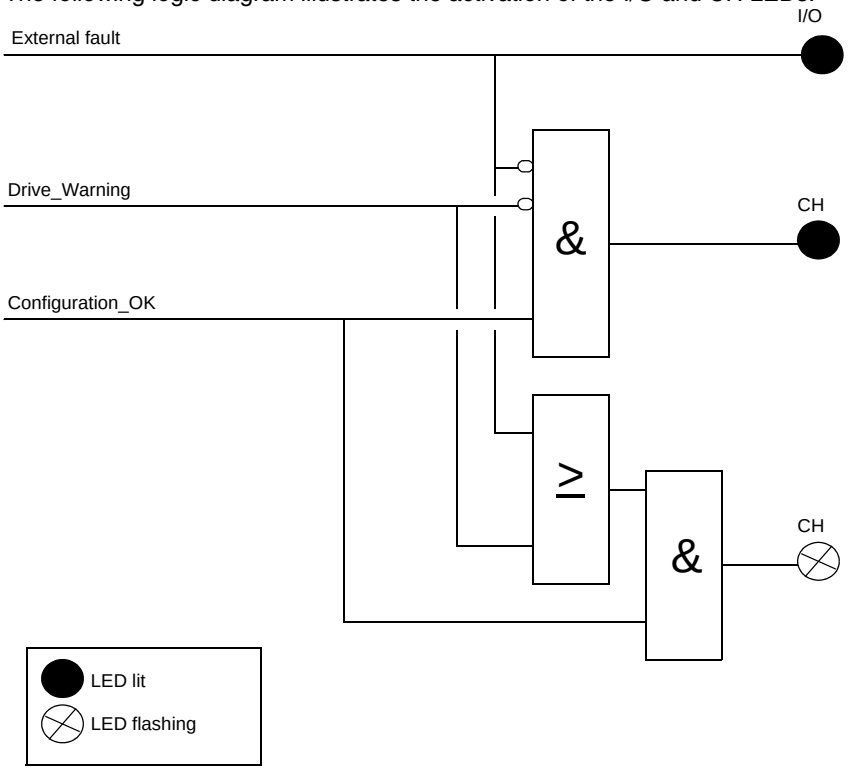
Fault Generation

The following logic diagram illustrates the generation of AXIS_SUMMARY_FAULT and external faults:



Activation of LEDs

The following logic diagram illustrates the activation of the I/O and CH LEDs:



Drive faults

Introduction

Certain functions can be used upload the default information to the application.

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, p. 278*).

"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)

9.6 Performances of the TSX CSY 84

Performances of the TSX CSY 84

At a Glance

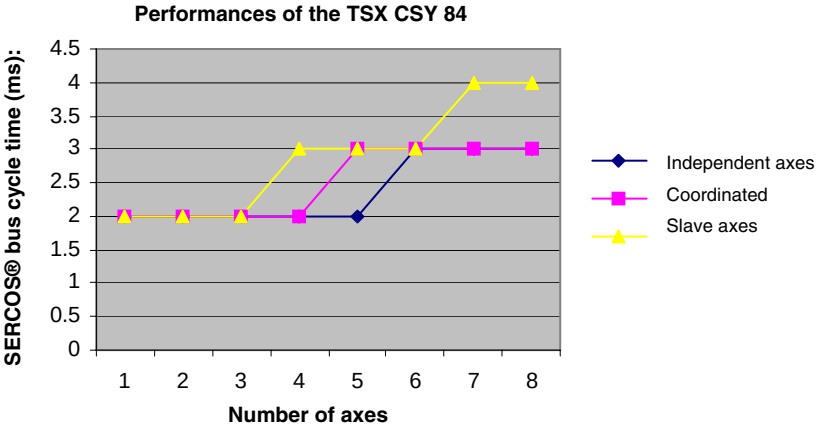
The performances of the module are:

- 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:



Test conditions:

Warning: 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 *)
```

Interface language

10

At a Glance

Subject of this Chapter

This chapter describes the services available through the PL7 interface and supplies the list of variables for the TSX CSY 84 / 164 modules.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
10.1	PL7 Interface	361
10.2	List of Variables	374

10.1 PL7 Interface

Introduction

Aim of this sub-chapter

This chapter describes the services, which may be accessed via the PL7 interface: READ_PARAM, WRITE_PARAM, SAVE_PARAM, RESTORE_PARAM, WRITE_CMD, TRF_RECIPE, etc.

What's in this Section?

This section contains the following topics:

Topic	Page
Interface Language	362
Parameter Management	364
WRITE_PARAM and READ_PARAM	366
SAVE_PARAM and RESTORE_PARAM	367
WRITE_CMD	368
WRITE_CMD Examples	370
READ_STS	372
TRF_RECIPE and MOD_PARAM	373

Interface Language

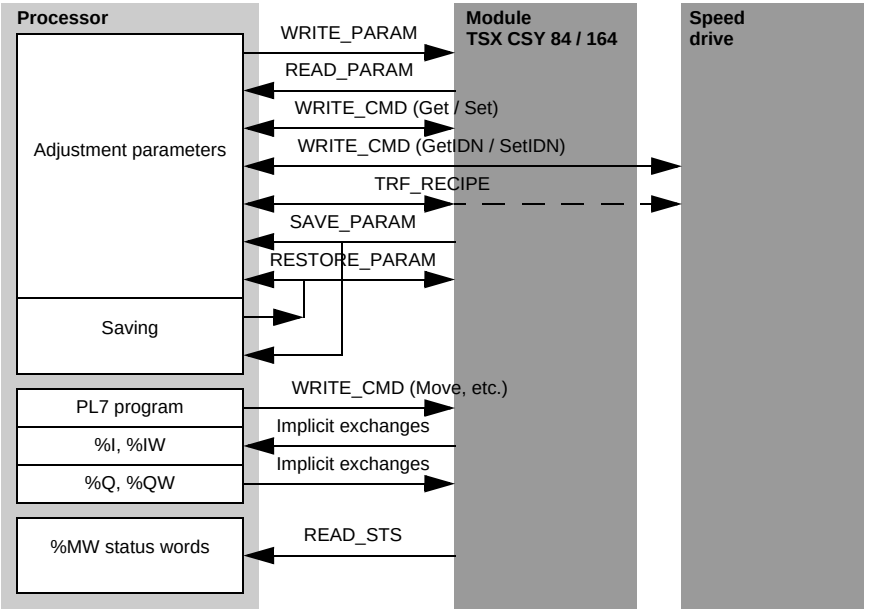
Module Channels

The TSX CSY 84 / 164 modules are made up of up to 32 channels which support the following functions:

Channels	Function supported
%CHxy.0	SERCOS® function
%CHxy.1 to %CHxy.8	Real axes
%CHxy.9 to %CHxy.12	Imaginary axes
%CHxy.13 to %CHxy.16	Remote axes
%CHxy.17 to %CHxy.20	Coordinated axis group
%CHxy.21 to %CHxy.24	Slave axis group
%CHxy.25 to %CHxy.31	Cam Profiles

Overview of Exchanges

The exchanges between the processor, the axis command module and the variable speed controllers are performed as follows:



Parameter Management

General

The parameters of an axis command module and speed drives are managed separately via PL7 and UniLink. For example, it is possible to define the different values for limits in an axis command module and speed drives.

The parameters of an axis command module may be divided into 2 categories:
- Parameters used only by the axis command module,
- parameters used both by the axis command module and the speed drives.

Note: Drives that operate with a linear type unit are not supported by the TSX CSY 84 / 164 modules.

Axis Command Module Parameters

Type "C" (Controller) parameters are exchanged whilst floating, in the axis command module units. These parameters are related to the path calculation or cam profiles. They are configured in PL7 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.

Speed Drive Parameters

When the application modifies a parameter in the axis command module, the corresponding parameter in the speed drives 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 speed drive 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 Command Module and the Drives

The axis command module does not update the speed drive 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 command module and in full with the speed drives (e.g. acceleration, speed and position values).

If the application requires the value of the axis command module parameters to be synchronized with the speed driver 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 command module and the drives.

Units are configured using PL7. Axis command module units are converted into speed drive units automatically by the TSX CSY 84 / 164 modules 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 (application) and the axis command module.

READ_PARAM: explicit reading of the parameters in the axis command module and updating of the adjustment words %MW/D/Fxy.ir.

WRITE_PARAM: explicit writing of parameters in the axis command module. This instruction allows the modification by the program of defined adjustment values in configuration.

Syntax of the instruction READ_PARAM

READ_PARAM %CHxy.i: reading the adjustment parameters of channel i, of the module situated at the xy address (rack number, position in the rack).

For example: READ_PARAM %CH3.1 (reading the parameters of channel 1, of the module situated in position 3 of rack 0).

Syntax of instruction WRITE_PARAM

WRITE_PARAM %CHxy.i: writing the adjustment parameters of channel i, of the module situated at the xy address (rack number, position in the rack).

For example: READ_PARAM %CH104.2 (writing the parameters of channel 2, of the module situated in position 4 of rack 1).

Monitoring the exchange

The 2 bits which follow may be used to monitor the exchanges of adjustment parameters between the processor and the module:

Bit	Meaning
%MWxy.i.0:X2	This bit is set at 1 when the exchange is in process. It is reset to 0 when the exchange has been completed.
%MWxy.i.1:X2	This bit is set at 1 if the parameters passed on are out of range or erroneous.

SAVE_PARAM and RESTORE_PARAM

Reminder

These services can be used to save or restore adjustment parameters.

SAVE_PARAM: explicit backup of the axis command 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).

Syntax of instruction SAVE_PARAM

SAVE_PARAM %CHxy.i: backup for the adjustment parameters of channel i, of the module situated at the xy address (rack number, position in the rack).

For example: SAVE_PARAM %CH3.1 (saving the parameters of channel 1, of the module situated in position 3 of rack 0).

Syntax of instruction RESTORE_PARAM

RESTORE_PARAM %CHxy.i: restoration of the adjustment parameters of channel i, of the module situated at the xy address (rack number, position in the rack).

For example: RESTORE_PARAM %CH104.2 (restoration of the parameters of channel 2, of the module situated in position 4 of rack 1).

Monitoring the exchange

The 2 bits which follow may be used to monitor the writing of the adjustment parameters in the module:

Bit	Meaning
%MWxy.i.0:X2	This bit is set at 1 when the exchange is in process. It is reset to 0 when the exchange has been completed.
%MWxy.i.1:X2	This bit is set at 1 if the parameters passed on are out of range or erroneous.

WRITE_CMD

Reminder

This service allows a command to be sent to the axis command module.

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 movement command, for example).

Syntax of instruction WRITE_CMD

WRITE_CMD %CHxy.i : writing the command information of channel i, of the module situated at the xy address (rack number, position in the rack).

For example: WRITE_CMD %CH3.1 (writing the command information of channel 1, of the module situated in position 3 of rack 0).

WRITE_CMD Interface

The command to be carried out is defined in the word %MWxy.i.26 and the result of the command is available in the words %MWxy.i.19 to %MWxy.i.24

Address	Type	Symbol	Meaning
%MWxy.i.19	Word	ERROR_CMD	Write error of the command WRITE_CMD
%MDxy.i.20	Double Word	RETURN_CMD_1	Return 1 of the function
%MFxy.i.22	Floating point	RETURN_CMD_2	Return 2 of the function
%MFxy.i.24	Floating point	RETURN_CMD_3	Return 3 of the function
%MWxy.i.26	Word	ACTION_CMD	Action to be carried out
%MDxy.i.27	Double Word	PARAM_CMD_1	Parameter 1
%MDxy.i.29	Double Word	PARAM_CMD_2	Parameter 2
%MFxy.i.31	Floating point	PARAM_CMD_3	Parameter 3
%MFxy.i.33	Floating point	PARAM_CMD_4	Parameter 4

Interface WRITE_CMD 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 by coordinated axis (position and speed):

Address	Type	Symbol	Meaning
...	...	...	...
%MDxy.i.27	Double Word	PARAM_CMD_1	Parameter 1
%MDxy.i.29	Double Word	PARAM_CMD_2	Parameter 2
%MFxy.i.31	Floating point	PARAM_CMD_3	Parameter 3
%MFxy.i.33	Floating point	PARAM_CMD_4	Parameter 4
%MFxy.i.35	Floating point	PARAM_CMD_5	Parameter 5 (*)
...	...	...	...
%MFxy.i.61	Floating point	PARAM_CMD_18	Parameter 18 (*)
(*) Only with the "coordinated axis group" functions.			

Exchange check

The 2 bits which follow may be used to monitor the writing of command information in the module:

Bit	Meaning
%MWxy.i.0:X1	This bit is set at 1 when the exchange is in process. It is reset to 0 when the exchange has been completed.
%MWxy.i.1:X1	This bit is set at 1 if the parameters passed on are out of range or erroneous.

WRITE_CMD Examples

Example 1: Initializing Defaults

Initialize defaults on the bus ring of module 4 situated in rack 1:

```
!   (*If  WRITE_CMD not in progress, then initialization of defaults*)
    IF NOT %MW104.0.0:X1 THEN %MW104.0.26 := 409;
    WRITE_CMD %CH104.0;
    END_IF;
```

Example 2: Writing the Optical Power

Defining (writing) the optical power:

```
!   (*If WRITE_CMD not in progress, then writing (set) of the optical
    power*)
    IF NOT %MW104.0.0:X1 THEN %MW104.0.26 := 2547;
    %MF104.0.31 := 51.5;            //Parameter 3: optical power = 51.5
    WRITE_CMD %CH104.0;
    END_IF;
```

Example 3: Reading the transmission ratio

Read the transmission ratio (BAUD_RATE):

```
!   (*If  WRITE_CMD not in progress, then reading (get) of the
    transmission ratio*)
    IF NOT %MW104.0.0:X1 THEN %MW104.0.26 := 1558;
    WRITE_CMD %CH104.3;
    END_IF;
!   (*If WRITE_CMD has been completed and there is no fault, the value
    of the transmission ratio is accessible in return 1*)
    IF NOT %MW104.0.0:X1 AND NOT %MW104.0.1:X1
    THEN BAUD_RATE: =%MD104.0.20;  //transmission ratio in return 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":

```
!   (*If  WRITE_CMD not in progress, then move real axis 3*)
    IF NOT %MW104.3.0:X1 THEN %MW104.3.26 := 513;
    %MD104.3.27 := 0;              //Parameter 1: type of movement = absolute
    %MF104.3.31 := 105.2;          //Parameter 3: position = 105.2
    %MF104.3.33 := 5.0;            //Parameter 4: speed = 5
    WRITE_CMD %CH104.3;
    END_IF;
```

Example 5: Reading the position of a slave

Reading the relative position of a follower slave when the master is at 102.5:

```
!   (*If WRITE_CMD not in progress, then reading the position of the
    follower slave*)
    IF NOT %MW104.26.0:X1 THEN %MW104.26.26 := 537;
    %MF104.26.31 := 102.5;          //Parameter 3: position of the master =
                                    102.5
    WRITE_CMD %CH104.26;
    END_IF;
!   (*If WRITE_CMD has been completed and there is no fault, the
    position of the slave is accessible in return 2*)
    IF NOT %MW104.26.0:X1 AND NOT %MW104.26.1:X1
    THEN SLAVE_POSITION             //position in return 2
    :=%MF104.26.22;
    END_IF;
```

READ_STS

Reminder

This service allows one to read the status words associated with the axis command module or with the different channels explicitly.

Syntax of the READ_STS (module) instruction

READ_STS %CHxy.MOD: Reading the general diagnostics information of the module situated at the xy address (rack number, position in the rack).

For example: READ_STS %CH203.MOD (reading the diagnostics information of the module situated in position 3 of rack 2).

Syntax of the READ_STS (channel) instruction

READ_STS %CHxy.i: reading the diagnostics information of channel i, of the module situated at the xy address (rack number, position in the rack).

For example: READ_STS %CH102.1 (reading the diagnostics information of channel 1of the module situated in position 2 of rack 1).

Monitoring the exchange

The 2 bits which follow may be used to monitor the exchanges of status words between the module and the processor:

Bit	Meaning
%MWxy.i.0:X0	This bit is set at 1 when the reading is in process. It is reset to 0 when the exchange has been completed.
%MWxy.i.1:X0	This bit gives the report of the exchange. It is set to 1 in the case of default.

TRF_RECIPE and MOD_PARAM

Reminder

- **TRF_RECIPE:**
this instruction is explained in this manual (See *TRF_RECIPE Instruction, p. 272*).

- **MOD_PARAM:**
this instruction is explained in this manual (See *Dynamic Reconfiguration of Groups, p. 420*).

10.2 List of Variables

At a Glance

Subject of this Section

This section lists the accessible variables for the TSX CSY 84 / 164 modules.

What's in this Section?

This section contains the following topics:

Topic	Page
Implicitly Exchanged Input Variables	375
Implicitly Exchanged Output Variables	378
Adjustment Parameters for the SERCOS® Function	381
Adjustment Parameters for an Individual Axis	382
Adjustment Parameters for a Set of Follower Axes	385
Constant Words	389

Implicitly Exchanged Input Variables

Input Bits and Words

The following 32 input bits and the input word are implicitly exchanged between the processor and the axis control module:

Address	Type	Symbol	Description
%Ixy.i.ERR	Bit	ERR	Default channel
%Ixy.i.0	Bit	RAMPING	Indicates whether the axis is accelerating or decelerating
%Ixy.i.1	Bit	STEADY	The speed is steady
%Ixy.i.2	Bit	STOPPING	The movement is decelerating to a stop
%Ixy.i.3	Bit	PROFILE_END	The last profile command has been sent to the module
%Ixy.i.4	Bit	IN_POSITION	The axis is within the in-position band
%Ixy.i.5	Bit	AXIS_HOMING	The axis is homing. For an imaginary axis, this bit is inactive
%Ixy.i.6	Bit	AXIS_HOMED	The axis position reading is referenced off the home position
%Ixy.i.7	Bit	AXIS_NOT_FOLLOWING	The drive is not recognizing module commands
%Ixy.i.8	Bit	HOLDING	The axis is holding in wait position
%Ixy.i.9	Bit	RESUMING	The axis is moving after a hold
%Ixy.i.10	Bit	DRIVE_ENABLED	The drive is enabled
%Ixy.i.11	Bit	DRIVE_DIAG	The drive is performing a class 3 diagnostic
%Ixy.i.12	Bit	DRIVE_WARNING	The drive is performing a class 2 diagnostic
%Ixy.i.13	Bit	DRIVE_FAULT	The drive is performing a class 1 diagnostic
%Ixy.i.14	Bit	DRIVE_DISABLED	The drive is disabled
%Ixy.i.15	Bit	AXIS_SUMMARY_FAULT	Drive fault
%Ixy.i.16	Bit	AXIS_COMM_OK	Communication between the module and the drive is OK
%Ixy.i.17	Bit	AXIS_IS_LINKED	The axis belongs to a set of axes
%Ixy.i.18	Bit	AXIS_IN_COMMAND	The axis is active and can be controlled
%Ixy.i.19	Bit	/	Reserved
%Ixy.i.20	Bit	AXIS_AT_TARGET	The axis is within the in-position band for the target position

Address	Type	Symbol	Description
%Ixy.i.21	Bit	AXIS_POS_LIMIT	The axis has reached the positive limit
%Ixy.i.22	Bit	AXIS_NEG_LIMIT	The axis has reached the negative limit
%Ixy.i.23	Bit	AXIS_WARNING	MotionWarning status returned by the drive
%Ixy.i.24	Bit	BIAS_REMAIN	Offset added to the command position
%Ixy.i.25	Bit	AXIS_MODE_MANUAL	0 : automatic mode (by default) 1 : manual mode
%Ixy.i.26	Bit	DRIVE_REALTIME_BIT1	Drive bit
%Ixy.i.27	Bit	DRIVE_REALTIME_BIT2	Drive bit
%Ixy.i.28	Bit	AXIS_HOLD	The axis is stopped and waiting for a command
%Ixy.i.29	Bit	AXIS_HALT	The axis has stopped
%Ixy.i.30	Bit	AXIS_FASTSTOP	The axis has faststopped
%Ixy.i.31	Bit	AXIS_READY	The axis is ready to respond to a command
%Ixy.i.32	Bit	CONF_OK	The channel is configured
%IFxy.i.0	Floating point	POSITION	Actual position

Implicitly Exchanged Output Variables

Output Bits and Words

The following 32 output bits and the output word are implicitly exchanged between the processor and the axis command module:

Address	Type	Symbol	Description
%Qxy.i.0	Bit	/	Reserved
%Qxy.i.1	Bit	/	Reserved
%Qxy.i.2	Bit	CONTROL_ACQUIRE	Control acquisition
%Qxy.i.3	Bit	/	Reserved
%Qxy.i.4	Bit	CONTROL_JOG_POS	• used on a rising edge to perform continuous movements in a positive direction. • used on a falling edge to stop the movement in progress.
%Qxy.i.5	Bit	CONTROL_JOG_NEG	• used on a rising edge to perform continuous movements in a negative direction. • used on a falling edge to stop the movement in progress.
%Qxy.i.6	Bit	REALTIME_CONTROL_ BIT1	Control bit associated with an IDN command
%Qxy.i.7	Bit	REALTIME_CONTROL_ BIT2	Control bit associated with an IDN command
%Qxy.i.8	Bit	/	Reserved
%Qxy.i.9	Bit	/	Reserved
%Qxy.i.10	Bit	CONTROL_ENABLE	Enables control
%Qxy.i.11	Bit	CONTROL_FOLLOW	Follow control for an axis or a set of slave axes
%Qxy.i.12	Bit	CONTROL_RESUME	Resumes control after a stop
%Qxy.i.13	Bit	CONTROL_INC_POS	used on a rising edge to perform incremental movements on a referenced axis in a positive direction.
%Qxy.i.14	Bit	CONTROL_INC_NEG	used on a rising edge to perform incremental movements on a referenced axis in a negative direction.
%Qxy.i.15	Bit	CONTROL_CLEAR_FAULT	Fault clear control
%Qxy.i.16	Bit	/	Reserved
%Qxy.i.17	Bit	/	Reserved
%Qxy.i.18	Bit	ALLOW_ACQUIRE	Enables acquisition

Address	Type	Symbol	Description
%Qxy.i.19	Bit	/	Reserved
%Qxy.i.20	Bit	/	Reserved
%Qxy.i.21	Bit	/	Reserved
%Qxy.i.22	Bit	/	Reserved
%Qxy.i.23	Bit	/	Reserved
%Qxy.i.24	Bit	/	Reserved
%Qxy.i.25	Bit	/	Reserved
%Qxy.i.26	Bit	ALLOW_ENABLE	Disables axis control
%Qxy.i.27	Bit	ALLOW_FOLLOW	Cancels follow control for an axis or a set of slave axes
%Qxy.i.28	Bit	ALLOW_RESUME	Permits a movement to be continued after a stop using the HOLD command
%Qxy.i.29	Bit	ALLOW_MOVE	Permits a movement after a stop using the HALT command
%Qxy.i.30	Bit	ALLOW_NOT_FASTSTOP	Control after a Faststop
%Qxy.i.31	Bit	ALLOW_NOT_FAULT	Enables fault control
%QDxy.i.0	Double Word	REMOTE_POSITION	Simulated position

Adjustment Parameters for the SERCOS® Function

List of Parameters

With channel 0, the following parameters are exchanged by the commands WRITE_PARAM and READ_PARAM:

Address	Type	Symbol	Description
%MWxy.i.35	Word	CYCLE_TIME	SERCOS® ring cycle time (see chapter 6 (See *Configuration of the SERCOS® Function (channel 0), p. 89*), SERCOS® Function Configuration)
%MWxy.i.36	Word	BAUD_RATE	Ring flow rate (in Bauds)
%MWxy.i.37	Word	OPTICAL_POWER	Optical power in the fiber

Adjustment Parameters for an Individual Axis

List of Parameters

With channels 1 to 16, the following parameters are exchanged by the WRITE_PARAM and READ_PARAM commands:

Address	Type	Symbol	Description
%MWxy.i.35:X0	Bit	ENABLE_ROLLOVER	Enables rollover mode
%MWxy.i.35:X1	Bit	TEST_POSITION_BAND	Enables position test
%MWxy.i.35:X2	Bit	Reserved	Always 0
%MWxy.i.35:X3	Bit	Reserved	Always 0
%MWxy.i.35:X4	Bit	DISABLE_LIMITS_CHECKING	Disables limits checking
%MWxy.i.35:X5	Bit	/	Reserved
%MWxy.i.35:X6	Bit	_FREEWHEEL_STOP (**)	0 = when there is CSY module error or the drive is disabled, a fast stop is generated on the drive deceleration ramp (DECSTOP ramp on LEXIUM configurable using UNILINK,IDN P 3022 or ASCII code). 1 = freewheel stop upon error or disable
%MFxy.i.36	Floating point	ACCEL	Acceleration value
%MFxy.i.38	Floating point	DECEL	Deceleration value
%MWxy.i.40	Word	ACCEL_TYPE	Acceleration type
%MFxy.i.41	Floating point	IN_POSITION_BAND	Value of the in-position band
%MFxy.i.43	Floating point	ENABLE_POSITION_BAND	Value of the monitoring window
%MFxy.i.45	Floating point	ROLLOVER_MAX	Maximum rollover
%MFxy.i.47	Floating point	ROLLOVER_MIN	Minimum rollover
%MFxy.i.49	Floating point	ACCEL_MAX	Maximum acceleration
%MFxy.i.51	Floating point	DECEL_MAX	Maximum deceleration
%MFxy.i.53	Floating point	SPEED_MAX	Maximum speed

Address	Type	Symbol	Description
%MFxy.i.55	Floating point	POSITION_MAX	Maximum position
%MFxy.i.57	Floating point	POSITION_MIN	Minimum position
%MFxy.i.59	Floating point	SCALE_NUMERATOR	Scale factor numerator (*)
%MFxy.i.61	Floating point	SCALE_DENOMINATOR	Scale factor denominator (*)
%MWxy.i.63	Word	ACCEL_UNITS	Acceleration unit
%MWxy.i.64	Word	SPEED_UNITS	Unit of speed
%MWxy.i.65	Word	POSITION_UNITS	Position units

(*) See configuration of an individual axis and the GetGearRatio function.
(**): important:

Note: The action of this bit is set only after writing to the channel configuration %Kx constant words.

In order to do this, perform a SAVE_PARAM of the channel (after having performed the WRITE_PARAM).
For example: SAVE _PARAM on channel 1 of the module slot 4:

```
IF %M37 THEN
SAVE_PARAM %CH4.1;
RESET %M37;
END_IF;
```

Adjustment Parameters for a Set of Follower Axes

List of Parameters

Channels 21 to24 are used to exchange the following parameters with the WRITE_PARAM and READ_PARAM commands for the TSX CSY 84 /164 modules and the MOD_PARAM command for the TSX CSY 164 module:

Address	Type	Symbol	Description
%MWxy.i.35	Word	MASTER_CHANNEL	Number of the master axis
%MWxy.i.36	Word	SLAVE_CHANNEL_1	Number of slave axis 1
%MWxy.i.37	Word	FOLL_DESCRIPTION_1	Definition of slave axis 1
%MWxy.i.37:X0	Bit	FOLL_WHERE_1	0 = Controller
%MWxy.i.37:X1	Bit	FOLL_TYPE_1	0 = Ratio mode; 1 = Cam mode
%MWxy.i.37:X2	Bit	FOLL_POSITION_1	0 = following the actual position; 1 = following the commanded position
%MWxy.i.37:X3	Bit	FOLL_FOLLOW_ON_ HALT_1	1 = stop the slave axis if master/slave link is removed
%MWxy.i.37:X4	Bit	/	Always 0
%MWxy.i.37:X5	Bit	/	Always 0
%MWxy.i.37:X6	Bit	FOLL_HALT_MASTER_1	0 = nothing, 1 = normal master stop upon slave error
%MWxy.i.37:X7	Bit	FOLL_BIAS_REMAINS_1	1 = dynamic offset on position of master
%MWxy.i.37:X8 to %MWxy.i.37:X10	Bit	FOLL_START_1	Start condition: 0 = immediately 1 = master position reaches trigger in negative direction 2 = master position reaches trigger in positive direction 3 = master position > trigger 4 = master position < trigger
%MWxy.i.37:X11	Bit	FOLL_FAULT_MASTER_ 1 (*)	0 = No effect on management of the master upon slave error 1 = if HALT_MASTER (%MWxy,i37:X6) = 0 No master stop. 1 = if HALT_MASTER (%MWxy,i,37:X6) = 1 when there is CSY module error, faulty drive or one of the slave drives is disabled, a fast stop is generated on the drive deceleration ramp of the MASTER axis (DECSTOP decleration ramp on LEXIUM configurable using UNILINK, IDN P 3022 or ASCII code).

Address	Type	Symbol	Description
%MWxy.i.37:X12	Bit	FOLL_FAULT_SLAVE_1 (*)	0 = No effect on management of slaves upon master error 1 = when there is CSY module error or a faulty drive or when the MASTER drive of the slave group is disabled, a fast stop is generated on the drive deceleration ramp of the SLAVE drives (DECSTOP decleration ramp on LEXIUM configurable using UNILINK, IDN P 3022 or ASCII code).
%MWxy.i.37:X13 to %MWxy.i.37:X15	Bit	/	Always 0
%MFxy.i.38	Floating point	NUMERATOR_1	Numerator for slave axis 1
%MFxy.i.40	Floating point	DENOMINATOR_1	Denominator for slave axis 1
%MFxy.i.42	Floating point	TRIGGER_POSITION_1	Value of trigger for slave axis 1
%MWxy.i.44	Word	SLAVE_CHANNEL_2	Number of slave axis 2
%MWxy.i.45	Word	FOLL_DESCRIPTION_2	Definition of slave axis 2. The description of the bits is identical to that of the axis of slave 1.
%MFxy.i.46	Floating point	NUMERATOR_2	Numerator for slave axis 2
%MFxy.i.48	Floating point	DENOMINATOR_2	Denominator for slave axis 2
%MFxy.i.50	Floating point	TRIGGER_POSITION_2	Value of trigger for slave axis 2
%MWxy.i.52	Word	SLAVE_CHANNEL_3	Number of slave axis 3
%MWxy.i.53	Word	FOLL_DESCRIPTION_3	Definition of slave axis 3. The description of the bits is identical to that of the axis of slave 1.
%MFxy.i.54	Floating point	NUMERATOR_3	Numerator for slave axis 3
%MFxy.i.56	Floating point	DENOMINATOR_3	Denominator for slave axis 3
%MFxy.i.58	Floating point	TRIGGER_POSITION_3	Value of trigger for slave axis 3
%MWxy.i.60	Word	SLAVE_CHANNEL_4	Number of slave axis 4
%MWxy.i.61	Word	FOLL_DESCRIPTION_4	Definition of slave axis 4. The description of the bits is identical to that of the axis of slave 1.

Address	Type	Symbol	Description
%MFxy.i.62	Floating point	NUMERATOR_4	Numerator for slave axis 4
%MFxy.i.64	Floating point	DENOMINATOR_4	Denominator for slave axis 4
%MFxy.i.66	Floating point	TRIGGER_POSITION_4	Value of trigger for slave axis 4
%MWxy.i.68	Word	SLAVE_CHANNEL_5	Number of slave axis 5
%MWxy.i.69	Word	FOLL_DESCRIPTION_5	Definition of slave axis 5. The description of the bits is identical to that of the axis of slave 1.
%MFxy.i.70	Floating point	NUMERATOR_5	Numerator for slave axis 5
%MFxy.i.72	Floating point	DENOMINATOR_5	Denominator for slave axis 5
%MFxy.i.74	Floating point	TRIGGER_POSITION_5	Value of trigger for slave axis 5
%MWxy.i.76	Word	SLAVE_CHANNEL_6	Number of slave axis 6
%MWxy.i.77	Word	FOLL_DESCRIPTION_6	Definition of slave axis 6. The description of the bits is identical to that of the axis of slave 1.
%MFxy.i.78	Floating point	NUMERATOR_6	Numerator for slave axis 6
%MFxy.i.80	Floating point	DENOMINATOR_6	Denominator for slave axis 6
%MFxy.i.82	Floating point	TRIGGER_POSITION_6	Value of trigger for slave axis 6

(*) important:

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.
For example: MOD_PARAM on slave group 24 of the module slot 4:

```
IF %M38 THEN
MOD_PARAM %CH4.24(0,0,0,0);
RESET %M38;
END_IF;
```

Constant Words

List of Constant Words

The table below lists the constant words:

Address	Type	Symbol	Description
%KWxy.i.0	Word	CHANNEL_ID	Channel identification
%KWxy.i.2	Word	SERCOS_ADD	SERCOS® address of the axis (only for a real axis or a remote axis)

Appendices

11

At a glance

Subject of this chapter

This chapter provides all the appendices for the document: performance and limits, list of motion data, IDN read/write functions, etc.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
11.1	List of Accessible Functions	392

11.1 List of Accessible Functions

At a Glance

Subject of this Section

This section recalls the different accessible functions for the TSX CSY 84 / 164modules.

What's in this Section?

This section contains the following topics:

Topic	Page
Accessible Functions for SERCOS® Channel 0	393
Accessible Functions for a Real Axis	395
Accessible Functions for an Imaginary Axis	399
Accessible Functions for a Remote Axis	403
Accessible Functions for a Group of Co-ordinated Axes	405
Accessible Functions for a Group of Slave Axes	407
Accessible Functions for a Cam Profile	409

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 drives	None
GetNumberInSet	541	Feedback 1: No. of axes	None
GetSercosAddress	549	Feedback 1: Address	Parameter 1: Axis
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® drives:

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 2: Cœfficient Acc/Dec 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 2: Cœfficient Acc/Dec 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
GetDecelMax	1117	Feedback 2: Deceleration	None
GetDefaultSpeed	1065	Feedback 1: Speed	None
GetChangeACCDECMove	1172	Feedback 1: 1: FCt enabled: Acknowledgement of PARAM_CMD_2 in opcodes 513 and 520. 0: FCt disable	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

Function	Code	Feedback	Parameters
SetChangeACCDECMove	2172	None	Parameter 1: 1: FCt enabled: Acknowledgement of PARAM_CMD_2 in opcodes 513 and 520. 0: FCt disable
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
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: 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® drives:

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	Ext Ret	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 2: Coefficient ACC/DEC 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 2: Coefficient ACC/DEC 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
GetDefaultSpeed	1065	Feedback 1: Speed	None
GetChangeACCDECMove	1172	Feedback 1: 1: FCt enabled: Acknowledgement of PARAM_CMD_2 in opcodes 513 and 520. 0: FCt disable	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

Function	Code	Feedback	Parameters
SetChangeACCDECMove	2172	None	Parameter 1: 1: FCt enabled: Acknowledgement of PARAM_CMD_2 in opcodes 513 and 520. 0: FCt disable
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

III

At a Glance

Subject of this Part

This part describes the new functions specific to the TSX CSY 164 module.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
12	Compatibility	413
13	Configuration of the TSX CSY 164 module	415
14	Functions specific to the TSX CSY 164	419
15	Performances of the TSX CSY 164	427

Compatibility

12

Compatibility

Software Compatibility

- To develop an application using the TSX CSY 164 modules, the PL7 Junior/Pro software must be minimum version SV 4.3 + Motion TSX CSY 164 Add-on or SV ≥ 4.4.
- For a TSX CSY 84 module, a program is compatible with a TSX CSY 164 module after reconfiguring the PL7 application.

Configuration of the TSX CSY 164 module

13

At a Glance

Subject 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's in this Chapter?

This chapter contains the following topics:

Topic	Page
Configuration of the TSX CSY 164 module	416
Configuring channels	417

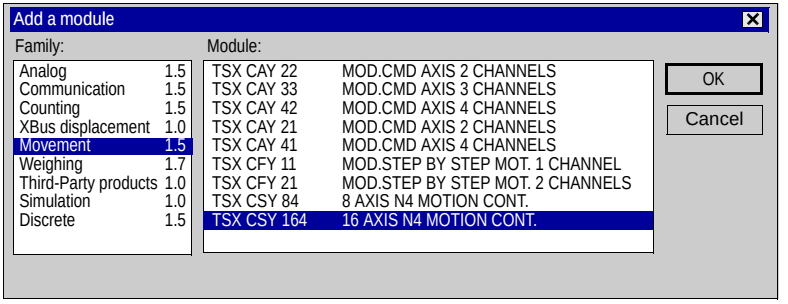
Configuration of the TSX CSY 164 module

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 *Module Configuration, p. 85*).

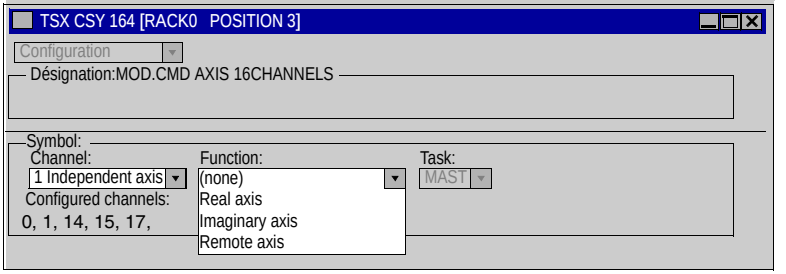
Configuring channels

Introduction

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

- Channel 0: SERCOS® function (See *Configuration of the SERCOS® Function (channel 0), p. 89*),
- Channels 1 to 16: independent axis (real axis, imaginary axis, or remote axis),
- Channels 17 to 20: coordinated axes group (See *Configuration of a set of coordinated axes (channels 17 à 20), p. 97*),
- Channels 21 to 24: slave axes group (See *Configuration of a set of follower axes (channels 21 to 24), p. 99*),
- Channels 25 to 31: cam profile (See *CamProfile configuration, p. 103*).

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 variable speed controller). 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 variable speed controller 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 a remote axis is the same as configuring a real or imaginary axis in which only the position data 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 *Dynamic Reconfiguration of Groups, p. 420*).

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 *Dynamic Reconfiguration of Groups, p. 420*).

Functions specific to the TSX CSY 164

14

At a Glance

Subject of this Chapter

This chapter describes the new functions associated with the TSX CSY 164 module.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Dynamic Reconfiguration of Groups	420
Monitoring Function of the Position Deviations Between Axes	422

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 %CHxy,i: writing %Mx adjustment words of group i, of the module situated at the xy 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:
%MWxy,i,0:x15: this bit is set to 1 when the exchange is in process. It is reset to 0 when the exchange has been completed.
%MWxy,i,1:x15: this bit is set to 1 when the exchange is erroneous (Parameters out of range, etc.)

If you are running PL7, this function is written as follows:

- **Mod_Param % CHxy.i (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_RECIPE (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 xy.0.15	Floating point	Instance number	Parameter of the TRF_RECIPE, 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	
%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_ RECIPE 26200: error on the %MWxy.0.3 word

- The size of the data (first parameter of TRF_RECIPE) is incorrect => error code 23
- The instance number is incorrect => error code 10. The 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_RECIPE (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 %MWxy.0.3 word;

- The size of the data (first parameter of TRF_RECIPE) is incorrect => error code 23
- The instance number is incorrect => error code 10. The 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 %MWxy.0.3 word;

- The instance number is incorrect => error code 10. The 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_RECIPE (opcode 16201) on channel 0.
The result of the function can be read on a universal %MW.
TRF_RECIPE %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_RECIPE %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 %MWxy, i;3 word:

- 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

15

Performances of the TSX CSY 164 Module

At a Glance

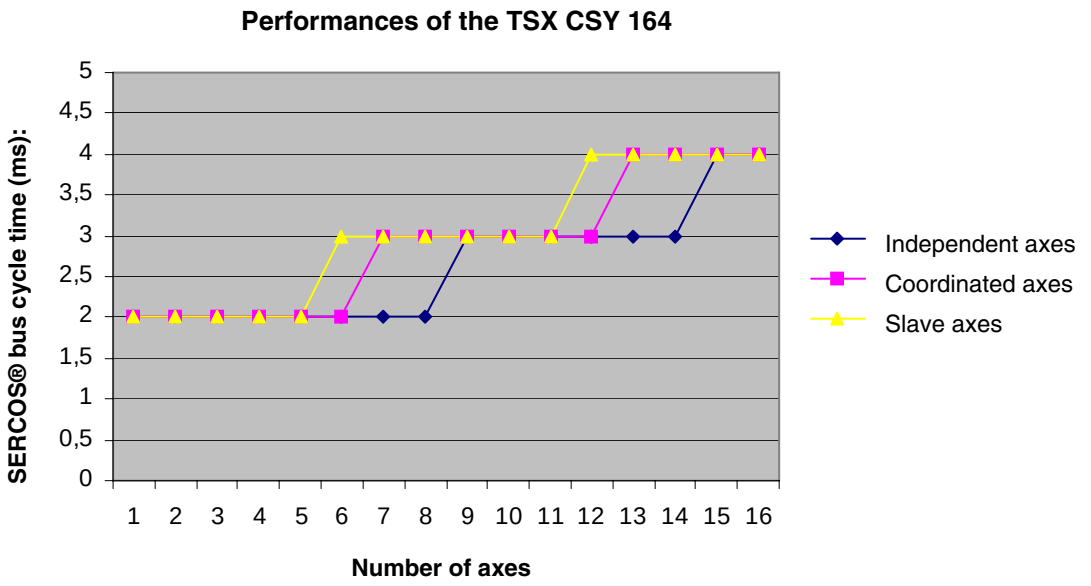
The performances of the module are:

- 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:



Test conditions:

Warning: 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

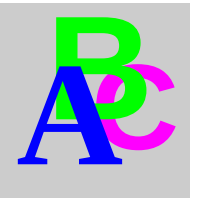
```
(* change of the SERCOS® cycle time => %MW30 = 2000 or 3000 or
4000 *)
IF %M30 THEN %MW101.0.26:=2565;
%MD101.0.27:=%MW30;
IF NOT %MW101.0:X1 THEN
WRITE_CMD %CH101.0;
RESET %M30;
SET %M31;
END_IF;
END_IF;

(* 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



Symbols

"Cam profile" function
    SERCOS® module, 81
"Coordinated axis group" function
    SERCOS® module, 75
"Imaginary axis" function
    SERCO(r) module, 70
"Real axis" function
    SERCOS(r) modules, 69
"Remote axis" function
    SERCOS(r) module, 71
"SERCOS®" function, 68
"Slave axis group" function
    SERCOS® module, 78

A

ABS_MOVE, 161
ABS_MOVE_NEG, 161
ABS_MOVE_POS, 161
acceleration/deceleration parameter
    Accel, 218
    AccelMax, 220
    AccelType, 222
    DecelMax, 221
AccelerationType, 224
Access to module parameters
    SERCOS® module, 85, 416
ACTION_CMD, 368
Activation of LEDs
    SERCOS® module, 350
Adjusting an individual axis
    SERCOS® Module, 382, 385
Adjustment
    Screen
        General, 199
        SERCOS® ring, 275
adjustment
    screen
        real or imaginary axes, 204
        remote axes, 207
        slave axes, 239
    TRF_RECIPE function, 209, 270
Adjustment functions
    SERCOS® module, 395, 399
adjustment functions
    SERCOS® module, 393, 403, 405, 407, 409
ALLOW bit, 145
ALLOW Bits, 146
Axis command module parameters
    SERCOS® module, 364
Axis configuration, 41
    SERCOS® module, 40
Axis X à Axis E
    SERCOS® module, 97
AXIS_AT_TARGET, 129
AXIS_COMM_OK, 127
AXIS_FASTSTOP, 140
AXIS_HALT, 139
AXIS_HOLD, 138
AXIS_HOMED, 117
AXIS_HOMING, 116

AXIS_IN_COMMAND, 132
AXIS_IS_LINKED, 128
AXIS_MANUAL_MODE, 135
Axis_Manual_Mode, 153
AXIS_NEG_LIMIT, 131
AXIS_NOT_FOLLOWING, 118
AXIS_POS_LIMIT, 130
AXIS_READY, 141
AXIS_SUMMARY_FAULT, 126, 350
AXIS_WARNING, 133

B

BIAS_REMAIN, 134

C

Cam
    SERCOS® module, 81
Cam mode
    SERCOS® module, 78
cam profile parameters
    InterpType, 268
    length, 264
    LookUpFollowerPosition, 265
    ProfileId, 266, 269
CamProfile configuration
    SERCOS® Module, 103
Channel 0
    SERCOS® module, 68
channel diagnostics, 308
channel fault, 308
CONF_OK, 142
Configuration functions
    SERCOS® module, 395
configuration functions
    SERCOS® module, 403, 407, 409
Configuration of the module
    SERCOS® module, 85, 416
CONT_MOVE, 161
CONTROL bit, 145
CONTROL bits, 149
Cycle time
    SERCOS® module, 89
cyclic data
    SERCOS® module, 18

D

Debug
    Screen
        General, 290
debug
    diagnostic, 295
    drive, 305
    enable, 294
    following, 304
    manual command, 302
    movement, 299
    position, 306
    reference, 303
    send command, 301
Declaration of the module
    SERCOS® module, 85, 416
Défauts voie
    SERCOS® module, 330
Deletion of a module
    SERCOS® module, 85, 416
Diagnostic functions
    SERCOS® module, 395
diagnostic functions
    SERCOS® module, 393, 403, 405, 407
diagnostic module, 307
digital link
    SERCOS® module, 18
drive faults, 352
Drive Parameters
    IDN_P, 279
    IDN_S, 279
Drive parameters
    IDN_UP, 282
    IDN_US, 282
DRIVE_DIAG, 122
DRIVE_DISABLED, 125
DRIVE_ENABLED, 121
DRIVE_FAULT, 124
DRIVE_REALTIME_BIT1, 136
DRIVE_REALTIME_BIT2, 137
DRIVE_WARNING, 123
Drives
    SERCOS(r) modules, 69

E

Enable position checking
SERCOS(r) module, 417
enable position checking
SERCOS(r) module, 90
Enter symbols
SERCOS® module, 43

F

faulty channel
SERCOS® module, 320
faulty module, 307
SERCOS® module, 320
Flow
SERCOS® module, 89
ForcedHomed, 157

G

GetActualPhase, 312
GetActualSpeed, 181
GetCommandedPosition, 186
GetCommandPhase, 313
GetCommandSpeed, 182
GetMotionFault, 323
GetMotionWarning, 327
GetMoveQueueLength, 168
GetNumberInSet, 318
GetNumberOfDrivesInRing, 316
GetSercosAddress, 315
GetSpeedOverride, 169
GetUnrolledCommandedPosition, 185
GetUnrolledPosition, 184

H

HOLDING, 119
Home, 154, 156

I

IDN, 18
Implementing a real axis
SERCOS® module, 30
Implementing a remote input
SERCOS® module, 30
Implementing an imaginary axis
SERCOS® module, 30
IN_POSITION, 115
INCR_MOVE, 161
Input Word and Bit
SERCOS® module, 375
Instruction
TRF_RECIPE, 272
Internal variable symbols
SERCOS® module, 43
Interpolation
SERCOS® module, 81
IsLoopUp, 317

L

Limits
SERCOS(r) module, 417
limits
SERCOS(r) module, 90

M

manual mode, 151
Manufacturing parameters
SERCOS® module, 89
MAST task
SERCOS® module, 66
Master
SERCOS(r) module, 99
Master increment
SERCOS® Module, 103
Module faults
SERCOS® module, 329
modulo
SERCOS® module, 229
MotionStatus, 321
MoveImmed, 164
Movement
SERCOS(r) module, 417
movement
SERCOS(r) module, 90
Movement control functions
SERCOS® module, 395, 399

movement control functions
SERCOS® module, 393, 403, 405, 407
Movement functions
SERCOS® module, 395, 399
movement functions
SERCOS® module, 405
MovementControl, 146
MovementWarning, 327
MoveQueue, 166

N

non-cyclic data
SERCOS® module, 18

O

Operating modes, 19
operating modes
SERCOS® module, 18
Optical power
SERCOS® module, 89
Output Bits and Words
SERCOS® module, 378

P

PARAM_CMD, 368
parameters of a slave group
FollowerConfig, 246
Performance optimization
SERCOS® module, 66
position limit and modulo parameter
EnableRollover, 232
PositionLimit, 227
RolloverLimit, 231
position limit parameter
InPositionBand, 234
Premium package
SERCOS® module, 20
Presymbolization
SERCOS® module, 84
PROFILE_END, 114
Programming errors
SERCOS® Module, 340

R

RAMPING, 111
Ratio mode
SERCOS® module, 78
Read/Write Functions for IDN Parameters, 393
Read/write functions for IDN parameters, 395
READ_PARAM, 366
READ_STS, 372
RealTimeCtrlBit, 170
Reference speed
SERCOS® module, 75
Remote value
SERCOS(r) module, 71
REMOTE_POSITION, 71
resolution parameters
GearRatio, 225
RESTORE_PARAM, 367
RESUMING, 120
RETURN_CMD, 368

S

SAVE_PARAM, 367
Scale factor
SERCOS(r) module, 417
scale factor
SERCOS(r) module, 90
SERCOS® module, 95
SERCOS Architecture, 20
Sercos ring parameters
OpticalPower, 277
SERCOS® function adjustment, 381
Set_Functional_Mode, 153
SetCommandPhase, 313
SetPosition, 159
SetSpeedOverride, 169
Slave
SERCOS(r) module, 99
Slave group parameters
AbsFollowerBias, 257
slave group parameters

FollowerMode, 245
FollowerBias, 256
MasterOffset, 252
MasterTriggerPosition, 254
Slave increment
SERCOS® Module, 103
SlaveMode
SERCOS® module, 244
Special function
SERCOS® module, 97
Speed drive parameters
SERCOS® module, 364
Speed modulation coefficient
SERCOS(r) module, 69
speed modulation coefficient
SERCOS® module, 169
speed parameter
DefaultSpeed, 215
SpeedLimit, 216
standard EN 61491, 17
STEADY, 112
Stop and bias sequence
SERCOS® module, 78
STOPPING, 113
Syntax of the symbols
SERCOS® module, 84

T

Tangential speed
SERCOS® module, 75
Trajectory calculation
SERCOS® module, 20
TRF_RECIPE, 209, 270, 272

U

Unhome, 158
Units, 417
units, 90, 95

V

variable speed controller connection
SERCOS® module, 39

W

WRITE_CMD, 368
WRITE_PARAM, 366