# Premium and Atrium using EcoStruxure™ Control Expert

## SERCOS® Motion Control Module TSX CSY 85 Setup Manual

(Original Document)

12/2018

35010451.09

www.schneider-electric.com

Schneider Electric

# Table of Contents

# Safety Information

## Important Information

### NOTICE

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

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

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

| DANGER |
|---|
| **DANGER** indicates a hazardous situation which, if not avoided, **will result in** death or serious injury. |

| WARNING |
|---|
| **WARNING** indicates a hazardous situation which, if not avoided, **could result in** death or serious injury. |

| CAUTION |
|---|
| **CAUTION** indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury. |

| *NOTICE* |
|---|
| ***NOTICE*** is used to address practices not related to physical injury. |

## PLEASE NOTE

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

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

# About the Book

## At a Glance

### Document Scope

This manual contains specific instructions for installing the motion control software for the SERCOS® TSX CSY 85 module.

Only the specific features of the TSX CSY 85 module will be described in this manual. In the absence of a specific description, the TSX CSY 85 module functions are identical to those of the TSX CSY 84 module, as described in the TSX CSY 84/164 module setup manual.

**NOTE:** All mounting procedures for the TSX CSY 84 module are the same as for the TSX CSY 85 module.

### Validity Note

This documentation is valid for EcoStruxure™ Control Expert 14.0 or later.

### Product Related Information

| ⚠ WARNING |
|---|
| UNINTENDED EQUIPMENT OPERATION<br>The application of this product requires expertise in the design and programming of control systems. Only persons with such expertise should be allowed to program, install, alter, and apply this product.<br>Follow all local and national safety codes and standards.<br>**Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

# Chapter 1
# SERCOS® Architecture

## Aim of This Chapter

This chapter provides a brief introduction to the digital link between a master and its slaves, as defined by standard EN 61491 "Serial data link for real-time communication between controls and drives".

## What Is in This Chapter?

This chapter contains the following topics:

## Introduction

### Digital Link

The digital link between an axis control module (master) and intelligent servodrives (slaves) is defined by European standard EN 61491 relative to 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 servodrives, which reduces cabling costs.

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

### Exchanged Data

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

- cyclic data exchanged from master to slaves (position 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 (e.g. to complete a parameter read).

### Identification of Exchanged Objects

All objects may be accessed using an identification number: IDN.

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

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

## Operating Modes

The bus operating modes follow 5 phases as follows:

At start up

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

In normal operation

| Phase | Operating mode |
|---|---|
| Phase 4 | Active cyclic exchanges. |

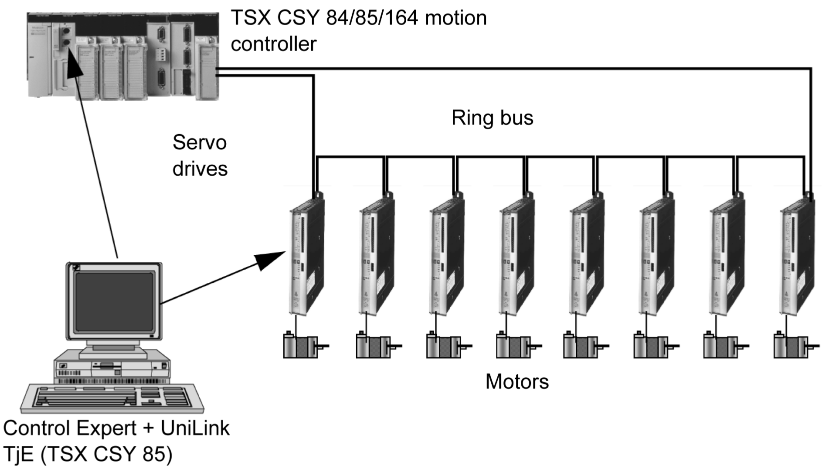
Because each servodrive acts as a repeater on the bus, the system switches to phase 0 in the event of a power supply cut, a communication error, a fault on one of the servodrives or the disconnection of the power supply to the bus.

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

## Introduction to SERCOS® Architecture

### Overview

The overview of the SERCOS® architecture is as follows:


TSX CSY 84/85/164 motion controller
Ring bus
Servo drives
Motors
Control Expert + UniLink TjE (TSX CSY 85)

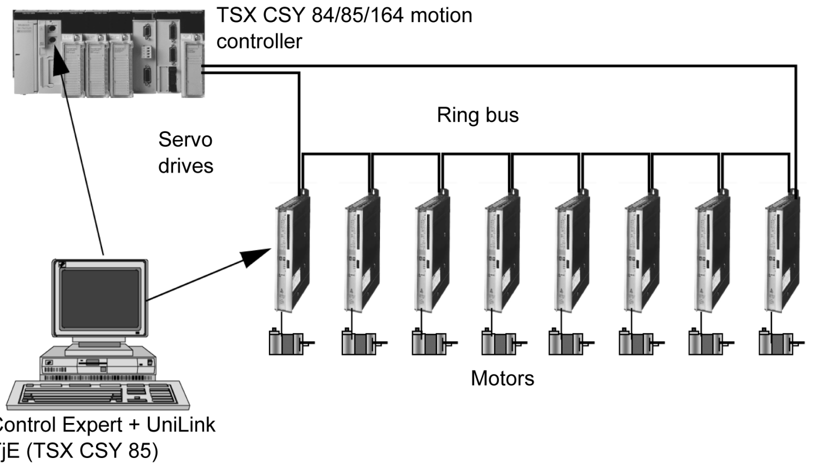
### Introduction to the Premium Package

The Premium package consists of:

- The range of Premium PLCs (rack, supply, processor, etc.): TSX/PCX/PMX 57,
- TSX CSY 84 or TSX CSY 85 motion controllers, which can control up to 8 servodrives via the ring bus,
- TSX CSY 164 motion controller, which can control up to 16 servodrives via the ring bus,
- The range of Lexium 17S servodrives (4 A to 56 A): LXM 17S
- The range of motors: BPH...,
- UniLink software, which can be used to enter parameters and adjust servodrives
- TjE (trajectory editor)

## Functional Diagram

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


TSX CSY 84/85/164 motion controller
Ring bus
Servo drives
Motors
Control Expert + UniLink TjE (TSX CSY 85)

**NOTE:** It is possible to use a servodrive 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

TSX CSY 84/85 motion controllers calculate trajectories and interpolate multiple axes.

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

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

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

## Development of the TSX CSY xxx Modules

### At a Glance

The version of the module is displayed:

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

| | CSY 84 V1.1 | CSY 84 V1.2 | CSY 84 V1.3 | CSY 84 V1.6 | CSY 85 V1.0 | CSY 85 V1.6 | CSY 164 V1.0 | CSY 164 V1.3 | CSY 164 V1.6 |
|---|---|---|---|---|---|---|---|---|---|
| Option to close the position loop via an incremental encoder or SSI on Lexium 17S and 17HP servo drives **(1)** (Not documented) | X | X | X | X | X | X | X | X | X |
| Option to switch the command position to torque **(1) and (2)** (Not documented) | X | X | X | X | X | X | X | X | X |
| Option to switch the command position to speed command **(1) and (2)** (Not documented) | X | X | X | X | X | X | X | X | X |
| Read or write the servodrive parameters | X | X | X | X | X | X | X | X | X |
| Position loop with external encoder **(1)** | X | X | X | X | X | X | X | X | X |
| Torque mode**(1)** | X | X | X | X | X | X | X | X | X |
| Speed mode **(1)** | X | X | X | X | X | X | X | X | X |
| Manual mode | | X | X | X | X | X | X | X | X |
| Behavior upon fault or independent axis disable (FREEWHEEL_STOP) **(3))** | | | X | X | X | X | X | X | X |
| Propagation of stops in a slave group | | | X | X | X | X | X | X | X |
| Alignment function of the emergency-stop inclines | | | X | X | X | X | X | X | X |

| | CSY84 V1.1 | CSY84 V1.2 | CSY84 V1.3 | CSY84 V1.6 | CSY85 V1.0 | CSY85 V1.6 | CSY164 V1.0 | CSY164 V1.3 | CSY164 V1.6 |
|---|---|---|---|---|---|---|---|---|---|
| Selection of stop behavior following latching command | | | X | X | X | X | X | X | X |
| Stop of a member axis of a slave group in an absolute position following an Unlink | | | X | X | X | X | | X | X |
| Stop of a member axis of a slave group in an absolute position | | | X | X | X | X | | X | X |
| "Target sequence activation" function | | | X | X | X | X | | X | X |
| Modulo counter service | | | X | X | X | X | | X | X |
| Switching ACC and DEC parameters in the MOVE | | | X | X | X | X | | X | X |
| Sequence stop (Unlink) of the slave group set off by a trigger | | | X | X | X | X | | X | X |
| Dynamic configuration of individual channels | | | | | | | X | X | X |
| Dynamic reconfiguration of groups **(4)** | | | | | | | X | X | X |
| Monitoring function of the deviations between axes | | | | | | | X | X | X |
| Specific interpolation functions | | | | | X | X | | | |
| Specific cam functions | | | | | X | X | | | |
| Absolute move above rollover limit for independent axes | | | | X | | X | | | X |
| Movement functions for coordinated axis compatible with MMS Quantum | | | | X | | X | | | X |
| Function GetPositionCounter and SetPositionCounter | | | | X | | X | | | X |

| | CSY 84 V1.1 | CSY 84 V1.2 | CSY 84 V1.3 | CSY 84 V1.6 | CSY 85 V1.0 | CSY 85 V1.6 | CSY 164 V1.0 | CSY 164 V1.3 | CSY 164 V1.6 |
|---|---|---|---|---|---|---|---|---|---|
| New adjustment functions for real axis: EnableDriveWarning and DisableDriveWarning | | | | X | | X | | | X |

**(1)** Requires modification of files for the TSX CSY 84/85/164 (Sercos.cfg) modules.

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

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

**(4)** The MOD_PARAM function is not available for the TSX CSY 84 and TSX CSY 85 modules.

**NOTE:** The CSY 84/164 V1.3 and CSY 85 V1.0 are the minimal versions. Later versions are currently available.

# Chapter 2
# Implementation methodology

## Aim of this chapter

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

## What Is in This Chapter?

This chapter contains the following topics:

## Introduction

### TSX CSY 85 Module Functions

The TSX CSY 85 module can be used to perform the following functions:

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

**NOTE:** The functions in bold type are the only ones required to set up the interpolated trajectory function specific to the TSX CSY 85 module.

**NOTE:** The TSX CSY 85 module also provides 7 cam profiles (channels 25 to 31).

### Real Axis

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

### Imaginary Axis

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

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

### Remote Axis

A remote axis can be used to upload an item of remote position data to the module. Typically, the TSX CSY 85 module must take up a sequential position from an encoder controlled by an external mechanism and connected to the output of the auxiliary position of the servodrive.

### Group of Co-ordinated Axes

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

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

## Slave Axis Group

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

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

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

## Implementing an Independent Axis

### Introduction

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

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

### Implementation Methodology for a Real Axis

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

Implementation methodology for a real axis occurs in 3 phases:
- Phase 1: Configuration of the servodrive using the UniLink software, creation of the trajectory using the TjE (Trajectory Editor) software.
- Phase 2: configuring the TSX CSY 84 module, using the Control Expert configuration editor (module declaration and parameter configuration for all axes used),
- Phase 3: Writing of the application program, transfer of this program to the PLC, transfer of trajectory data to the PLC, and application debugging.

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

### Implementation Methodology for an Imaginary Axis

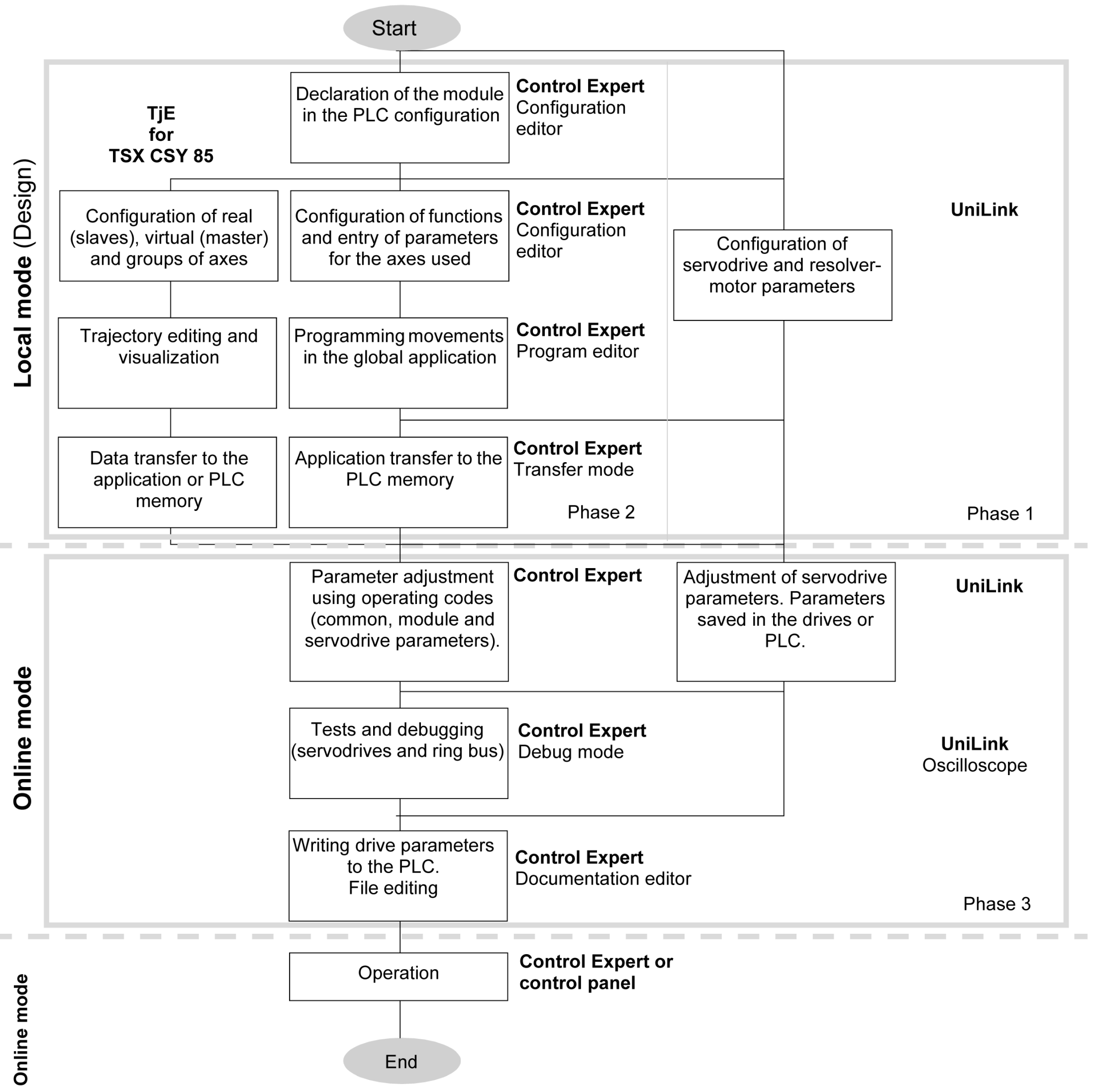
An imaginary axis is not connected to a servodrive (it is not a physical axis). Except for operations linked to a servodrive that does not exist, an imaginary axis is implemented in the same way as a real axis.

### Implementation Methodology for a Remote Input

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

## Implementation Methodology for a Real Axis

### Overview


Start
Local mode (Design)
TjE
for
TSX CSY 85
Declaration of the module in the PLC configuration
Control Expert
Configuration editor
Configuration of real (slaves), virtual (master) and groups of axes
Configuration of functions and entry of parameters for the axes used
Control Expert
Configuration editor
UniLink
Configuration of servodrive and resolver-motor parameters
Trajectory editing and visualization
Programming movements in the global application
Control Expert
Program editor
Data transfer to the application or PLC memory
Application transfer to the PLC memory
Control Expert
Transfer mode
Phase 2
Phase 1
Online mode
Parameter adjustment using operating codes (common, module and servodrive parameters).
Control Expert
Adjustment of servodrive parameters. Parameters saved in the drives or PLC.
UniLink
Tests and debugging (servodrives and ring bus)
Control Expert
Debug mode
UniLink
Oscilloscope
Writing drive parameters to the PLC.
File editing
Control Expert
Documentation editor
Phase 3
Online mode
Operation
Control Expert or control panel
End

## 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.

# Chapter 3
# TSX CSY 85 Functions at a Glance

## Aim of This Chapter

This chapter introduces the specific functions of the TSX CSY 85 module.

## What Is in This Chapter?

This chapter contains the following topics:

## The Specific Features of the TSX CSY 85 Module at a Glance

### Introduction

As we have already seen, the TSX CSY 85 is identical to the TSX CSY 84 in every way, with the exception of a set of software interpolation functions.

With the assistance of the trajectory editor software **TjE**, trajectories can easily be created from a number of points and associated with groups of 2 or 3 axes using these new functions.

**NOTE:** The maximum number of points allowed when creating trajectories can varies depending on the firmware version:
- For firmware version 1.1 or earlier:
  The maximum number of reference points allowed is 60.
- For firmware version 1.2 or later:
  The maximum number of reference points allowed is 200.

Up to 7 cam profiles are available on this module. This means that up to 2 groups of 3 axes or even 3 groups of 2 axes can be managed. (These groups must be independent and run simultaneously.)

The trajectories are created using points, which define segments according to types of interpolation:

- Strictly linear
- Linear with link using polynomial interpolation (cubic or 5th degree)
- Linear with link using circular interpolation
- Circular

**NOTE:** The trajectory editor software is supplied with the **TSX CSY 85 trajectory function CD,** which includes:
- The TjE software (trajectory editor tools)
- The TjE user guide

## Principle

The trajectory is created by linking real slave axes to a master axis via a slave axis group. Cam profiles (table of 5000 master points maximum for 4996 real master points) are calculated automatically in order to ascertain this trajectory.

Description of principle:

- The cam profiles associated with each slave axis are calculated from the interpolation table using the internal module algorithm.
- The slave axes are associated with the master axis.
- The master trajectory is calculated by programming the slave axes using the standard TRF_RECIPE function and parameters specific to each type of interpolation.
- Then, positions on each associated slave axis can be obtained instantaneously by managing the master trajectory (manual or automatic, via program settings or the user interface).

**NOTE:** The interpolation algorithm can be used to create trajectories with absolute positioning (trajectory reference point same as that of the axes) or relative positioning (movement starts at the current position on the axis).

**NOTE:** The new functions supported can also be used both to ascertain the maximum speed permitted on the master axis and, as a consequence, on each axis for each segment of the trajectory, and to perform a complete movement at maximum possible speed (constant speed on all trajectories), taking into account the maximum acceleration rates supported by the motors according to the dynamics of the trajectory.

## Limitation of the TSX CSY 85 Module

Control Expert software accepts the MOD_PARAM function when used with the TSX CSY 85 module but it is not operational. Hence, the MOD_PARAM function is not available for the TSX CSY 85 module but only for the TSX CSY 164 module.

## A Reminder of How the TRF_RECIPE Instruction Works

### Reminder

This instruction is used to read or write servodrive parameters with the "Real axis" function.

**NOTE:** It may also be used to read or write cam profiles and to initiate the execution of special functions.

### Syntax of TRF_RECIPE Instruction

TRF_RECIPE `%CHr.m.c` (length, address `%MW`): Transfers parameters from the servodrive, 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 `%MWr.m.c.10` (ACTION_TRF).

**Example**: TRF_RECIPE `%CH1.4.3(10,100)` (load servodrive 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 `%MWr.m.c.10` and the result of the command is available in the word `%MWr.m.c.3` to `%MWr.m.c.8`.

| Address | Type | Symbol | Meaning |
|---|---|---|---|
| `%MWr.m.c.3` | Word | ERROR_TRF | Write error of the TRF_RECIPE command |
| `%MDr.m.c.4` | Double word | RETURN_TRF_1 | Return 1 of the function |
| `%MFr.m.c.6` | Floating point | RETURN_TRF_2 | Return 2 of the function |
| `%MFr.m.c.8` | Floating point | RETURN_TRF_3 | Return 3 of the function |
| `%MWr.m.c.10` | Word | ACTION_TRF | Action to be carried out |
| `%MDr.m.c.11` | Double word | PARAM_TRF_1 | Parameter 1 |
| `%MDr.m.c.13` | Double word | PARAM_TRF_2 | Parameter 2 |
| `%MFr.m.c.15` | Floating point | PARAM_TRF_3 | Parameter 3 |
| `%MFr.m.c.17` | Floating point | PARAM_TRF_4 | Parameter 4 |

## Actions Carried Out by TRF_RECIPE

The actions that can be carried out using the TRF_RECIPE service are:

| Function | ACTION_TRF (%MWr.m.c.10) | Meaning |
|---|---|---|
| Real axis (1) | 14905 | New function of the TSX CSY 85: Read minimum possible speed of trajectory using interpolation tables sent to the module via code 26900. |
| Real axis (1) | 16001 | Write variable servodrive parameters to the PLC memory. |
| Axis group | 16901 | New function of the TSX CSY 85: Read motion parameters following a trajectory calculation (code 26900).<br>● Key master positions<br>● Maximum possible speed<br>● Calculated speed (speed actually applied) |
| Real axis (1) | 26001 | Download servodrive parameters from the PLC memory. |
| Axis group (1) | 26900 | New function of the TSX CSY 85: Calculate trajectories in accordance with interpolation segments entered as input parameters. |

Legend

| | |
|---|---|
| (1) | PARAM_TRF_1 to PARAM_TRF_4 = 0 |

## A Reminder of How the WRITE_CMD Instruction Works

### Reminder

This service allows a command to be sent to the motion controller.

`WRITE_CMD`: Explicit writing of command words in the module. This operation is carried out via internal words `%MW`, which contain the command to be carried out and its parameters (a motion control, for example).

### Syntax of WRITE_CMD Instruction

`WRITE_CMD %CHr.m.c`: Write channel i command information for the module located in address xy (rack number, position in the rack).

**Example**: `WRITE_CMD %CH0.3.1` (write channel 1 command information for the module located in position 3 of rack 0).

### WRITE_CMD Interface

The command to be carried out is defined in word `%MWr.m.c.26` and the result of the command is available in words `%MWr.m.c.19` to `%MWr.m.c.24`.

| Address | Type | Symbol | Meaning |
|---|---|---|---|
| `%MWr.m.c.19` | Word | ERROR_CMD | WRITE_CMD command write error |
| `%MDr.m.c.20` | Double word | RETURN_CMD_1 | Return 1 of the function |
| `%MFr.m.c.22` | Floating point | RETURN_CMD_2 | Return 2 of the function |
| `%MFr.m.c.24` | Floating point | RETURN_CMD_3 | Return 3 of the function |
| `%MWr.m.c.26` | Word | ACTION_CMD | Action to be carried out |
| `%MDr.m.c.27` | Double word | PARAM_CMD_1 | Parameter 1 |
| `%MDr.m.c.29` | Double word | PARAM_CMD_2 | Parameter 2 |
| `%MFr.m.c.31` | Floating point | PARAM_CMD_3 | Parameter 3 |
| `%MFr.m.c.33` | Floating point | PARAM_CMD_4 | Parameter 4 |

### Monitoring the Exchange

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

| Bit | Meaning |
|---|---|
| `%MWr.m.c.0:X1` | This bit is set to 1 while the exchange is in progress. It is reset to 0 when the exchange has been completed. |
| `%MWr.m.c.1:X1` | This bit is set to 1 if the parameters transmitted are out of range or erroneous. |

## Configuring an Individual Axis (Channels 1 to 12)

### Introduction

Use Control Expert software to configure the TSX CSY 85 module.

An individual axis can be either a real axis (channels 1 to 8) or an imaginary axis (channels 9 to 12). The configuration of a real axis enables control of a physical axis (which uses a servodrive). In this case, a certain consistency must be ensured between the parameters entered in the TSX CSY 85 module configuration screen and those defined during servodrive configuration.

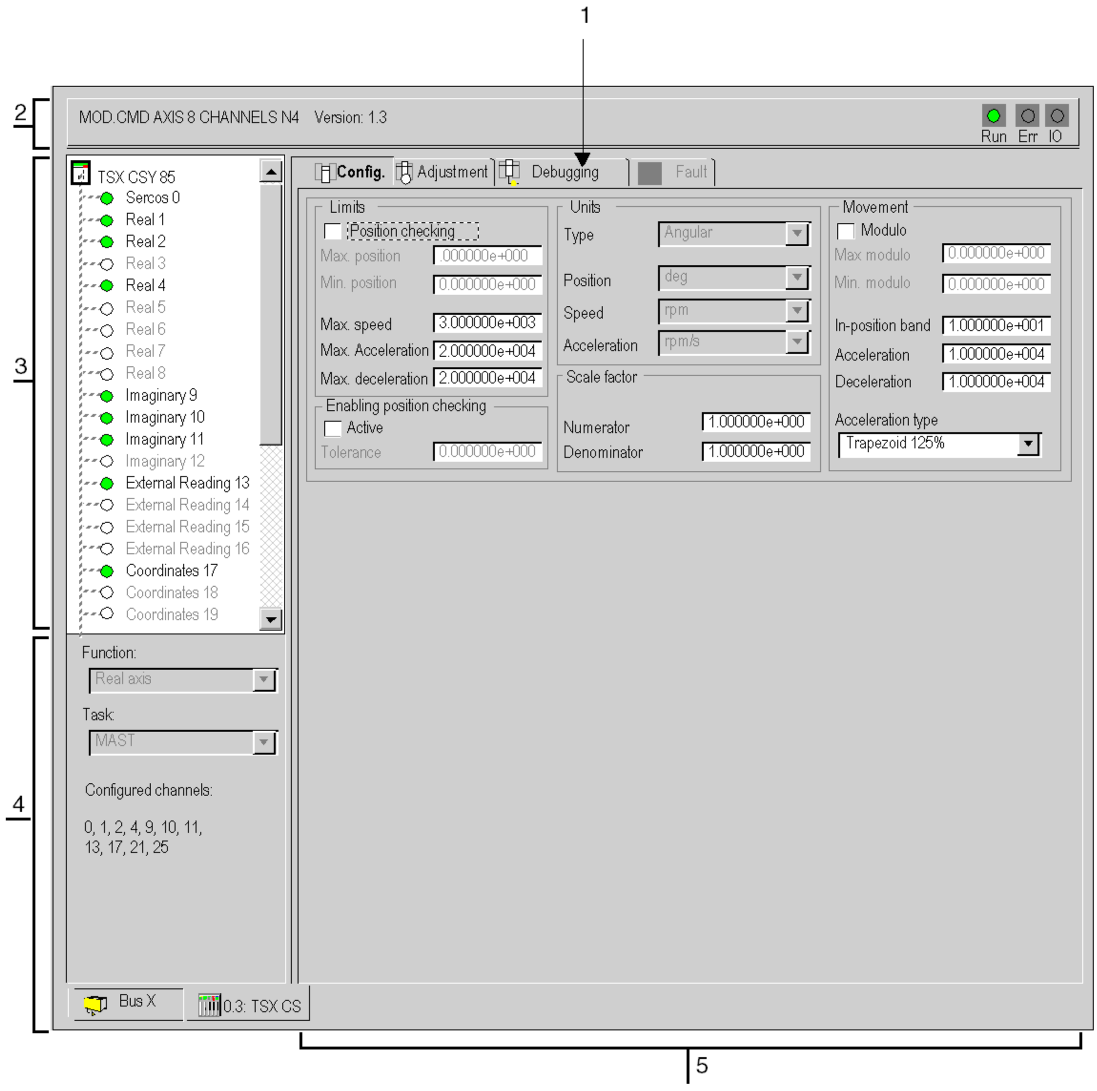
First configure the individual axes, two or three real axes and one axis that will then be used as the master of the real axes to be configured subsequently as slave axes.

In order to be able to use the new interpolation functions described in the rest of this documentation, you must observe the following rules:

- Only the following measurement units (for real axes) may be used:
  - Type = Linear
  - Position = mm
  - Speed = mm/s
  - Acceleration = mm/s$^2$
- Limit parameters **Max. speed** , **Max. acceleration** and **Max. deceleration** in the **Limits** zone are used by the interpolation algorithm in order to calculate the possible speeds on each segment of the trajectory.
- Use these values to set the parameters in the **Movement** zone, which include:
  - **Max acceleration** = **Acceleration**.
  - **Max deceleration** = **Deceleration**.
- The following rules govern the configuration of the axis, which will subsequently be the trajectory master:
  - **Max. speed** = vectorial sum of the maximum speeds of each real axis on the trajectory (slave axes).
  - **Acceleration and Deceleration** = Minimum values for acceleration and deceleration of the real axes on the trajectory

## Configuration Screen

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



Function: Real axis

Task: MAST

Configured channels:

0, 1, 2, 4, 9, 10, 11,
13, 17, 21, 25

Limits
- [ ] Position checking
- Max. position .000000e+000
- Min. position 0.000000e+000
- Max. speed 3.000000e+003
- Max. Acceleration 2.000000e+004
- Max. deceleration 2.000000e+004

Enabling position checking
- [ ] Active
- Tolerance 0.000000e+000

Units
- Type Angular
- Position deg
- Speed rpm
- Acceleration rpm/s

Scale factor
- Numerator 1.000000e+000
- Denominator 1.000000e+000

Movement
- [ ] Modulo
- Max modulo 0.000000e+000
- Min. modulo 0.000000e+000
- In-position band 1.000000e+001
- Acceleration 1.000000e+004
- Deceleration 1.000000e+004
- Acceleration type Trapezoid 125%

The table below presents the different elements of the Configuration screen.

| Address | Element | Function |
|---|---|---|
| 1 | Tabs | The tab in the foreground indicates the current mode (**Debug** in this example). Each mode can be selected using the corresponding tab. Available modes are:<br>● **Debug**, accessible only in online mode,<br>● **Diagnostic** (Default), accessible only in online mode,<br>● **Adjustment**,<br>● **Configuration**. |
| 2 | **Module** zone | Displays the short title of the module.<br>In the same zone there are 3 lamps showing the status of the module in online mode:<br>● **RUN** shows the operating mode of the module.<br>● **ERR** indicates an internal module error.<br>● **I/O** indicates a fault outside the module or an application fault. |
| 3 | **Channel** zone | Allows you to:<br>● display the tabs, by clicking on the device reference:<br>❍ **Description**: provides the device characteristics.<br>❍ **I/O objects** *(see EcoStruxure™ Control Expert, Operating Modes)*: provides a preview symbol of the input/output objects.<br>❍ **Default**: gives access to the device defaults (in online mode).<br>● choose the channel,<br>● displays the **Symbol**, channel name defined by the user (via the variables editor). |
| 4 | **General parameters** zone | Used to unforce bits and view the counting function:<br>● **Unforce:** button used to unforce forced bits.<br>● **Function:** recalls the configured counting function. This information cannot be modified.<br>● **Task:** displays the **MAST** or **FAST** task configured. This information cannot be modified. |
| 5 | **Current parameters** zone | This zone displays the I/O status and the different parameters for the count in progress. If the contents of the counting register cannot be used following an error on the inputs, the message or lamp **Invalid measurement** appears in red. |

The remainder of this chapter describes the parameters of the **current parameters** zone

## 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.<br>For an infinite axis, this box should not be checked. |
| Max. position | Maximum position limit. This value is entered as a floating point. |
| Min. position | Minimum position limit. This value is entered as a floating point. |
| Max. speed | Maximum permissible speed. This value is independent of the value defined in the servodrive (real axis). This value is entered as a floating point. Setting the maximum speed to 0 disables speed monitoring. |
| Max. acceleration | Maximum permissible acceleration. This value is independent of the value defined in the servodrive (real axis). This value is entered as a floating point. |
| Max. deceleration | Maximum permissible deceleration. This value is independent of the value defined in the servodrive (real axis). This value is entered as a floating point. |

## Enabling Position Checking Zone Parameters

### Description

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

## Units Zone Parameters

### Description

| Parameters | Description |
|---|---|
| Type | Type of physical units used to express the position, speed and acceleration readings: Angular, Linear, Linear English or Encoder points. |
| Position | Position unit<br>● Angular: mrad, rad, deg, arcmin, revs<br>● Linear: µm, mm, cm, m<br>● Linear English: in, ft, yd, mile<br>● Encoder points: points. |

| Parameters | Description |
|---|---|
| Speed | Speed unit.<br>● Angular: mrad/s, rad/s, rad/min, deg/s, deg/min, arcmin/s, revs/s, revs/min<br>● Linear: µm/s, mm/s, mm/min, cm/s, cm/min, m/s, m/min<br>● Linear English: in/s, in/min, ft/s, ft/min, yd/min, mile/s<br>● Encoder points: points/ms, points/s, points/min |
| Acceleration | Acceleration unit.<br>● Angular: mrad/$s^2$, rad/$s^2$, deg/$s^2$, arcmin/$s^2$, revs/$s^2$, revs/min/s<br>● Linear: µm/$s^2$, mm/$s^2$, cm/$s^2$, m/$s^2$, m/$min^2$, g's<br>● Linear English: in/$s^2$, ft/$s^2$, yd/$min^2$, mil/$s^2$<br>● Encoder points: points/$ms^2$, points/$s^2$ |

## Scale Factor Zone Parameters

### Description

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

## Movement Zone Parameters

### Description

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

## Rescaling

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

For example, if the axis is configured as angular axis with a scale factor of 1/1:
- Position unit in revolutions and speed unit in rev/s: An incremental movement of position 1 and speed 1 will execute 1 revolution in 1 second.
- Position unit in degrees and speed unit in rev/min: An incremental movement of position 360 and speed 60 will execute 1 revolution in 1 second.

## Changing the Unit Type

When the type of unit is changed in relation to the servodrive, which remains in revolutions, the reference unit of the module is mm (linear type) and inches (linear English type). The transformation achieved by the module is equal to:
- 1 revolution for the servodrive = 1 mm for the module (linear type)
- 1 revolution for the servodrive = 1 inch for the module (linear English type)

For example, if the axis is configured as linear with a scale factor of 1/1:
- Position unit in mm and speed unit in mm/s: An incremental movement of position 1 and speed 1 will execute 1 revolution in 1 second.
- Position unit in mm and speed unit in mm/s: An incremental movement of position 1000 and speed 1000 will execute 1 revolution in 1 second (or 1000 mm in 1 second).

## Using the Scale Factor

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

## Cam Profile Configuration

### Introduction

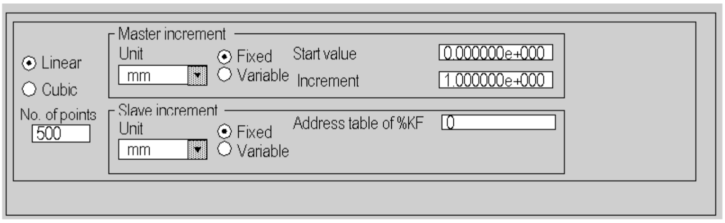
A cam profile (channels 25 to 31) can be used to define the position of a slave axis according to the position of the master axis using a table of points. In order to use the interpolation functions of the TSX CSY 85 module, you must define a cam profile based on a real axis on the trajectory and observe the following rules:

- Interpolation type = **Linear**
- **No. of points** = Number of points of the trajectory on the axis concerned + 5
- Master increment:
  - **Unit = mm**
  - Increment **Fixed**
  - Start value = 0.0
  - Increment = 1.0
- Slave increment:
  - **Unit = mm**
  - Increment **Fixed**
  - Start value = 0.0
  - Increment = 1.0

**NOTE:** The number of points in cam profiles for axes in the same group must be identical. If this number is too small, the module will send back error code 5. The total number of points configured for the cams affect the interpolation calculation time (approximately 1 second for every 100 points configured). The maximum number of points for a TSX CSY 85 module is 10000. We therefore recommend the use of a minimum number of points in order to achieve a trajectory with sufficient precision.

## Configuration Example

The cam profile configuration screen on the TSX CSY 85 module has 3 zones, which must be configured as described above and can be used to define the table of points for the master and slave.


[✓ Linear] Cubic
No. of points: 500

Master increment
Unit: mm
[✓ Fixed] Variable
Start value: 0.000000e+000
Increment: 1.000000e+000

Slave increment
Unit: mm
[✓ Fixed] Variable
Address table of %KF: 0

Table parameters:

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

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 cam profile points is always the same. This button operates alternately with the Variable button. |
| Start value | For a fixed increment, this field can be used to enter the start value for the cam profile. This value is entered as a floating point. |
| Increment | For a fixed increment, this field can be used to define the value of the increment. This value is entered as a floating point. |
| Variable | When this button is selected, the increment between 2 consecutive cam profile points is variable. The value of the points is defined by a table of constant words %KF, whose length is equal to the number of points. |
| `%KF` table address | For a variable increment, this field can be used to enter the start address for the table of points for the master. |

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

The table will completely describe the modulo and an additional point will be added (modulo + 1) for which the slave value is the same as the first value in the table.
For example, for a modulo of 360°, the values are 0 to 359, and the table is as follows: Table (master, slave): (0, **x0**); (1, x1); (2, x2); ...; (359, x359); (360, **x0**)

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

**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 cam profile points is always the same. This button operates alternately with the Variable button. |
| Start value | For a fixed increment, this field can be used to enter the start value for the cam profile. This value is entered as a floating point. |
| Increment | For a fixed increment, this field can be used to define the value of the increment. This value is entered as a floating point. |
| Variable | When this button is selected, the increment between 2 consecutive cam profile points is variable. The value of the points is defined by a table of constant words %KF, whose length is equal to the number of points. |
| `%KF` table address | For a variable increment, this field can be used to enter the start address of the table of points for the slave. |

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

### Introduction

A slave axis group is a collection of axes, composed of slave axes (a maximum of 6), which follow the movements of a master axis.

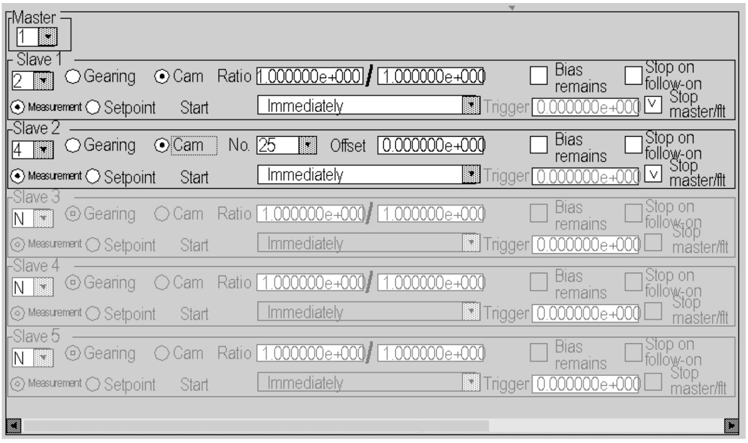
The master axis may be a real axis, an imaginary axis or an external setpoint. The slave axes are real or imaginary axes.

The rules to be observed for a TSX CSY 85 module using the new interpolation functions are as follows:

- Each group may comprise 2 or 3 interpolated axes.
- A cam profile must be defined for each slave axis (maximum of 7).
- As a consequence of the above, up to 7 interpolated axes are possible (for example, 2 groups of 2 axes and one group of 3 axes).
- Only two axes may be interpolated. The third axis in a group MUST be linked to the master either via a cam profile or via a fixed link.
- Each axis must feature the following parameters:
  - Select the **Cam** option.
  - **Start = Immediate**
  - **Offset = 0.0**
  - Select the **Setpoint** option.
  - Select the **Stop master/flt** option.

### Configuration Screen

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


Master: 1

Slave 1: 2 — ○ Gearing [✓ Cam] — Ratio 1.000000e+000 / 1.000000e+000 — ☐ Bias remains — ☐ Stop on follow-on
[✓ Measurement] ○ Setpoint — Start: Immediately — Trigger 0.000000e+000 — [✓ Stop master/flt]

Slave 2: 4 — ○ Gearing [✓ Cam] — No. 25 — Offset 0.000000e+000 — ☐ Bias remains — ☐ Stop on follow-on
[✓ Measurement] ○ Setpoint — Start: Immediately — Trigger 0.000000e+000 — [✓ Stop master/flt]

Slave 3: N — Gearing ○ Cam — Ratio 1.000000e+000 / 1.000000e+000 — ☐ Bias remains — ☐ Stop on follow-on
Measurement ○ Setpoint — Start: Immediately — Trigger 0.000000e+000 — ☐ Stop master/flt

Slave 4: N — Gearing ○ Cam — Ratio 1.000000e+000 / 1.000000e+000 — ☐ Bias remains — ☐ Stop on follow-on
Measurement ○ Setpoint — Start: Immediately — Trigger 0.000000e+000 — ☐ Stop master/flt

Slave 5: N — Gearing ○ Cam — Ratio 1.000000e+000 / 1.000000e+000 — ☐ Bias remains — ☐ Stop on follow-on
Measurement ○ Setpoint — Start: Immediately — Trigger 0.000000e+000 — ☐ Stop master/flt

## Master

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

## Slave Zones

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

## Slave Zone Parameters

Description

| Parameter | Description |
|---|---|
| Slave 1 (to 6) | Used to set the number of the slave axis (axes 1 to 12). |
| Measurement | When this button is checked, the slave axis follows the measured position of the master axis. This button operates alternately with the Setpoint button. |
| Setpoint | When this button is checked, the slave axis follows the position of the master axis setpoint. This button operates alternately with the Measurement button. |
| Gearing | When the button is checked, the slave axis follows the master axis in Ratio mode; that is to say following a ratio determined by the Ratio field. This button operates alternately with the Cam button. |
| Cam | When this button is checked, the slave axis follows the master axis in Cam mode; that is to say following the cam profile whose number is selected in the No. field. This button operates alternately with the Gearing button. |
| Ratio | In Ratio mode, these 2 fields are used to input the numerator and the denominator, which define the ratio between the master axis and the slave. These values are entered as floating points. |
| Start | Used to select the start condition:<br>● Immediate<br>● When the master position, increased by the offset value, reaches the threshold value defined in the Trigger field (CW travel)<br>● When the master position plus the offset value, reaches the threshold value defined in the Trigger field (CCW direction)<br>● When the master position plus the offset value is greater than or equal to the threshold value defined in the Trigger field<br>● When the master position plus the offset value is less than or equal to the threshold value defined in the Trigger field |
| No. | In Cam mode, this field is used to select the cam profile number (between 25 and 31). |

| 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.<br>The resulting position of the slave will be given by the index in the cam profile master table. This index is equal to the current position of the master + offset (for example, a cam profile defined from 0 to 1000 for the master coordinates). To start following a master position equal to 100000, the offset value should be equal to -100000.0.<br>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:<br>● A dynamic offset is added automatically to the master position in order to define the position of the slave.<br>● The additional slave movements are not halted when the link with the master is removed. |
| Trigger | When the start condition is dependent on the position of the master axis in relation to a threshold, this field is used to input the threshold value. This value is entered as a floating point.<br>Activation will take place when the current position of the master + offset > (or <, >, <) the value of the threshold (Trigger). |
| Stop on follow-on | When this box is checked, confirming the link between the master and the slave will stop any possible additional movement of the slave axis, following a deceleration profile determined automatically. |
| Stop master/flt | When this box is checked, the master stops when there is a contour discrepancy (fault) between the master and the slave. |

## Position of a Slave Axis

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

## Scale Factor

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

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

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

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

# Chapter 4
# Interpolation Functions

## Aim of This Chapter

This chapter describes the new interpolation functions supported by the TSX CSY 85 module.

## What Is in This Chapter?

This chapter contains the following sections:

## Section 4.1
## Trajectory Calculation Function: TrjCompute (ACTION_TRF (26900)

### Aim of this Section

This section introduces the trajectory calculation function `TrjCompute`, obtained via code ACTION_TRF = 26900, which is uploaded to the module by means of instruction TRF_RECIPE, along with the various types of segment that can be created.

### What Is in This Section?

This section contains the following topics:

## TrjCompute: Trajectory Calculation Function (ACTION_TRF = 26900)

### At a Glance

The `TrjCompute` function can be used to upload trajectories to the TSX CSY 85 module.

It relies on the following elements for its execution:

- The default instruction to transfer information to the module TRF_RECIPE *(see page 28)*
- The action code ACTION_TRF = 26900 (`%MWr.m.c.10`) and
- The table of parameters defining the trajectory

**NOTE:** The parameter table can be created either directly by entering the parameters into the word table, or by importing this table using the graphics-based **TjE** (Trajectory Editor) software.

The following pages indicate the syntax to be used, followed by a description of all interpolations supported by this function and their codes.

**⚠ WARNING**

**UNEXPECTED APPLICATION BEHAVIOR**

You must reset the `ALLOW_ACQUIRE` bit of the group of slave axes before launching the `TrjCompute` function.

You can then revalidate the group by positioning the `ALLOW_ACQUIRE` and `CONTROL_ACQUIRE` bits to 1.

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

### Function Syntax

The `TrjCompute` function is executed:

- On a slave axis group type channel (channels 21 to 24)
- With the following parameter settings:
  - `%MWr.m.c.10` (ACTION_TRF) = 26900
  - `%MDr.m.c.11` (param_trf_1) = 0,
  - `%MDr.m.c.13` (param_trf_2) = 0,
  - `%MDr.m.c.15` (param_trf_3) = 0,
  - `%MDr.m.c.17` (param_trf_4) = 0,
- and the TRF_RECIPE function, whose parameters are taken from the word table, which contains the description of the trajectory as well as the length of this table.

TRF_RECIPE `%CHr.m.c` (table_length, trajectory_table)

| Parameter | Description |
|---|---|
| `%CHr.m.c` | Channel associated with module channel group consisting of the slave axes and master axis of the trajectory to be calculated. |
| Table_length | Number of words in the trajectory table. The calculation formula is as follows:<br>Table_length = 6 + number of axes + number of words per point * number of points:<br>● **Number of axes**: Number of slave axes 2 or 3.<br>● **Number of words per point**: 17 for 2 axes and 19 for 3 axes.<br>● **Number of points**: Number of reference points characterizing the movement. |
| Trajectory_table | First word in the table containing the parameters of the trajectory, this table contains a header and as many blocks as there are trajectory segments required to complete this movement. This table is described in detail below. |

**Example**: `TRF_RECIPE %CH3.21 (93,100);` for movement on 2 axes, with 5 characteristic points and one table located in `%MW100`.

## Feedback Code

Feedback codes for the TRF RECIPE function following the trajectory calculation 26900 are available in words `%MDWxy.i.4` and `%Mfxy.i.6` indicating:

- · Return_trf_1: the number of useful cam points to which you must add 5 for the configuration of the CSY85 in PREMIUM
- · Return_trf_2: the maximum distance of the master to cover the entire trajectory.

The trajectory calculation time may require several seconds, according to the complexity of the described trajectory. Example: For 35 master points, the calculation time is approximately 3 seconds.

Data returned by action code ACTION_TRF = 26900 (`%MWr.m.c.10`)

| Object | Type | Symbol | Description |
|---|---|---|---|
| `%MDr.m.c.4` | Integer | RETURN_TRF_1 | Number of points in the trajectory |
| `%MFr.m.c.6` | Floating point | RETURN_TRF_2 | Length of the trajectory |

## Trajectory Table Header

The table below describes the content of the header of the trajectory table starting at address `%MWi`:

| Memory address | Description |
|---|---|
| `%MWi` | Number of points to be interpolated |
| `%MWi+1` | Number of slave axes used for the trajectory |
| `%MWi+2` | Memory shift for each interpolation point:<br>● 17 with 2 slave axes<br>● 19 with 3 slave axes |
| `%MWi+3` | Interpolation table version (enter the value 0 for now) |
| `%MWi+4` | Reserved |
| `%MWi+5` | Reserved |
| `%MWi+6` | Identifier (ID) of X axis |
| `%MWi+7` | Identifier (ID) of Y axis |
| `%MWi+8` | Identifier (ID) of Z axis<br>**Note**: If the movement only uses two axes, this word does not exist and the table header only comprises 8 words. |

**NOTE:** Do not confuse the number of the channel associated with the axis with the axis ID. You can use the `GetAxisID` function (code 523) to obtain the ID of an axis (use the `WRITE_CMD` instruction).

## Trajectory Table Body

The table body contains a series of blocks, which characterize each segment of the trajectory. Each block contains an identical number of words, which contain the interpolation parameters for each segment.

The table below describes a typical segment of a trajectory table, where the table header starts at address `%MWi`:

| Memory address | Parameter | Description |
|---|---|---|
| `%MFj` (1) | XCoord | Coordinate based on the X axis of the characteristic point of the segment. |
| `%MFj+2` | YCoord | Coordinate based on the Y axis of the characteristic point of the segment. |
| `%MFj+4` | ZCoord | Coordinate based on the Z axis of the characteristic point of the segment.<br>**Note**: If the movement only uses two axes, this parameter does not exist and the block associated with this segment only contains 17 words instead of 19. |
| `%MFj+6` | ParF0 | Vset: Setpoint speed of this segment. |
| `%MFj+8` | ParF1 | Parameter ParF1, determined by the type of interpolation. |

| Memory address | Parameter | Description |
|---|---|---|
| `%MFj+10` | ParF2 | Parameter ParF2, determined by the type of interpolation. |
| `%MFj+12` | ParF3 | Parameter ParF3, determined by the type of interpolation. |
| `%MWj+14` | ParW0 | Segment interpolation type:<br>● 0: Linear<br>● 1: Linear with 3º polynomial interpolation connection<br>● 2: Linear with link using 5º polynomial interpolation connection<br>● 10: Linear with circular interpolation connection<br>● 11: Circular via calculation of circle radius<br>● 12: Circular via calculation of circle center<br>● 100: Only for point P0, to indicate that the movement is along a tangent axis<br>● 101: Only for point P0, to indicate that the movement is along a tangent axis but shifted by 180° in relation to code 100 (other side of the curve). |
| `%MWj+15` | ParW1 | Parameter ParW1, determined by the type of interpolation. |
| `%MWj+16` | ParW2 | Parameter ParW2, determined by the type of interpolation. |
| `%MWj+17` | ParW3 | Parameter ParW3, determined by the type of interpolation. |
| `%MWj+18` | ParW4 | Parameter ParW4, determined by the type of interpolation. |
| | | |
| Legend | | |
| (1) j = 6 + number of axes, j = 9 if the trajectory is configured on 3 axes, j = 8 if it is configured on 2 axes. | | |

**NOTE:** Some or all of these different parameters are used depending on the type of interpolation selected for each segment. For a complete trajectory table, the blocks characterizing each segment must be positioned consecutively one after the other.

**NOTE:** The Characteristic Point of a segment (XCoord, YCoord, ZCoord) is the point reached by the segment. The P0 point is the start point of the trajectory and is required in specific cases, such as for a closed trajectory, in order to be able to calculate the trajectory length.

## Error Codes

When this function is used, the error codes common to all types of interpolation, which can be returned in word `%MWr.m.c.3`, are as follows:

| Code | Description |
|---|---|
| 9502 | The maximum number of points for a cam has been exceeded. |
| 9503 | The maximum number of axes has been exceeded. |
| 9505 | The point number configured for one of the cams is not sufficient to execute the function. |
| 9507 | Cam not configured. |
| 9510 | The maximum number of points for a table has been exceeded. |

## GetInterpoErrorPoint: Error Point Detection Function (ACTION_TRF = 14902)

### At a Glance

This function is very useful when debugging the trajectory. It provides the number of the point that caused the error following a trajectory calculation using the `TrjCompute` function.

### Function Syntax

The interface is as follows:

| Memory address | Parameter | Description |
|---|---|---|
| `%MWr.m.c.10` | ACTION_TRF | 14902 |
| `%MDr.m.c.11` | param_trf_1 | not used |
| `%MDr.m.c.13` | param_trf_2 | not used |
| `%MFr.m.c.15` | param_trf_3 | not used |
| `%MFr.m.c.17` | param_trf_4 | not used |
| `%MDr.m.c.6` | RETURN_TRF_1 | number of the point that caused the error |

**Example**:

```
TRF_RECIPE %CH3.21 (0,0);
IF %M4 AND NOT %MW3.21:X3 THEN
%MW3.21.10:=14902;
TRF_RECIPE %CH3.21(0,0);
RESET %M4;
END_IF;
```

This function will mainly be used in case of an error, when modifying a point on the trajectory via the PLC's application program.

## Linear Interpolation: Type 0

### At a Glance

This type of interpolation can be used to generate a rectilinear trajectory between the previous point and the point defining the segment.

The major problem with this type of interpolation lies in the fact that a series of segments of this type may define broken lines and therefore impose mechanical and electrical restrictions on the devices being controlled (sudden variations in speed, for example). The only way to reduce these restrictions is to reduce the speed of movement or to stop at each point in the table. However, this type of segment may be suitable for transitional movements, for repositioning or in the event of maintenance work.

### Block Parameters

The table below indicates the parameters to be entered in the trajectory table for this type of segment:

| Parameter | Description |
|---|---|
| ParW0 | Enter 0 to indicate that the type of interpolation to be used is linear interpolation. |
| ParW1 | Indicate the number of points in the linear segment in this parameter. This number of points (minimum 1) represents the number of intermediate points that the module has to calculate in order to define the trajectory on the segment. The TSX CSY 85 module defines these points then calculates a linear interpolation between them in order to determine the position that will be sent to the drives every 4 ms.<br>With linear interpolation, simply put the minimum value: 1. |
| ParW4 | For this type of segment, only tangent mode can be used, but only if a third axis is being used for a 3-dimensional movement.<br>**Note**: Tangent mode (bit 8 at 1) will be available in a later version of the TjE/TSX CSY 85 combination, but for the moment only angular positioning of the third Z axis is supported.<br>For a movement on two axes, enter the value 0. |

### Error Codes

When this type of segment is used, the error codes that can be returned in word `%MDr.m.c.3` are as follows:

| Code | Description |
|---|---|
| 9504 | The table contains two consecutive identical points. |
| 9515 | The number of points in the segment is set to 0. |

## Linear Interpolation With 3º Polynomial Interpolation Connection: Type 1

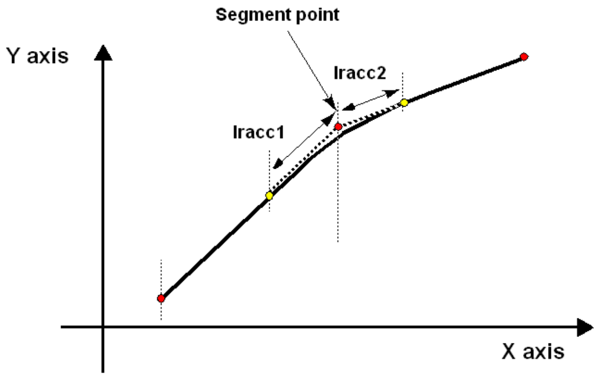
### At a Glance

This type of interpolation can be used to link linear segments by smoothing transitions using 3º interpolation.

The trajectory does not pass through the point defined in the table but follows a curve defined by the parameters. The smaller the connection zone, the closer the trajectory will be to the point, although the larger the curve will be, thus reducing the maximum possible speed.

### Illustration

The diagram below illustrates an example of a type 1 segment:



### Block Parameters

The table below indicates the parameters to be entered in the trajectory table for this type of segment:

| Parameter | Description |
|---|---|
| ParW0 | Enter 1 to indicate that the type of interpolation to be used is linear interpolation with 3° polynomial interpolation connection. |
| ParW1 = `NpLin` | Indicate the number of points in the linear part of the segment in this parameter. This number of points (minimum 1) represents the number of intermediate points that the module has to calculate in order to define the trajectory on this linear part. |
| ParW2 = `NpRacc` | Indicate the number of points in the cubic interpolation part of the segment in this parameter. This number of points (minimum 1) represents the number of intermediate points that the module has to calculate in order to define the trajectory on this part. It should be enough to ensure precision of the trajectory, and a minimum value of 15 is recommended. |

| Parameter | Description |
|---|---|
| ParW3 = KF | 3° interpolation shape coefficient. Between 50 and 200 this coefficient can be used to modify the shape of the connection zone. The default value is 100, and it is advisable to keep this default value if the desired movement allows, as it ensures that there will not be any motion "overshoot".<br>**Note**: The more the KF coefficient increases, the nearer the curve comes to the segment characteristic point, and the more it decreases, the further away the curve goes. |
| ParW4 | For this type of segment, only tangent mode can be used. If a third tangent type axis is being used, select tangent mode (bit 8 of ParW4 at 1) to indicate that the motion of the third axis will follow the curve using tangent mode (positioning a tool for example so that it is always at right-angles to the curve and follows its tangent).<br>**Note**: Tangent mode (bit 8 at 1) will be available in a later version of the TjE/TSX CSY 85 combination, but for the moment only angular positioning of the third Z axis is supported.<br>For a movement on two axes, enter the value 0. |
| ParF1 = lracc1<br>ParF2 = lracc2 | lracc1 is the length of the initial connection.<br>lracc2 is the length of the final connection.<br>lracc1 and lracc2, expressed in mm, determine the length of the interpolation zone.<br>If these lengths are too long (for example lracc1 > distance between the current point (Pn) and the previous point (Pn-1)), the maximum possible length should be used. This maximum length is automatically calculated by the module according to the previous section for lracc1 and the next section for lracc2.<br>**Example**: If the sum of lracc2 of Pn-1 and lracc1 of Pn is greater than the distance between Pn-1 and Pn, the linear part of the trajectory will resume at a reversal point on the trajectory between these two points.<br>In addition if lracc1 > 2*lracc2 then interpolation considers that lracc1=2*lracc2<br>Similarly if lracc2 > 2*lracc1 then interpolation considers that lracc2=2*lracc1 |

## Error Codes

When this type of segment is used, the error codes that can be returned in word %MWr.m.c.3 are as follows:

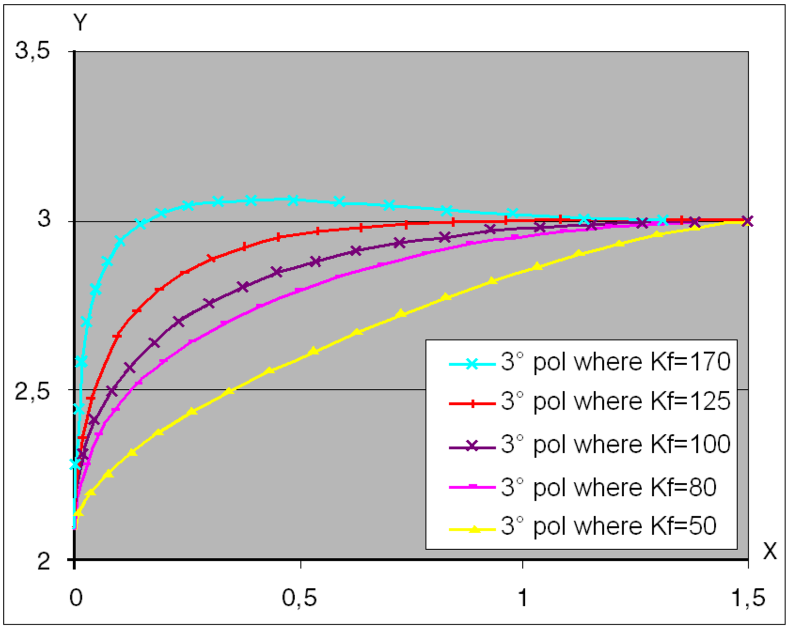
| Code | Description |
|---|---|
| 9501 | One of the lengths lracc1 or lracc2 equals zero (illegal scenario). |
| 9504 | The table contains two consecutive identical points. |
| 9514 | Link too long: Next segment = 0 |
| 9515 | The number of points in the linear segment is set to 0. |
| 9516 | The number of points in the 3º polynomial interpolation segment is set to 0. |

**NOTE:** The error codes indicated are those for the TSX CSY 85 module. The error codes for **TjE** are the same but without the 9 (9501 corresponds to 501 etc).

## Shape Coefficient KF

The diagram below illustrates how the shape of the trajectory changes depending on the value of the shape coefficient KF (between 50 and 200):

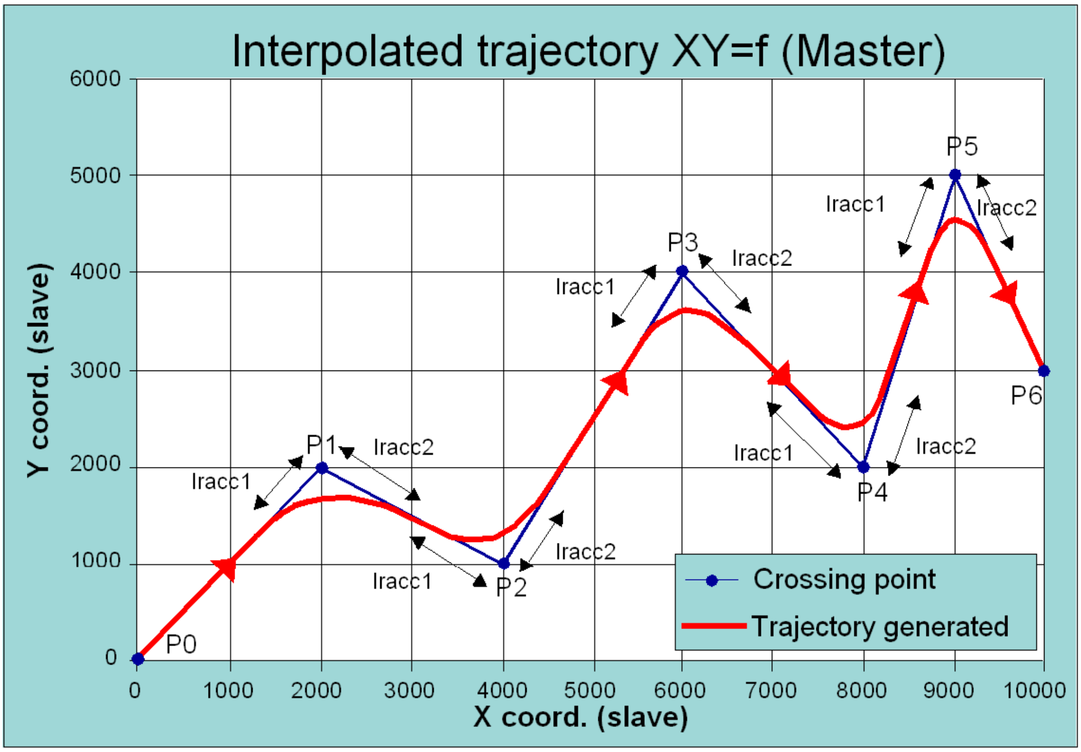
Master trajectory curve as a function of KF



**NOTE:** The more the `KF` coefficient increases, the nearer the curve comes to the segment characteristic point, and the more it decreases, the further away the curve goes.

## Example Trajectory

The diagram below illustrates an example of a trajectory using type 1 segments:



## Linear Interpolation With 5º Polynomial Interpolation Connection: Type 2

### At a Glance

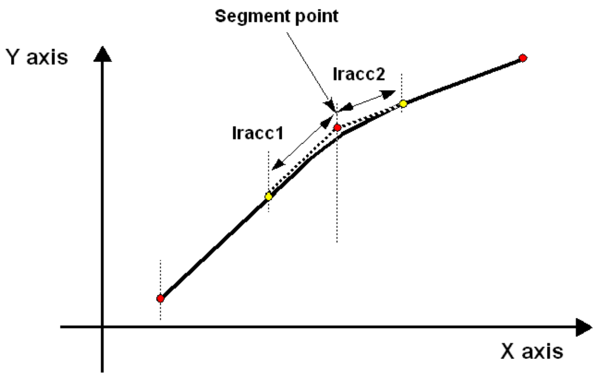
This type of interpolation can be used to link linear segments by smoothing transitions using 5º interpolation.

The trajectory does not pass through the point defined in the table but follows a curve defined by the parameters. The smaller the connection zone, the closer the trajectory will be to the point, although the larger the curve will be, thus reducing the maximum possible speed.

Compared with 3º interpolation, this type of interpolation makes movement more flexible, although if the acceleration limit is reached, the speed on the segment may be restricted on this type of connection.

### Illustration

The diagram below illustrates an example of a type 2 segment:



## Block Parameters

The table below indicates the parameters to be entered in the trajectory table for this type of segment:

| Parameter | Description |
|---|---|
| ParW0 | Enter 2 to indicate that the type of interpolation to be used is linear interpolation with 5° polynomial interpolation connection. |
| ParW1 = `NpLin` | Indicate the number of points in the linear part of the segment in this parameter. This number of points (minimum 1) represents the number of intermediate points that the module has to calculate in order to define the trajectory on this linear part. |
| ParW2 = `NpRacc` | Indicate the number of points in the 5° interpolation part of the segment in this parameter. This number of points (minimum 1) represents the number of intermediate points that the module has to calculate in order to define the trajectory on this part. It should be enough to ensure precision of the trajectory, and a minimum value of 15 is recommended. |
| ParW3 = `KF` | 5° interpolation shape coefficient. Between 50 and 200 this coefficient can be used to modify the shape of the connection zone. The default value is 100, and it is advisable to keep this default value if the desired movement allows, as it ensures that there will not be any motion "overshoot".<br>**Note**: The more the `KF` coefficient increases, the nearer the curve comes to the segment characteristic point, and the more it decreases, the further away the curve goes. |
| ParW4 | For this type of segment, only tangent mode can be used. If a third tangent type axis is being used, select tangent mode (bit 8 of ParW4 at 1) to indicate that the motion of the third axis will follow the curve using tangent mode (positioning a tool for example so that it is always at right-angles to the curve and follows its tangent).<br>**Note**: Tangent mode (bit 8 at 1) will be available in a later version of the TjE/TSX CSY 85 combination, but for the moment only angular positioning of the third Z axis is supported.<br>For a movement on two axes, enter the value 0. |
| ParF1 = `lracc1`<br>ParF2 = `lracc2` | `lracc1` is the length of the initial connection.<br>`lracc2` is the length of the final connection.<br>`lracc1` and `lracc2`, expressed in mm, determine the length of the interpolation zone.<br>If these lengths are too long (for example `lracc1` > distance between the current point (Pn) and the previous point (Pn-1)), the maximum possible length should be used. This maximum length is automatically calculated by the module according to the previous section for `lracc1` and the next section for `lracc2`.<br>**Example**: If the sum of `lracc2` of Pn-1 and `lracc1` of Pn is greater than the distance between Pn-1 and Pn, the linear part of the trajectory will resume at a reversal point on the trajectory between these two points. Hence the curve will be continuous, as will its differential coefficient.<br>In addition if `lracc1` > `lracc2`/0.66 then interpolation considers that `lracc1`=`lracc2`/0.66<br>Similarly if `lracc2` > `lracc1`/0.66 then interpolation considers that `lracc2`=`lracc1`/0.66 |

## Error Codes

When this type of segment is used, the error codes that can be returned in word `%MWr.m.c.3` are as follows:

| Code | Description |
|---|---|
| 9501 | One of the lengths `lracc1` or `lracc2` equals zero (illegal scenario). |
| 9504 | The table contains two consecutive identical points. |
| 9514 | Link too long: Next segment = 0 |
| 9515 | The number of points in the linear segment is set to 0. |
| 9516 | The number of points in the 5º polynomial interpolation segment is set to 0. |

**NOTE:** The error codes indicated are those for the TSX CSY 85 module. The error codes for **TjE** are the same but without the 9 (9501 corresponds to 501 etc).

## Linear Interpolation With Circular Interpolation Connection: Type 10

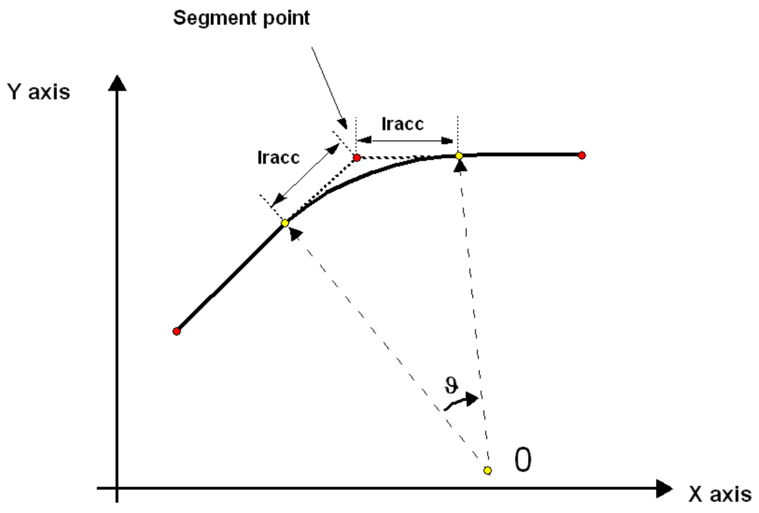
### At a Glance

This type of interpolation can be used to link linear segments via a circular trajectory (arcs of circles or full circles).

**NOTE:** This type of interpolation is only possible if the movement is along one plane (2 axes only).

### Illustration

The diagram below illustrates an example of a type 10 segment:



## Block Parameters

The table below indicates the parameters to be entered in the trajectory table for this type of segment:

| Parameter | Description |
|---|---|
| ParW0 | Enter 10 to indicate that the type of interpolation to be used is linear interpolation with circular interpolation connection. |
| ParW1 = `NpLin` | Indicate the number of points in the linear part of the segment in this parameter. This number of points (minimum 1) represents the number of intermediate points that the module has to calculate in order to define the trajectory on this linear part. |
| ParW2 = `NpRacc` | Indicate the number of points in the circular interpolation part in this parameter. This number of points (minimum 1) represents the number of intermediate points that the module has to calculate in order to define the trajectory on this part. It should be enough to ensure precision of the trajectory, and a minimum value of 15 is recommended. |
| ParW4 | Bit 0 at 0 for a connection making an angle less than 180°.<br>Bit 0 at 1 for a connection making an angle more than 180°.<br>If a third tangent type axis is being used, select tangent mode (bit 8 of ParW4 at 1) to indicate that the motion of the third axis will follow the curve using tangent mode (positioning a tool for example so that it is always at right-angles to the curve and follows its tangent).<br>**Note**: Tangent mode (bit 8 at 1) will be available in a later version of the TjE/TSX CSY 85 combination, but for the moment only angular positioning of the third Z axis is supported. |
| ParF1 = `lracc` | `lracc` is the length of the circular interpolation segment. |

## Error Codes

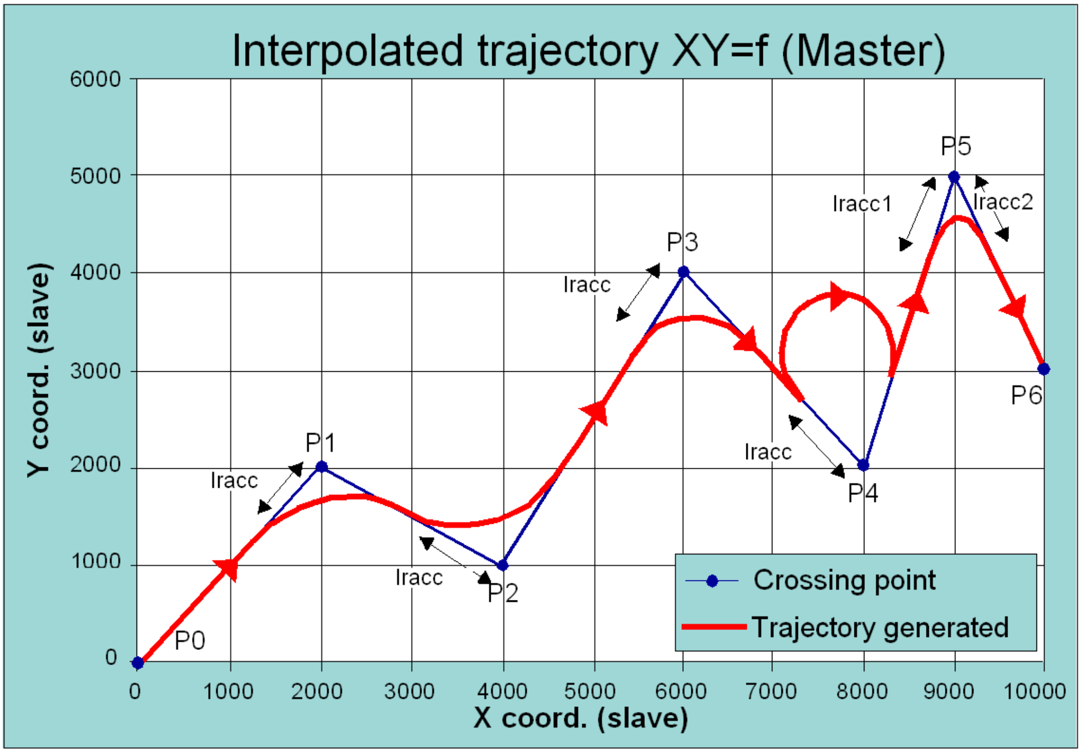
When this type of segment is used, the error codes that can be returned in word `%MWr.m.c.3` are as follows:

| Code | Description |
|---|---|
| 9501 | The length `lracc` equals zero (illegal scenario). |
| 9504 | The table contains two consecutive identical points. |
| 9506 | Use of circular type interpolation although more than two axes have been defined |
| 9508 | Circular connection with angle of 180° |
| 9509 | Circular connection with angle of 0° |
| 9514 | Link too long: Next segment = 0 |
| 9515 | The number of points in the linear segment is set to 0. |
| 9517 | The number of points in the circular interpolation segment is set to 0. |

**NOTE:** The error codes indicated are those for the TSX CSY 85 module. The error codes for **TjE** are the same but without the 9 (9501 corresponds to 501 etc).

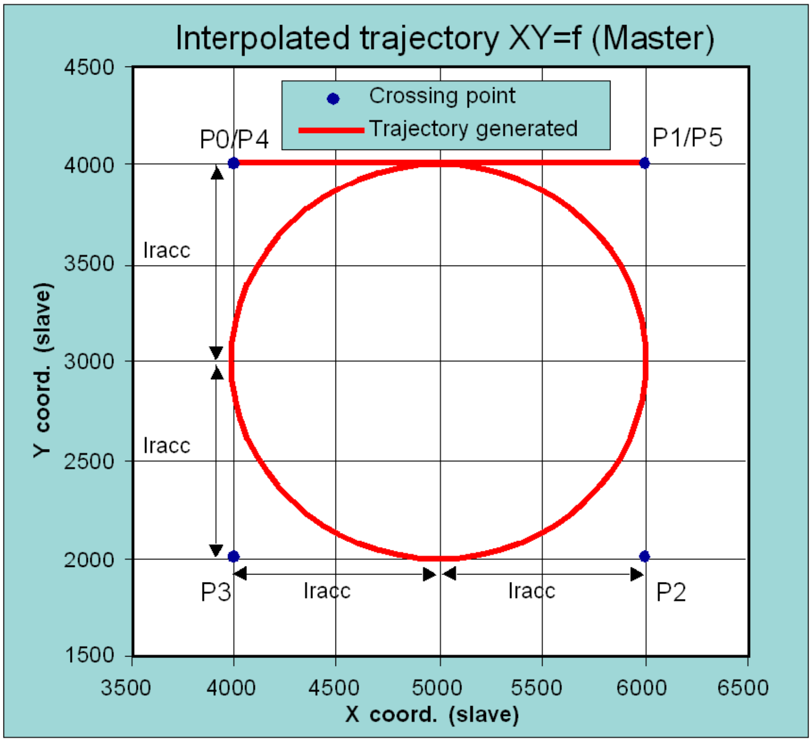
## Example Trajectory

The diagram below illustrates an example of a trajectory using type 10 and type 1 segments. Point 1 provides an example of circular motion with parameter ParW4 =. Note that in this case, the speed is not constant:



## Example of Full Circular Trajectory

The diagram below illustrates an example of a circular trajectory for which the corners of the circumscribed square have been defined with circular segments measuring half of the side of this square in length:



## Circular Interpolation With Designated Radius: Type 11

### At a Glance

This type of interpolation can also be used to link linear segments via a circular trajectory.

You must specify:

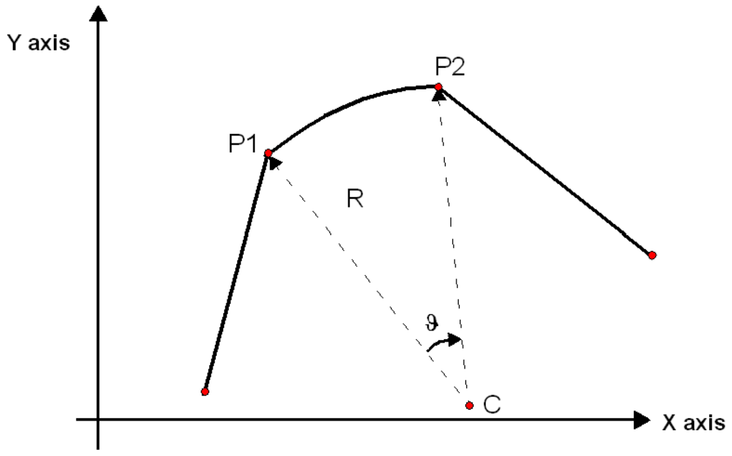
- The start point (Pn-1)
- The end point (Pn)
- The radius of the circle
- The direction of the trajectory (clockwise or counter-clockwise)

**NOTE:** Full circular motion is not supported by this type of interpolation. If you wish to travel a full circle, you must use type 10 interpolation.

**NOTE:** This type of interpolation is only possible if the movement is along one plane (2 axes only).

### Illustration

The diagram below illustrates an example of a type 11 segment:



## Block Parameters

The table below indicates the parameters to be entered in the trajectory table for this type of segment:

| Parameter | Description |
|---|---|
| ParW0 | Enter 11 to indicate that the type of interpolation to be used is linear interpolation with circular interpolation connection for which the radius is given. |
| ParW1 = `NpLin` | Indicate the number of points in the circle arc in this parameter. This number of points (minimum 1) represents the number of intermediate points that the module has to calculate in order to define the trajectory on this segment. |
| ParW4 | Bit 0 at 0 for counter-clockwise trajectory direction.<br>Bit 0 at 1 for clockwise trajectory direction.<br>If a third tangent type axis is being used, select tangent mode (bit 8 of ParW4 at 1) to indicate that the motion of the third axis will follow the curve using tangent mode (positioning a tool for example so that it is always at right-angles to the curve and follows its tangent).<br>**Note**: Tangent mode (bit 8 at 1) will be available in a later version of the TjE/TSX CSY 85 combination, but for the moment only angular positioning of the third Z axis is supported. |
| ParF1 = `R` | `Radius` is the length of the radius of the circle arc. |

## Error Codes

When this type of segment is used, the error codes that can be returned in word `%MWr.m.c.3` are as follows:

| Code | Description |
|---|---|
| 9504 | The table contains two consecutive identical points. |
| 9506 | Use of circular type interpolation although more than two axes have been defined |
| 9511 | The radius is less than half the distance between points Pn-1 and Pn. |
| 9512 | Circle impossible |
| 9513 | Radius equal to 0 |
| 9518 | The number of points in the circular interpolation segment is set to 0. |

**NOTE:** The error codes indicated are those for the TSX CSY 85 module. The error codes for **TjE** are the same but without the 9 (9504 corresponds to 504 etc).

## Circular Interpolation With Designated Center: Type 12

### At a Glance

This type of interpolation can also be used to link linear segments via a circular trajectory.

You must specify:

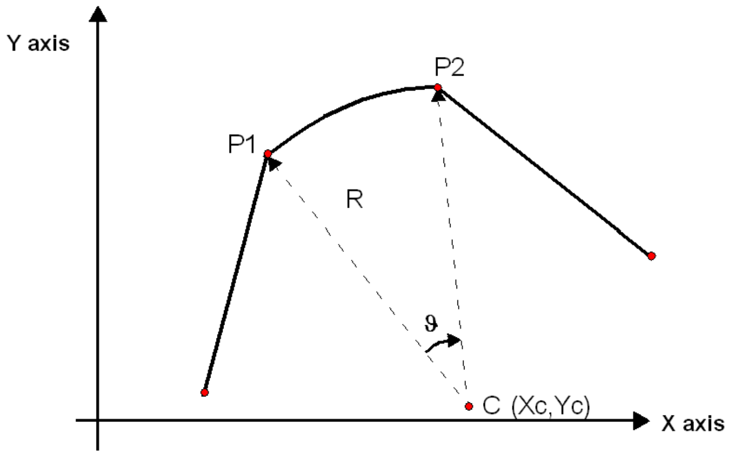
- The start point (Pn-1)
- The end point (Pn)
- The coordinates of the center of the circle
- The direction of the trajectory (clockwise or counter-clockwise)

**NOTE:** Full circular motion is possible if the start point is the same as the end point.

**NOTE:** This type of interpolation is only possible if the movement is along one plane (2 axes only).

### Illustration

The diagram below illustrates an example of a type 12 segment:



## Block Parameters

The table below indicates the parameters to be entered in the trajectory table for this type of segment:

| Parameter | Description |
|---|---|
| ParW0 | Enter 12 to indicate that the type of interpolation to be used is linear interpolation with circular interpolation connection for which the coordinates of the circle center are indicated. |
| ParW1 = `NpLin` | Indicate the number of points in the circle arc in this parameter. This number of points (minimum 1) represents the number of intermediate points that the module has to calculate in order to define the trajectory on this segment. |
| ParW4 | Bit 0 at 0 for counter-clockwise trajectory direction.<br>Bit 0 at 1 for clockwise trajectory direction.<br>If a third tangent type axis is being used, select tangent mode (bit 8 of ParW4 at 1) to indicate that the motion of the third axis will follow the curve using tangent mode (positioning a tool for example so that it is always at right-angles to the curve and follows its tangent).<br>**Note**: Tangent mode (bit 8 at 1) will be available in a later version of the TjE/TSX CSY 85 combination, but for the moment only angular positioning of the third Z axis is supported. |
| ParF1 = `Xc`<br>ParF2 = `Yc` | `Xc` is the position of the center of the circle according to the X axis.<br>`Yc` is the position of the center of the circle according to the Y axis.<br>**Note**: The TSX CSY 85 module automatically recalculates the exact position of the center. However, if the position indicated differs from the exact position by more than half the length of the circle radius error code 9519 is generated. |

## Error Codes

When this type of segment is used, the error codes that can be returned in word `%MWr.m.c.3` are as follows:

| Code | Description |
|---|---|
| 9506 | Use of circular type interpolation although more than two axes have been defined |
| 9512 | Circle impossible |
| 9518 | The number of points in the circular interpolation segment is set to 0. |
| 9519 | The designated center position is outside the tolerance limit permitted by the module. |

**NOTE:** The error codes indicated are those for the TSX CSY 85 module. The error codes for **TjE** are the same but without the 9 (9506 corresponds to 506 etc).

## Tangent Axis Interpolation: Type 100 or 101

### At a Glance

This type of interpolation enables use of a third axis that will automatically follow the curve created using two axes.

This third axis will allow a cutting tool to be positioned in cases where an angle of approach has been defined for the tool. This interpolation ensures constant angular positioning on each segment.

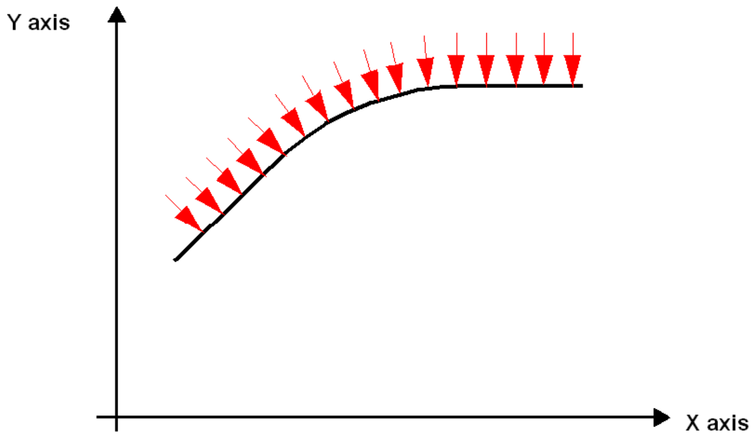
**NOTE:** This type of interpolation is only used on point P0 to indicate that the third axis is tangent type.

**NOTE: For each curve segment, the position on the third axis indicates the angle (in degrees) according to which the cutting tool is to be positioned if bit 8 of parameter ParW4 of the segment is at 0. If this bit 8 is at 1 the tool position will automatically be at a tangent to the curve.**
Tangent mode (bit 8 at 1) will be available in a later version of the TjE/TSX CSY 85 combination, but for the moment only angular positioning of the third Z axis is supported.

**NOTE:** Code 100 is used to position the tool on one side of the curve and code 101 for the other side.

### Illustration

The red arrows in the diagram below indicate the positioning of the third axis on the curve:



## Block Parameters

The table below indicates the parameters to be entered in the trajectory table for this type of segment:

| Parameter | Description |
|---|---|
| ParW0 | Enter 101 to indicate that the third axis of the movement is tangent type.<br>**Note**: This type is only used on the first point P0 of the curve as it indicates the operating mode of the third axis for the whole trajectory. |

## Section 4.2
## Master Position Control and Speed Management

### Aim of this Section

This section describes the master's position control function and the function for managing the speed of movement along the trajectory.

### What Is in This Section?

This section contains the following topics:

## GetTabResultF: Trajectory Calculation Result (ACTION_TRF = 16901)

### At a Glance

Having uploaded the trajectory to the module *(see page 47)*, the module has calculated the various cam profiles (which associate real slave axes with the master axis) for each axis.

You may now proceed to work on the movement and its trajectory by obtaining various different items of information.

The `GetTabResultF` function can be used to obtain the maximum permissible speed as well as the calculated speed (speed actually used).

### Function Syntax

`GetTabResultF` is executed:

- On a slave axis group type channel (channels 21 to 24)
- With the following parameter settings:
  - `%MWr.m.c.10` (ACTION_TRF) = 16901
  - `%MDr.m.c.11` (param_trf_1) = 0,
  - `%MDr.m.c.13` (param_trf_2) = 0,
  - `%MDr.m.c.15` (param_trf_3) = **0.0 to read the master table, 1.0 to read the maximum permissible speed table and 2.0 for the calculated speeds table**.
  - `%MDr.m.c.17` (param_trf_4) = 0,
- and the TRF_RECIPE function with parameter settings taken from the word table, which must receive the information requested by **param_trf_3** as well as the length of the table.

TRF_RECIPE `%CHr.m.c` (Table_length, Reception_table)

| Parameter | Description |
|---|---|
| `%CHr.m.c` | Channel associated with module channel group consisting of the slave axes and master axis of the trajectory for which information is required.<br>**Example**: %CH3.21 |
| Table_length | Number of words in the reception table: **Table_length** = 4 x number of points in trajectory. |
| Reception_table | First word in the table to which the values requested by parameter **param_trf_3** have been assigned. This table contains two floating points for each point on the trajectory. These words contain the speeds (param_trf_3 = 1.0 or 2.0) or the characteristic positions for each segment type (param_trf_3 = 0.0). A more detailed description of the information received when you wish to obtain the position of the master appears below.<br>**Example**: `%MW0`: |

## Master Position

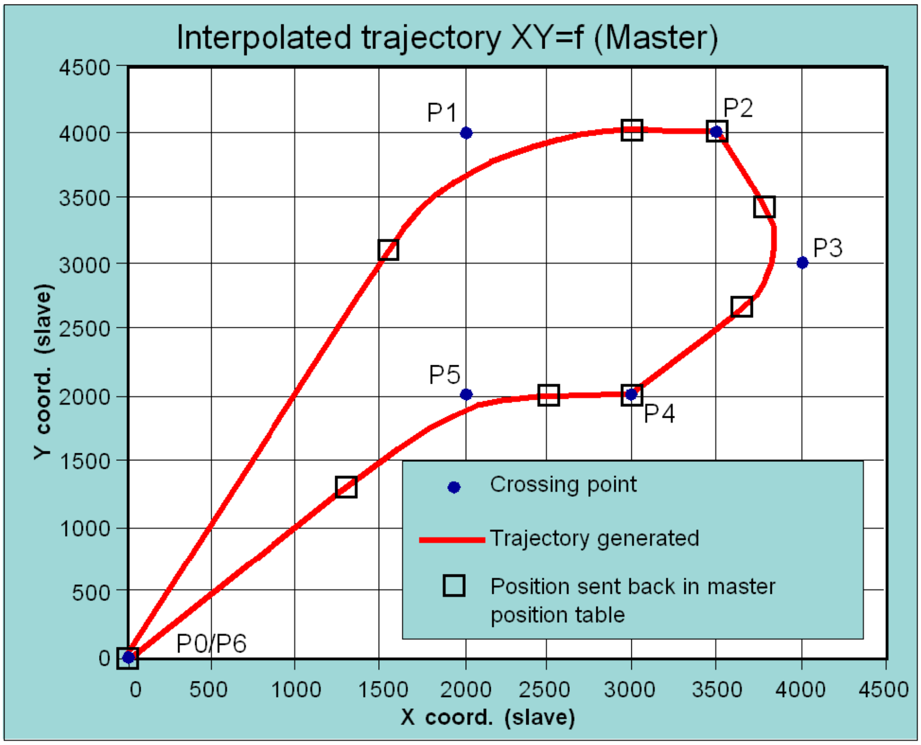
When you request the position of the master, you will receive two floating-point words for each point (P0, P1, , Pn...) in the slave axis segments. These floating-point words indicate the position of the master points that are identical to the slave points for type 0 *(see page 52)*, 11 *(see page 64)* and 12 *(see page 66)* segments. In the case of type 1 *(see page 53)*, 2 *(see page 57)* and 10 *(see page 60)* segments, interpolation creates two segments in order to represent the segment required and the points (and therefore the speeds) do not coincide.

The table below lists example points for a trajectory illustrated below.

| Point | X | Y | Type | Master position | Point on illustration |
|---|---|---|---|---|---|
| P0 | 0 | 0 | Not known | 0 | Always 0 |
| | | | | 0 | Always 0 |
| P1 | 2000 | 4000 | 10 | 3472,136 | End of linear section between P0 and P1, start of circular link |
| | | | | 5263,540 | End of circular link at P1 |
| P2 | 3500 | 4000 | 0 | 5263,540 | End of linear section between P1 and P2 |
| | | | | 5263,540 | End of linear section between P1 and P2 |
| P3 | 4000 | 3000 | 10 | 6381,574 | End of linear section between P2 and P3, start of circular link |
| | | | | 7248,053 | End of circular link at P3 |
| P4 | 3000 | 2000 | 0 | 8162,267 | End of linear section between P3 and P4 |
| | | | | 8162,267 | End of linear section between P3 and P4 |
| P5 | 2000 | 2000 | 1 | 8162,267 | End of linear section between P4 and P5, start of third-degree link |
| | | | | 10161,267 | End of third-degree link at P5 |
| P6 | 0 | 0 | Linear link | 11990,695 | End of linear section between P5 and P6 (total master length) |
| | | | | 11990,695 | End of linear section between P5 and P6 (total master length) |

## Illustration

Illustration for above table:



## MoveImmedInterpo: Master Speed Control (ACTION_CMD = 905)

### At a Glance

Having uploaded the trajectory to the module *(see page 47)*, the module has calculated the various cam profiles (which associate real slave axes with the virtual master axis) for each axis.

You may now proceed to modify the movement and its trajectory.

The `MoveImmedInterpo` function can be used to position the master on the trajectory using the maximum permissible speed based on the various segments of which it consists.

On each segment, the module tries to achieve the maximum speed between the speed required and the speed permitted on the segment, observing the acceleration and deceleration rates set on the master.

### Function Syntax

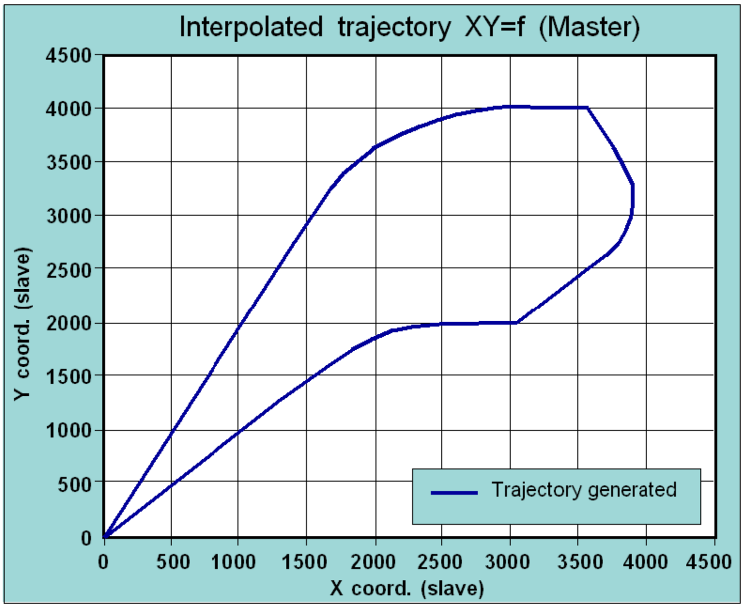
The `MoveImmedInterpo` function is executed:

- On channel 0 of the module
- With the following parameter settings:
  - `%MWr.m.c.26` (`ACTION_CMD`) = 905
  - `%MDr.m.c.27` (param_cmd_1) = Axis group ID (601 to 604)
  - `%MDr.m.c.29` (param_cmd_2) = 10 if absolute speed (param_cmd_4), any other value indicates that the speed in param_cmd_4 is a percentage of the maximum speed calculated
  - `%MDr.m.c.31` (param_cmd_3) = Target position
  - `%MDr.m.c.33` (param_cmd_4) = Speed required, either absolute or relative (0 for 0% to 10000 for 100%) based on the value of param_cmd_2
- The `WRITE_CMD` instruction assigned to channel 0 of the module

**NOTE:** To obtain the axis ID, use the `GetAxisID` function (code 523) and the `WRITE_CMD` instruction.

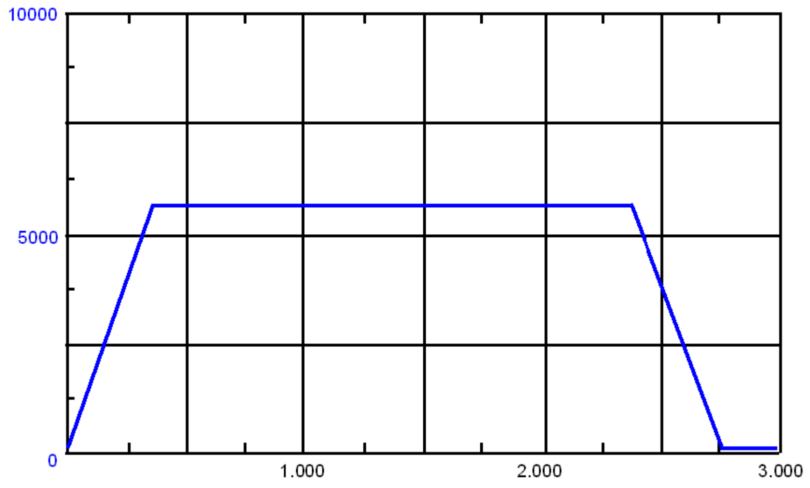
## Example Trajectory

The diagram below illustrates an example trajectory:



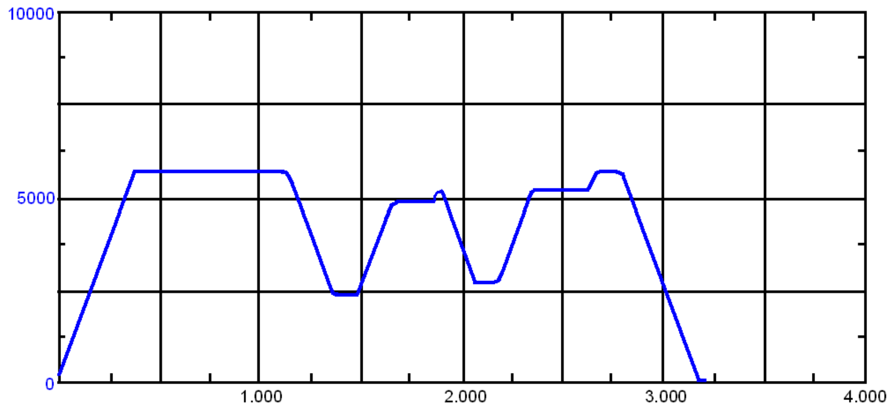
## Example of Non-Modulated Movement

The diagram below illustrates an example of non-modulated movement (param_cmd_2 = 10) based on the example trajectory above:



## Example of Modulated Movement

The diagram below illustrates an example of modulated movement (param_cmd_2 other than 10, param_cmd_4 at 0 for 0% to 10000 for 100%) based on the example trajectory above:



**NOTE:** On this illustration, note that the master varies its speed based on the segments (decelerating on tight segments). When the speed is modulated, movement is performed in full. However, when the speed is **not modulated** (see diagram above), the axes may switch to fault mode as the speed is too great for the dynamics of the movement.

## Error Codes

When this type of function is used, the error codes that can be returned in word `%MWr.m.c.3` are as follows:

| Code | Description |
|---|---|
| 7001 | For type 1, 2 and 10 interpolation, one of the parameters ParF1 or ParF2 has been linked to 0. |
| 7002 | Q_stop is outside the points of the trajectory. |
| 7003 | The motion is not controlled correctly. |
| 7004 | The active movement has been stopped. |
| 7038 | The specified group (ID) is not configured. |
| 7046 | The trajectory of the selected group has not been calculated. |
| 9015 | The module version does not support this function. |

## GetMinimumConstantSpeed: Achieving Minimum Possible Constant Speed (ACTION_TRF = 14905)

### At a Glance

Having uploaded the trajectory to the module *(see page 47)*, the module has calculated the various cam profiles (which associate real slave axes with the master axis) for each axis.

You may now proceed to work on the movement and its trajectory by obtaining various different items of information.

The `GetMinimumConstantSpeed` function can be used to obtain the minimum speed permitted across all segments of the trajectory.

### Function Syntax

The `GetMinimumConstantSpeed` function is executed:

- On a slave axis group type channel (channels 21 to 24)
- With the following parameter settings:
  - `%MWr.m.c.10` (ACTION_TRF) = 14905
  - `%MDr.m.c.11` (param_trf_1) = 0,
  - `%MDr.m.c.13` (param_trf_2) = 0,
  - `%MDr.m.c.15` (param_trf_3) = 0,
  - `%MDr.m.c.17` (param_trf_4) = 0,
  - `%MDr.m.c.6` (return_trf_2) = Contains the required speed
- Plus the TRF_RECIPE function, the parameters of which are not used

**Function call syntax:**

```
TRF_RECIPE %CHr.m.c (0, 0)
```

### Error Codes

When this type of function is used, the error codes that can be returned in word `%MWr.m.c.3` are as follows:

| Code | Description |
|---|---|
| 1 | Incorrect group number |
| 3 | The function was executed before the trajectory was calculated. |

# Chapter 5
# SERCOS language objects

## Subject of this Chapter

This chapter describes the language objects associated with SERCOS modules.

## What Is in This Chapter?

This chapter contains the following sections:

## Section 5.1
## The language objects and IODDTs of the SERCOS module

### Subject of this Section

This section covers general points relating to the language objects and IODDTs of the SERCOS module.

### What Is in This Section?

This section contains the following topics:

## Introduction to IODDTs Associated With TSX CSY 85 Modules

### General

The IODDTs are predefined by the manufacturer. They contain input/output language objects belonging to a channel of an application-specific module.

The TSX CSY 85 module has 7 types of IODDT:

- `T_CSY_CMD`,
- `T_CSY_RING`,
- `T_CSY_TRF`,
- `T_CSY_IND`,
- `T_CSY_FOLLOW`,
- `T_CSY_COORD`,
- `T_CSY_CAM`.

**NOTE:** There are two ways to create an IODDT variable:
- **I/O Objects tab** *(see EcoStruxure™ Control Expert, Operating Modes)*,
- data editor *(see EcoStruxure™ Control Expert, Operating Modes)*.

### Language Object Types

Each IODDT contains a set of language objects which can be used to control and check how they operate.

There are two types of language object:

- **implicit exchange objects**, which are exchanged automatically for every cycle of the task associated with the module,
- **explicit exchange objects**, which are exchanged at the request of the application, by using explicit exchange instructions.

Implicit exchanges are concerned with the status of modules, communication signals, slaves etc.

Explicit exchanges are used to set the parameters of the module and to perform module diagnostics.

## Implicit Exchange Language Objects Associated with the Application-Specific Function

### At a Glance

An integrated application-specific interface or the addition of a module automatically enhances the language objects application used to program this interface or module.

These objects correspond to the input/output images and software data of the module or integrated application-specific interface.

### Reminders

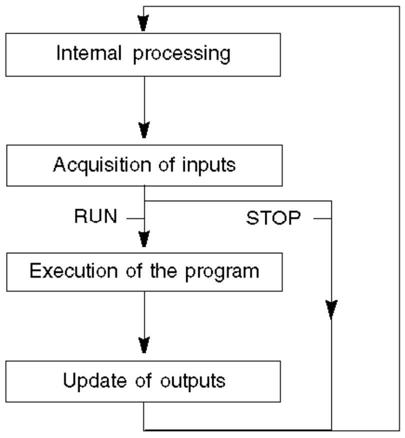
The module inputs (`%I` and `%IW`) are updated in the PLC memory at the start of the task, the PLC being in RUN or STOP mode.

The outputs (`%Q` and `%QW`) are updated at the end of the task, only when the PLC is in RUN mode.

**NOTE:** When the task occurs in STOP mode, either of the following are possible, depending on the configuration selected:
- outputs are set to fallback position (fallback mode)
- outputs are maintained at their last value (maintain mode)

### Figure

The following diagram shows the operating cycle of a PLC task (cyclical execution).



## Explicit Exchange Language Objects Associated with the Application-Specific Function

### Introduction

Explicit exchanges are performed at the user program's request using these instructions:
- READ_STS (read status words)
- WRITE_CMD (write command words)
- WRITE_PARAM (write adjustment parameters)
- READ_PARAM (read adjustment parameters)
- SAVE_PARAM (save adjustment parameters)
- RESTORE_PARAM (restore adjustment parameters)

For more details about instructions, refer to *EcoStruxure™ Control Expert, I/O Management, Block Library*.

These exchanges apply to a set of %MW objects of the same type (status, commands or parameters) that belong to a channel.

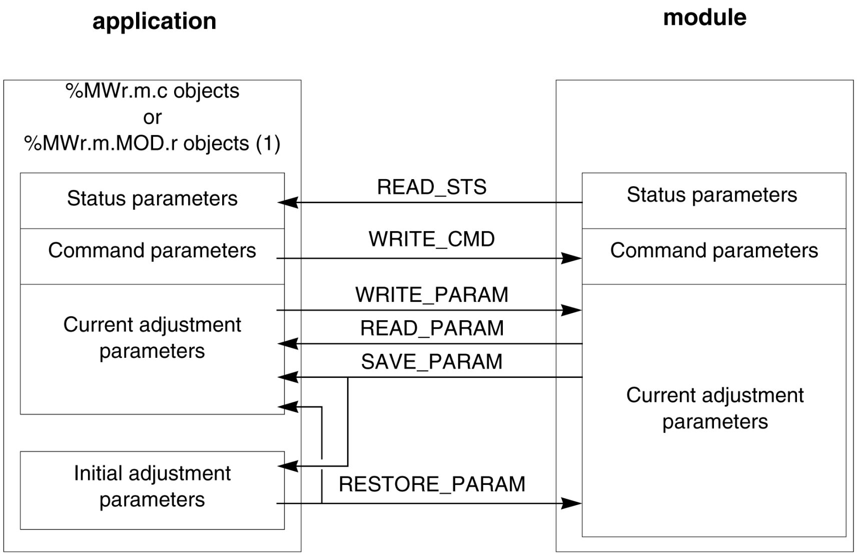
These objects can:
- provide information about the module (for example, type of error detected in a channel)
- have command control of the module (for example, switch command)
- define the module's operating modes (save and restore adjustment parameters in the process of application)

**NOTE:** To avoid several simultaneous explicit exchanges for the same channel, it is necessary to test the value of the word EXCH_STS (`%MWr.m.c.0`) of the IODDT associated to the channel before calling any EF addressing this channel.

**NOTE:** Explicit exchanges are not supported when X80 analog and digital I/O modules are configured through an eX80 adapter module (BMECRA31210) in a Quantum EIO configuration. You cannot set up a module's parameters from the PLC application during operation.

## General Principle for Using Explicit Instructions

The diagram below shows the different types of explicit exchanges that can be made between the application and module.



(1) Only with READ_STS and WRITE_CMD instructions.

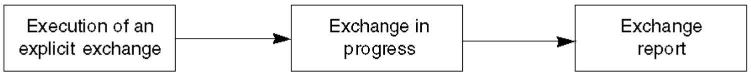
## Managing Exchanges

During an explicit exchange, check performance to see that the data is only taken into account when the exchange has been correctly executed.

To do this, two types of information is available:

- information concerning the exchange in progress *(see page 88)*
- the exchange report *(see page 88)*

The following diagram describes the management principle for an exchange.



**NOTE:** In order to avoid several simultaneous explicit exchanges for the same channel, it is necessary to test the value of the word EXCH_STS (`%MWr.m.c.0`) of the IODDT associated to the channel before calling any EF addressing this channel.

## Management of Exchanges and Reports with Explicit Objects

### At a Glance

When data is exchanged between the PLC memory and the module, the module may require several task cycles to acknowledge this information. IODDTs use two words to manage exchanges:

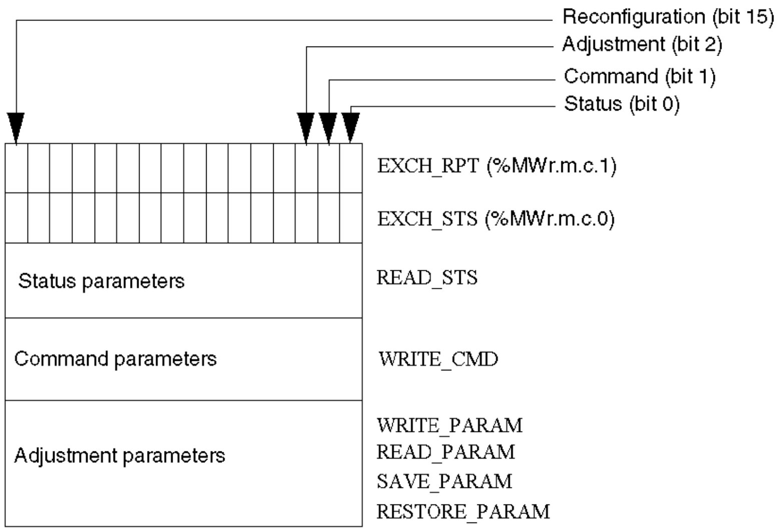
- `EXCH_STS` (`%MWr.m.c.0`): exchange in progress
- `EXCH_RPT` (`%MWr.m.c.1`): report

**NOTE:**
Depending on the localization of the module, the management of the explicit exchanges (`%MW0.0.MOD.0.0` for example) will not be detected by the application:

- For in-rack modules, explicit exchanges are done immediately on the local PLC Bus and are finished before the end of the execution task. So, the `READ_STS`, for example, is finished when the `%MW0.0.mod.0.0` bit is checked by the application.
- For remote bus (Fipio for example), explicit exchanges are not synchronous with the execution task, so the detection is possible by the application.

### Illustration

The illustration below shows the different significant bits for managing exchanges:



## Description of Significant Bits

Each bit of the words `EXCH_STS` (`%MWr.m.c.0`) and `EXCH_RPT` (`%MWr.m.c.1`) is associated with a type of parameter:

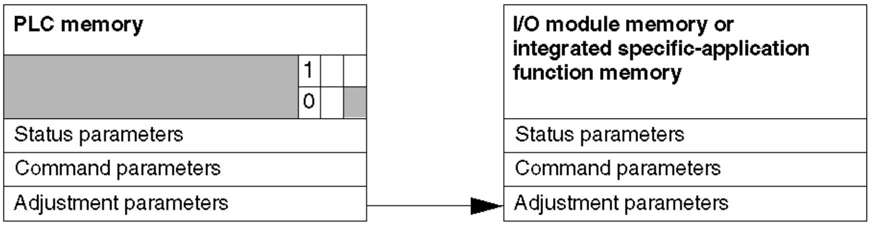
- Rank 0 bits are associated with the status parameters:
  - The `STS_IN_PROGR` bit (`%MWr.m.c.0.0`) indicates whether a read request for the status words is in progress.
  - The `STS_ERR` bit (`%MWr.m.c.1.0`) specifies whether a read request for the status words is accepted by the module channel.
- Rank 1 bits are associated with the command parameters:
  - The `CMD_IN_PROGR` bit (`%MWr.m.c.0.1`) indicates whether command parameters are being sent to the module channel.
  - The `CMD_ERR` bit (`%MWr.m.c.1.1`) specifies whether the command parameters are accepted by the module channel.
- Rank 2 bits are associated with the adjustment parameters:
  - The `ADJ_IN_PROGR` bit (`%MWr.m.c.0.2`) indicates whether the adjustment parameters are being exchanged with the module channel (via `WRITE_PARAM, READ_PARAM, SAVE_PARAM, RESTORE_PARAM`).
  - The `ADJ_ERR` bit (`%MWr.m.c.1.2`) specifies whether the adjustment parameters are accepted by the module. If the exchange is correctly executed, the bit is set to 0.
- Rank 15 bits indicate a reconfiguration on channel **c** of the module from the console (modification of the configuration parameters + cold start-up of the channel).
- The *r*, *m* and *c* bits indicates the following elements:
  - the **r** bit represents the rack number.
  - The **m** bit represents the position of the module in the rack.
  - The **c** bit represents the channel number in the module.

**NOTE:** **r** represents the rack number, **m** the position of the module in the rack, while **c** represents the channel number in the module.

**NOTE:** Exchange and report words also exist at module level `EXCH_STS` (`%MWr.m.MOD`) and `EXCH_RPT` (`%MWr.m.MOD.1`) as per IODDT type `T_GEN_MOD`.

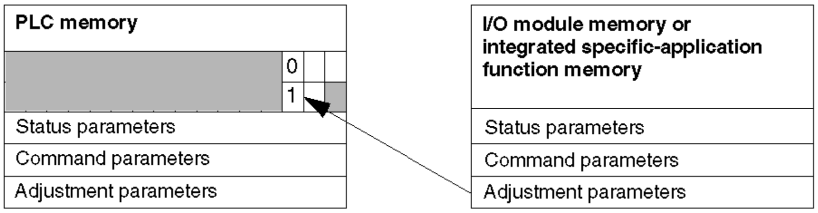
## Example

Phase 1: Sending data by using the `WRITE_PARAM` instruction



When the instruction is scanned by the PLC, the **Exchange in progress** bit is set to 1 in `%MWr.m.c`.

Phase 2: Analysis of the data by the I/O module and report.



When the data is exchanged between the PLC memory and the module, acknowledgement by the module is managed by the `ADJ_ERR` bit (`%MWr.m.c.1.2`).

This bit makes the following reports:

- **0**: correct exchange
- **1**: incorrect exchange)

**NOTE:** There is no adjustment parameter at module level.

## Execution Indicators for an Explicit Exchange: EXCH_STS

The table below shows the control bits of the explicit exchanges: `EXCH_STS` (`%MWr.m.c.0`)

| Standard Symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_IN_PROGR | BOOL | R | Reading of channel status words in progress | %MWr.m.c.0.0 |
| CMD_IN_PROGR | BOOL | R | Command parameters exchange in progress | %MWr.m.c.0.1 |
| ADJ_IN_PROGR | BOOL | R | Adjust parameters exchange in progress | %MWr.m.c.0.2 |
| RECONF_IN_PROGR | BOOL | R | Reconfiguration of the module in progress | %MWr.m.c.0.15 |

**NOTE:** If the module is not present or is disconnected, explicit exchange objects (`READ_STS` for example) are not sent to the module (`STS_IN_PROG` (%MWr.m.c.0.0) = 0), but the words are refreshed.

## Explicit Exchange Report: EXCH_RPT

The table below shows the report bits: `EXCH_RPT` (`%MWr.m.c.1`)

| Standard Symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_ERR | BOOL | R | Error detected while reading channel status words (1 = detected error) | %MWr.m.c.1.0 |
| CMD_ERR | BOOL | R | Error detected during a command parameter exchange (1 = detected error) | %MWr.m.c.1.1 |
| ADJ_ERR | BOOL | R | Error dectected during an adjust parameter exchange (1 = detected error) | %MWr.m.c.1.2 |
| RECONF_ERR | BOOL | R | Error detected during reconfiguration of the channel (1 = detected error) | %MWr.m.c.1.15 |

## Counting Module Use

The following table describes the steps realized between a couting module and the system after a power-on.

| Step | Action |
|---|---|
| 1 | Power on. |
| 2 | The system sends the configuration parameters. |
| 3 | The system sends the adjust parameters by WRITE_PARAM method.<br>**Note:** When the operation is finished, the bit %MWr.m.c.0.2 switches to 0. |

If, in the begining of your application, you use a WRITE_PARAM command, wait until the bit %MWr.m.c.0.2 switches to 0.

## Interface Language

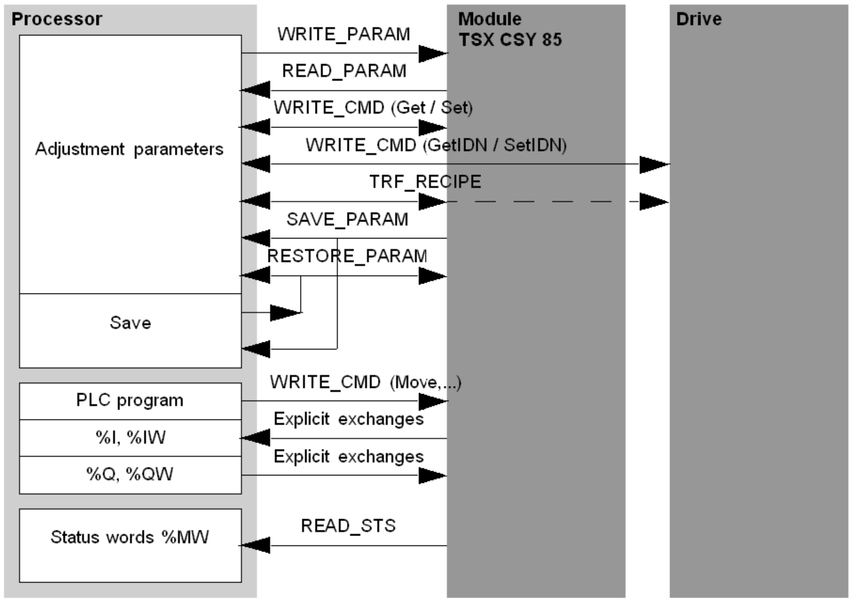
### Module Channels

The TSX CSY 85 module is made up of up to 32 channels which support the following functions:

| Channels | Function supported |
|---|---|
| %CHr.m.0 | SERCOS® function |
| %CHr.m.1 to %CHr.m.8 | Real axes |
| %CHr.m.9 to %CHr.m.12 | Imaginary axes |
| %CHr.m.13 to %CHr.m.16 | Remote Axes |
| %CHr.m.17 to %CHr.m.20 | Co-ordinated axis group |
| %CHr.m.21 to %CHr.m.24 | Slave axis group |
| %CHr.m.25 to %CHr.m.31 | Cam Profiles |

### Overview of Exchanges

The exchanges between the processor, the axis control module and the servodrives are performed as follows:



## Parameter Management

### General

The parameters of an axis control module and those of servodrives are managed separately via Control Expert and UniLink. For example, it is possible to define the different values for limits in an axis control module and servodrives.

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

**NOTE:** Drives that operate with a linear type unit are not supported by the module TSX CSY 85.

### Axis Control Module Parameters

Type "C" (Controller) parameters are exchanged whilst floating, in the axis control module units. These parameters are related to the path calculation or cam profiles. They are configured in Control Expert configuration screens, then modified by the application as required. The units of these parameters are the command axis module units as defined in the configuration screens.

### Servodrive Parameters

When the application modifies a parameter in the axis control module, the corresponding parameter in the servodrives is not updated (e.g. the minimum and maximum position limits, minimum and maximum acceleration, maximum speed, debugging window).

These parameters are considered to be servodrive system parameters. The user configures them via UniLink in order to protect the operational part and it is not necessary to change them when operating (the application program modifies the equivalent "C" type parameters in the command axis module).

### Parameters Used by the Axis Control Module and the Drives

The axis control module does not update the servodrive parameters according to these equivalent parameter values.

Type "S" parameters (identifiers (IDN) SERCOS® Standard) and type "P" parameters (identifiers (IDN) SERCOS® Owner) are exchanged whilst floating with the axis control module and in full with the servodrives (e.g. acceleration, speed and position values).

If the application requires the value of the axis control module parameters to be synchronized with the servodrive parameters, it can read "S" or "P" type parameters (TRF_RECIPE) and then write them in the "C" type parameters (WRITE_PARAM).

## "Unit" Parameters

Units are parameters used both by the axis control module and the drives.

Units are configured using Control Expert. The conversion of axis control module units into servodrive units is automatically performed by the module TSX XSY 85 using a scale factor configured by the user.

## WRITE_PARAM and READ_PARAM

### Reminder

These services allow the exchange of adjustment parameters between the processor (project) and the axis control module.

`READ_PARAM`: explicit reading of the parameters in the axis control module and updating of the adjustment words %MW/D/Fr.m.c.r. via IODDTs.

`WRITE_PARAM`: explicit writing of parameters in the axis control module. This instruction allows programmed modification of the adjustment values defined in the configuration.

These two instructions are applied to an IODDT variable associated with the TSX CSY 84 module. In the following paragraphs we will take the example of a variable called `Serco_Channel` of type T_CSY_IND.

### Syntax of the instruction READ_PARAM

`READ_PARAM (Sercos_Channel)`: reads the adjustment parameters of the channel associated with the IODDT `Sercos_Channel`.

### Syntax of instruction WRITE_PARAM

`WRITE_PARAM (Sercos_Channel)`: writes the adjustment parameters of the channel associated with the IODDT `Sercos_Channel`.

### Monitoring the Exchange

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

| Standard symbol | Meaning | Bit |
|---|---|---|
| ADJ_IN_PROGR | This bit is set to 1 while the exchange is in progress. It is reset to 0 when the exchange has been completed. | %MWr.m.c.0.2 |
| ADJ_ERR | This bit is set to 1 if the parameters transmitted are out of range or erroneous. | %MWr.m.c.1.2 |

## SAVE_PARAM and RESTORE_PARAM

### Reminder

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

`SAVE_PARAM`: explicit backup of the axis control module parameters. These parameters replace the initial values defined during configuration.

`RESTORE_PARAM`: explicit restoration of the initial adjustment parameters (written at the time of configuration or last backup).

These two instructions are applied to an IODDT variable associated with the TSX CSY 84 module. In the following paragraphs we will take the example of a variable called `Serco_Channel` of type T_CSY_IND.

### Syntax of instruction SAVE_PARAM

`SAVE_PARAM (Sercos_Channel)` : saves the adjustment parameters of the channel associated with the IODDT `Sercos_Channel`.

### Syntax of instruction RESTORE_PARAM

`RESTORE_PARAM (Sercos_Channel)`: restores the adjustment parameters of the channel associated with the IODDT `Sercos_Channel`.

### Monitoring the Exchange

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

| Standard symbol | Meaning | Bit |
|---|---|---|
| ADJ_IN_PROGR | This bit is set at 1 when the exchange is in process. It is reset to 0 when the exchange has been completed. | %MWr.m.c.0.2 |
| ADJ_ERR | This bit is set at 1 if the parameters passed on are out of range or erroneous. | %MWr.m.c.1.2 |

## WRITE_CMD

### Reminder

This service allows a command to be sent to the motion controller.

`WRITE_CMD`: Explicit writing of command words in the module. This operation is carried out via internal words %MW, which contain the command to be carried out and its parameters (a motion control, for example).

This instruction is applied to an IODDT type variable associated with the TSX CSY 84 and TSX CSY 164 modules. For an independent axis, we will take the example of a variable called `Serco_Channel` of type **T_CSY_IND**. For a co-ordinated axis, we shall take the example of a variable called `Serco_Channel_coord` of type **T_CSY_COORD**.

### Syntax of WRITE_CMD Instruction

`WRITE_CMD (Sercos_Channel)`: writes the command information of the channel associated with the IODDT `Sercos_Channel` (co-ordinated axis).

`WRITE_CMD (Sercos_Channel_coord)` for the co-ordinated axis.

### WRITE_CMD Interface

The command to be performed is defined in the `ACTION_CMD` word (%MWr.m.c.26) of the `Sercos_Channel` IODDT and the result of the command is available in the words described in the following table:

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_CMD | INT | RW | WRITE_CMD command write error | %MWr.m.c.19 |
| RETURN_CMD_1 | DINT | RW | Return 1 of the function | %MDr.m.c.20 |
| RETURN_CMD_2 | REAL | RW | Return 2 of the function | %MFr.m.c.22 |
| RETURN_CMD_3 | REAL | RW | Return 3 of the function | %MFr.m.c.24 |
| ACTION_CMD | INT | RW | Action to be carried out | %MWr.m.c.26 |
| PARAM_CMD_1 | DINT | RW | Parameter 1 | %MDr.m.c.27 |
| PARAM_CMD_2 | DINT | RW | Parameter 2 | %MDr.m.c.29 |
| PARAM_CMD_3 | REAL | RW | Parameter 3 | %MFr.m.c.31 |
| PARAM_CMD_4 | REAL | RW | Parameter 4 | %MFr.m.c.33 |

## WRITE_CMD interface in the case of a coordinated axis group

In the case of a coordinated axis group, the movement functions make it necessary to send 2 parameters per coordinated axis (position and speed). The IODDT associated with this axis is thus a little more sophisticated. The result of the command and the parameters are located in the following words of the `Sercos_Channel_Coord` IODDT:

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| PARAM_CMD_1 | DINT | RW | Parameter 1 | %MDr.m.c.27 |
| PARAM_CMD_2 | DINT | RW | Parameter 2 | %MDr.m.c.29 |
| PARAM_CMD_3 | REAL | RW | Parameter 3 | %MFr.m.c.31 |
| PARAM_CMD_4 | REAL | RW | Parameter 4 | %MFr.m.c.33 |
| PARAM_CMD_5 | REAL | RW | Parameter 5 | %MFr.m.c.35 |
| ... | ... | ... | ... | ... |
| PARAM_CMD_18 | REAL | RW | Parameter 18 | %MFr.m.c.61 |

## Monitoring the Exchange

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

| Standard symbol | Meaning | Bit |
|---|---|---|
| CMD_IN_PROGR | This bit is set to 1 while the exchange is in progress. It is reset to 0 when the exchange has been completed. | %MWr.m.c.0.1 |
| CMD_ERR | This bit is set to 1 if the parameters transmitted are out of range or erroneous. | %MWr.m.c.1.1 |

## WRITE_CMD Examples

### Example 1: Initializing Faults

To initialize faults on the bus ring of module 4 in rack 1. We declare the `Ring` IODDT as type `T_CSY_RING` and associate to channel 0 of the TSX CSY 85 module in rack 1, slot 4:

```
!   (*If  WRITE_CMD not in progress, then initialization of faults*)
    IF NOT Ring.CMD_IN_PROGR THEN Ring.ACTION_CMD:= 409;
    WRITE_CMD(Ring);
    END_IF;
```

### Example 2: Writing the Optical Power

To define (write) the optical power. We use the same IODDT:

```
!   (*If  WRITE_CMD not in progress, then writing (set) of the optical power*)
    IF NOT Ring.CMD_IN_PROGR THEN Ring.ACTION_CMD:= 2545;
    Ring.PARAM_CMD_3:= 51.5;            //Parameter 3: optical power = 51.5
    WRITE_CMD(Ring);
    END_IF;
```

### Example 3: Reading the transmission ratio

To read the transmission ratio (BAUD_RATE):

```
!   (*If  WRITE_CMD not in progress, then reading (get) of the transmission
    ratio*)
    IF NOT Ring.CMD_IN_PROGR THEN Ring.ACTION_CMD:= 1551;
    WRITE_CMD(Ring);
    END_IF;
!   (*If WRITE_CMD has been completed and if there is no fault, then the value
    of the transmission ratio is accessible in return 1*)
    IF NOT Ring.CMD_IN_PROGR AND NOT Ring.CMD_ERR
    THEN BAUD_RATE: =                   //transmission ratio in return 1
    Ring.RETURN_CMD_1;
    END_IF;
```

## Example 4: Moving a real axis

Moving the real axis 3 from module 4 set in rack 1, at position 105.2, at speed 5 with the command "MOVE absolute immediate". We declare the `Axis_3` IODDT of type `T_CSY_IND` and associate it with channel 3 of module 4 of rack 1:

```
!   (*If WRITE_CMD not in progress, then move real axis 3*)
    IF NOT Axis_3.CMD_IN_PROGR THEN Axis_3.ACTION_CMD:= 513;
    Axis_3.PARAM_CMD_1:= 0;           //Parameter 1: type of movement = absolute
    Axis_3.PARAM_CMD_3:= 105.2;       //Parameter 3: position = 105.2
    Axis_3.PARAM_CMD_4:= 5.0;         //Parameter 4: speed = 5
    WRITE_CMD(Axis_3);
    END_IF;
```

## Example 5: Reading the position of a slave

Read the relative position of a follower slave when the master is at 102.5. We declare the IODDT `Follow` of type `T_CSY_FOLLOW` and associate it with the channel corresponding to the follower slave:

```
!   (*If  WRITE_CMD not in progress, then reading the position of the follower
    slave*)
    IF NOT Follow.CMD_IN_PROGR THEN Follow.ACTION_CMD:= 537;
    Follow.PARAM_CMD_3:= 102.5;          //Parameter 3: position of the master = 102.5
    WRITE_CMD(Follow);
    END_IF;
!   (*If WRITE_CMD has been completed and if there is no fault, then the position
    of the slave is accessible in return 2*)
    IF NOT Follow.CMD_IN_PROGR AND NOT Follow.CMD_ERR
    THEN                                 //position in return 2
    SLAVE_POSITION:=Follow.RETURN_CMD_2;
    END_IF;
```

## READ_STS

### Reminder

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

This instruction may be applied to all types of IODDT that can be associated with the TSX CSY 85 module. For an independent axis, we will take the example of a variable called `Serco_Channel` of type **T_CSY_IND**.

It is also possible to apply this instruction to a TSX CSY 85 module. In this case, it has to be applied to an IODDT of type `T_CSY_CMD`. Let us use the example of the variable called `Sercos_module`.

### Syntax of the READ_STS (module) instruction

`READ_STS (Sercos_Module)`: reading of general diagnostics information from the module associated with the `Serco_Module` IODDT.

### Syntax of the READ_STS (channel) instruction

`READ_STS (Sercos_Channel)`: reading of general diagnostics information from the channel associated with the `Sercos_Channel` IODDT.

### Monitoring the Exchange

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

| Standard symbol | Meaning | Bit |
|---|---|---|
| STS_IN_PROGR | This bit is set at 1 when the reading is in process. It is reset to 0 when the exchange has been completed. | %MWr.m.c.0.0 |
| STS_ERR | This bit provides the exchange report. It is set to 1 in the case of fault. | %MWr.m.c.1.0 |

## Section 5.2
## Language Objects and IODDTs Associated Specifically with the SERCOS Module

### Subject of this Section

This section presents the language objects and IODDTs associated with the SERCOS module.

### What Is in This Section?

This section contains the following topics:

## Details Concerning T_CSY_CMD-type IODDT Implicit Exchange Objects

### At a Glance

The `T_CSY_CMD` IODDT possesses implicit exchange objects, which are described below. This type of IODDT applies to the `TSX_CSY_85` module.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_CMD`.

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

### List of Implicit Exchange Input Objects

The following table introduces the implicit exchange input objects of the `T_CSY_CMD` IODDT which apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CH_ERROR | EBOOL | R | Channel error bit. | %Ir.m.c.ERR |
| PROFILE_END | EBOOL | R | The last profile command has been sent to the module | %Ir.m.c.3 |
| IN_POSITION | EBOOL | R | The axis is within the in-position band | %Ir.m.c.4 |
| AXIS_HOMED | EBOOL | R | The axis position reading is referenced off the home position | %Ir.m.c.6 |
| HOLDING | EBOOL | R | The axis is holding in wait position | %Ir.m.c.8 |
| RESUMING | EBOOL | R | The axis is moving after a hold | %Ir.m.c.9 |
| DRIVE_ENABLED | EBOOL | R | The servodrive is enabled | %Ir.m.c.10 |
| DRIVE_FLT | EBOOL | R | The drive is performing a class 1 diagnostic | %Ir.m.c.13 |
| AXIS_SUMMARY_FLT | EBOOL | R | Drive fault | %Ir.m.c.15 |
| AXIS_IN_CMD | EBOOL | R | The axis is active and can be controlled | %Ir.m.c.18 |
| AXIS_HOLD | EBOOL | R | The axis is stopped and waiting for a command | %Ir.m.c.28 |
| AXIS_HALT | EBOOL | R | The axis has stopped | %Ir.m.c.29 |
| AXIS_FASTSTOP | EBOOL | R | The axis has faststopped | %Ir.m.c.30 |
| AXIS_READY | EBOOL | R | The axis is ready to respond to a command | %Ir.m.c.31 |

## List of Implicit Exchange Output Objects

The following table introduces the implicit exchange output objects of the `T_CSY_CMD` IODDT that apply to TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CONTROL_ACQUIRE | EBOOL | RW | control acquisition | %Qr.m.c.2 |
| CONTROL_ENABLE | EBOOL | RW | Control enable | %Qr.m.c.10 |
| CONTROL_RESUME | EBOOL | RW | Resumes control after a stop | %Qr.m.c.12 |
| CONTROL_CLEAR_FLT | EBOOL | RW | Fault clear control | %Qr.m.c.15 |
| ALLOW_ACQUIRE | EBOOL | RW | Acquisition enable control | %Qr.m.c.18 |
| ALLOW_ENABLE | EBOOL | RW | Disable axis control | %Qr.m.c.26 |
| ALLOW_RESUME | EBOOL | RW | Authorizes a movement to continue after a stop using the HOLD command | %Qr.m.c.28 |
| ALLOW_MOVE | EBOOL | RW | Authorizes a movement to continue after a stop using the HALT command | %Qr.m.c.29 |

## Details Concerning T_CSY_CMD-Type IODDT Explicit Exchange Objects

### At a Glance

This part introduces explicit exchange objects of the `T_CSY_CMD` IODDT which apply to the TSX CSY 85 module. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_CMD`.

### Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

### Indicators of Explicit Exchange Execution: EXCH_STS

The table below explains the meanings of the exchange control bits for the `EXCH_STS` channel (`%MWr.m.c.0`).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CMD_IN_PROGR | BOOL | R | Command parameters exchange in progress | %MWr.m.c.0.1 |

### Explicit Exchange Report: EXCH_RPT

The following table explains the meaning of the `EXCH_RPT` report bits (`%MWr.m.c.1`).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CMD_ERR | BOOL | R | Fault during exchange of command parameters | %MWr.m.c.1.1 |

## WRITE_CMD Interface Words

The following table explains the meanings of the variables associated with the `WRITE_CMD` whose action was specified in the word `ACTION_CMD`. These variables are updated by a `WRITE_CMD` (`IODDT_VAR1`).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_CMD | INT | RW | Error during WRITE_CMD | %MWr.m.c.19 |
| RETURN_CMD_1 | DINT | RW | Return 1 of the function | %MDr.m.c.20 |
| RETURN_CMD_2 | REAL | RW | Return 2 of the function | %MFr.m.c.22 |
| RETURN_CMD_3 | REAL | RW | Return 3 of the function | %MFr.m.c.24 |
| ACTION_CMD | INT | RW | Action to be carried out | %MWr.m.c.26 |
| PARAM_CMD_1 | DINT | RW | Parameter 1 | %MDr.m.c.27 |
| PARAM_CMD_2 | DINT | RW | Parameter 2 | %MDr.m.c.29 |
| PARAM_CMD_3 | REAL | RW | Parameter 3 | %MFr.m.c.31 |
| PARAM_CMD_4 | REAL | RW | Parameter 4 | %MFr.m.c.33 |

## Details Concerning T_CSY_RING-Type IODDT Implicit Exchange Objects

### At a Glance

The `T_CSY_RING` IODDT possesses implicit exchange objects, which are described below. This type of IODDT applies to channel 0 of the `TSX_CSY_85` module.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_RING`.

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

### List of Implicit Exchange Input Objects

The following table introduces the implicit exchange input objects of the `T_CSY_RING` IODDT which apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CH_ERROR | EBOOL | R | Channel error bit. | %Ir.m.c.ERR |
| RAMPING | EBOOL | R | Indicates whether the axis is accelerating or decelerating | %Ir.m.c.0 |
| STEADY | EBOOL | R | The speed is steady | %Ir.m.c.1 |
| STOPPING | EBOOL | R | The movement is decelerating to a stop | %Ir.m.c.2 |
| PROFILE_END | EBOOL | R | The last profile command has been sent to the module | %Ir.m.c.3 |
| IN_POSITION | EBOOL | R | The axis is within the in-position band | %Ir.m.c.4 |
| AXIS_HOMING | EBOOL | R | The axis is homing. For an imaginary axis, this bit is inactive | %Ir.m.c.5 |
| AXIS_HOMED | EBOOL | R | The axis position reading is referenced off the home position | %Ir.m.c.6 |
| AXIS_NOT_FOLLOWING | EBOOL | R | The drive is not recognizing module commands | %Ir.m.c.7 |
| HOLDING | EBOOL | R | The axis is holding in wait position | %Ir.m.c.8 |
| RESUMING | EBOOL | R | The axis is moving after a hold | %Ir.m.c.9 |
| DRIVE_ENABLED | EBOOL | R | The servodrive is enabled | %Ir.m.c.10 |
| DRIVE_DIAG | EBOOL | R | The drive is performing a class 3 diagnostic | %Ir.m.c.11 |
| DRIVE_WARNING | EBOOL | R | The drive is performing a class 2 diagnostic | %Ir.m.c.12 |
| DRIVE_FLT | EBOOL | R | The drive is performing a class 1 diagnostic | %Ir.m.c.13 |
| DRIVE_DISABLED | EBOOL | R | The drive is disabled | %Ir.m.c.14 |
| AXIS_SUMMARY_FLT | EBOOL | R | Drive fault | %Ir.m.c.15 |
| AXIS_COM_OK | EBOOL | R | Communication between the drive and the module is OK | %Ir.m.c.16 |
| AXIS_IS_LINKED | EBOOL | R | The axis belongs to a set of axes | %Ir.m.c.17 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| AXIS_IN_CMD | EBOOL | R | The axis is active and can be controlled | %Ir.m.c.18 |
| AXIS_AT_TARGET | EBOOL | R | The axis is within the in-position band for the target position | %Ir.m.c.20 |
| AXIS_POS_LIMIT | EBOOL | R | The axis has reached the positive limit | %Ir.m.c.21 |
| AXIS_NEG_LIMIT | EBOOL | R | The axis has reached the negative limit | %Ir.m.c.22 |
| AXIS_WARNING | EBOOL | R | MotionWarning status returned by the drive | %Ir.m.c.23 |
| AXIS_HOLD | EBOOL | R | The axis is stopped and waiting for a command | %Ir.m.c.28 |
| AXIS_HALT | EBOOL | R | The axis has stopped | %Ir.m.c.29 |
| AXIS_FASTSTOP | EBOOL | R | The axis has faststopped | %Ir.m.c.30 |
| AXIS_READY | EBOOL | R | The axis is ready to respond to a command | %Ir.m.c.31 |
| CONF_OK | EBOOL | R | The channel is configured | %Ir.m.c.32 |
| LSUB_STATUS_IN_USE | EBOOL | R | Specific application in progres | %Ir.m.c.33 |
| LSUB_STATUS_DATA_READY | EBOOL | R | Specific application data ready | %Ir.m.c.34 |

## List of Implicit Exchange Output Objects

The following table introduces the implicit exchange output objects of the `T_CSY_RING` IODDT that apply to TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CONTROL_ACQUIRE | EBOOL | RW | control acquisition | %Qr.m.c.2 |
| CONTROL_ENABLE | EBOOL | RW | Control enable | %Qr.m.c.10 |
| CONTROL_FOLLOW | EBOOL | RW | Follow control for an axis or a set of follower axes | %Qr.m.c.11 |
| CONTROL_RESUME | EBOOL | RW | Resumes control after a stop | %Qr.m.c.12 |
| CONTROL_CLEAR_FLT | EBOOL | RW | Fault clear control | %Qr.m.c.15 |
| ALLOW_ACQUIRE | EBOOL | RW | Acquisition enable control | %Qr.m.c.18 |
| ALLOW_ENABLE | EBOOL | RW | Disable axis control | %Qr.m.c.26 |
| ALLOW_FOLLOW | EBOOL | RW | Control to cancel following for an axis or a set of follower axes | %Qr.m.c.27 |
| ALLOW_RESUME | EBOOL | RW | Authorizes a movement to continue after a stop using the HOLD command | %Qr.m.c.28 |
| ALLOW_MOVE | EBOOL | RW | Authorizes a movement to continue after a stop using the HALT command | %Qr.m.c.29 |
| ALLOW_NOT_FASTSTOP | EBOOL | RW | Control after a Faststop | %Qr.m.c.30 |
| ALLOW_NOT_FLT | EBOOL | RW | Enable fault control | %Qr.m.c.31 |

## Parameter Report Word

The table below introduces the implicit exchange input word of the `T_CSY_RING` IODDT which applies to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| PARAM_RPT | INT | R | Parameters report signaling a programming fault. The least significant byte contains the error code and the most significant byte contains the address in the registers of the field that triggered the error. | %IWr.m.c.2 |

## Details Concerning T_CSY_RING-Type IODDT Explicit Exchange Objects

### At a Glance

This part introduces explicit exchange objects of the `T_CSY_RING` IODDT which apply to channel 0 of the TSX CSY 85 module. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_RING`.

### Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

### Indicators of Explicit Exchange Execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: `EXCH_STS` (`%MWr.m.c.0`). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_IN_PROGR | BOOL | R | Read channel status words in progress | %MWr.m.c.0.0 |
| CMD_IN_PROGR | BOOL | R | Command parameters exchange in progress | %MWr.m.c.0.1 |
| ADJ_IN_PROGR | BOOL | R | Adjust parameters exchange in progress | %MWr.m.c.0.2 |
| TRF_IN_PROGR | BOOL | R | TRF_RECIPE function in progress | %MWr.m.c.0.3 |
| RECONF_IN_PROGR | BOOL | R | Module reconfiguring | %MWr.m.c.0.15 |

### Explicit Exchange Report: EXCH_RPT

The following table introduces the report bits: `EXCH_RPT` (`%MWr.m.c.1`). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_ERR | BOOL | R | Channel status words read fault (1 = failure) | %MWr.m.c.1.0 |
| CMD_ERR | BOOL | R | Fault during exchange of command parameters (1 = failure) | %MWr.m.c.1.1 |
| ADJ_ERR | BOOL | R | Fault during an exchange of adjustment parameters (1 = failure) | %MWr.m.c.1.2 |
| TRF_ERR | BOOL | R | Fault while TRF_RECIPE function is in progress | %MWr.m.c.1.3 |
| RECONF_ERR | BOOL | R | Fault during channel reconfiguration (1 = failure) | %MWr.m.c.1.15 |

## Channel Fault Word

The following table introduces the channel fault bits: `CH_FLT`. These variables are updated by a `READ_STS` (`IODDT_VAR1`).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| EXT_FLT0 | BOOL | R | External fault 0: drive fault | %MWr.m.c.2.0 |
| EXT_FLT1 | BOOL | R | External fault 1: communication fault with the axis | %MWr.m.c.2.1 |
| EXT_FLT2 | BOOL | R | External fault 2 | %MWr.m.c.2.3 |
| INT_FLT | BOOL | R | Internal fault | %MWr.m.c.2.4 |
| CONF_FLT | BOOL | R | Configuration fault: different hardware and software configurations | %MWr.m.c.2.5 |
| COM_FLT | BOOL | R | Communication fault | %MWr.m.c.2.6 |
| APPLI_FLT | BOOL | R | Application fault: configuration, adjustment or command fault | %MWr.m.c.2.7 |
| FAN_STOPPED | BOOL | R | Fan fault (channel 0 only) | %MWr.m.c.2.8 |
| OVER_TEMP | BOOL | R | Overtemperature (channel 0 only) | %MWr.m.c.2.9 |
| SENSOR_FLT | BOOL | R | Temperature sensor fault (channel 0 only) | %MWr.m.c.2.10 |
| PROCESS_CONF | BOOL | R | Creation of move in progress object | %MWr.m.c.2.11 |
| PROCESS_CONF_FAILED | BOOL | R | Configuration fault (except for channel 0) | %MWr.m.c.2.12 |

## Objects of the TRF_RECIPE Function

The following table introduces the objects associated with the `TRF_RECIPE` function. These objects are automatically updated by the system each time the `TRF_RECIPE` function is used.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_TRF | INT | R | Read error for the `TRF_RECIPE` function | %MWr.m.c.3 |
| RETURN_TRF_1 | DINT | R | Return 1 of the `TRF_RECIPE` function | %MDr.m.c.4 |
| RETURN_TRF_2 | REAL | R | Return 2 of the `TRF_RECIPE` function | %MFr.m.c.6 |
| RETURN_TRF_3 | REAL | R | Return 3 of the `TRF_RECIPE` function | %MFr.m.c.8 |
| ACTION_TRF | INT | R | Action to be carried out by the `TRF_RECIPE` function | %MWr.m.c.10 |
| PARAM_TRF_1 | DINT | R | Parameter 1 of the `TRF_RECIPE` function | %MDr.m.c.11 |
| PARAM_TRF_2 | DINT | R | Parameter 2 of the `TRF_RECIPE` function | %MDr.m.c.13 |
| PARAM_TRF_3 | REAL | R | Parameter 3 of the `TRF_RECIPE` function | %MFr.m.c.15 |
| PARAM_TRF_4 | REAL | R | Parameter 4 of the `TRF_RECIPE` function | %MFr.m.c.17 |

## WRITE_CMD Interface Words

The following table explains the meanings of the variables associated with the `WRITE_CMD` whose action was specified in the word `ACTION_CMD`. These variables are updated by a `WRITE_CMD` (`IODDT_VAR1`).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_CMD | INT | RW | Error during WRITE_CMD | %MWr.m.c.19 |
| RETURN_CMD_1 | DINT | RW | Return 1 of the function | %MDr.m.c.20 |
| RETURN_CMD_2 | REAL | RW | Return 2 of the function | %MFr.m.c.22 |
| RETURN_CMD_3 | REAL | RW | Return 3 of the function | %MFr.m.c.24 |
| ACTION_CMD | INT | RW | Action to be carried out | %MWr.m.c.26 |
| PARAM_CMD_1 | DINT | RW | Parameter 1 | %MDr.m.c.27 |
| PARAM_CMD_2 | DINT | RW | Parameter 2 | %MDr.m.c.29 |
| PARAM_CMD_3 | REAL | RW | Parameter 3 | %MFr.m.c.31 |
| PARAM_CMD_4 | REAL | RW | Parameter 4 | %MFr.m.c.33 |

## READ_PARAM, WRITE_PARAM Interface Words

The table below explains the meanings of the parameters which can be accessed using the READ_PARAM and WRITE_PARAM functions for channels 1 to 16. These variables are updated using `READ_PARAM` (`IODDT_VAR1`) or `WRITE_PARAM` (`IODDT_VAR1`). You can also use the `SAVE_PARAM` and `RESTORE_PARAM` functions.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CYCLE_TIME | INT | RW | SERCOSring cycle time | %MWr.m.c.35 |
| BAUD_RATE | INT | RW | Baud rate on SERCOS bus (in Bauds) | %MWr.m.c.36 |
| OPTICAL_POWER | INT | RW | Optical power in the fiber | %MWr.m.c.37 |

## Details Concerning T_CSY_TRF-Type IODDT Explicit Exchange Objects

### At a Glance

This part introduces `T_CSY_TRF` IODDT explicit exchange objects that apply to the TSX CSY 85 module for channels 1 to 32. It groups the word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_TRF`.

### Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- All bits are not used.

### Execution Flag for the TRF_RECIPE Function

The following table introduces the control bit that indicates if the `TRF_RECIPE` is in progress.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| TRF_IN_PROGR | BOOL | R | TRF_RECIPE function in progress | %MWr.m.c.0.3 |

### Explicit Exchange Report: EXCH_RPT

The table below introduces the report bit specific to the `TRF_RECIPE` function.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| TRF_ERR | BOOL | R | Fault while TRF_RECIPE function is in progress | %MWr.m.c.1.3 |

## Objects of the TRF_RECIPE Function

The following table introduces the objects associated with the `TRF_RECIPE` function. These objects are automatically updated by the system each time the `TRF_RECIPE` function is used.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_TRF | INT | R | Read error for the `TRF_RECIPE` function | %MWr.m.c.3 |
| RETURN_TRF_1 | DINT | R | Return 1 of the `TRF_RECIPE` function | %MDr.m.c.4 |
| RETURN_TRF_2 | REAL | R | Return 2 of the `TRF_RECIPE` function | %MFr.m.c.6 |
| RETURN_TRF_3 | REAL | R | Return 3 of the `TRF_RECIPE` function | %MFr.m.c.8 |
| ACTION_TRF | INT | R | Action to be carried out by the `TRF_RECIPE` function | %MWr.m.c.10 |
| PARAM_TRF_1 | DINT | R | Parameter 1 of the `TRF_RECIPE` function | %MDr.m.c.11 |
| PARAM_TRF_2 | DINT | R | Parameter 2 of the `TRF_RECIPE` function | %MDr.m.c.13 |
| PARAM_TRF_3 | REAL | R | Parameter 3 of the `TRF_RECIPE` function | %MFr.m.c.15 |
| PARAM_TRF_4 | REAL | R | Parameter 4 of the `TRF_RECIPE` function | %MFr.m.c.17 |

## Details Concerning T_CSY_IND-Type IODDT Implicit Exchange Objects

### At a Glance

The `T_CSY_IND` IODDT possesses implicit exchange objects, which are described below. This type of IODDT is applied to the `TSX_CSY_85` module for channels 1 to 16.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_IND`.

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

### List of Implicit Exchange Input Objects

The following table introduces the implicit exchange input objects of the `T_CSY_IND` IODDT which apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CH_ERROR | EBOOL | R | Channel error bit. | %Ir.m.c.ERR |
| RAMPING | EBOOL | R | Indicates whether the axis is accelerating or decelerating | %Ir.m.c.0 |
| STEADY | EBOOL | R | The speed is steady | %Ir.m.c.1 |
| STOPPING | EBOOL | R | The movement is decelerating to a stop | %Ir.m.c.2 |
| PROFILE_END | EBOOL | R | The last profile command has been sent to the module | %Ir.m.c.3 |
| IN_POSITION | EBOOL | R | The axis is within the in-position band | %Ir.m.c.4 |
| AXIS_HOMING | EBOOL | R | The axis is homing. For an imaginary axis, this bit is inactive | %Ir.m.c.5 |
| AXIS_HOMED | EBOOL | R | The axis position reading is referenced off the home position | %Ir.m.c.6 |
| AXIS_NOT_FOLLOWING | EBOOL | R | The drive is not recognizing module commands | %Ir.m.c.7 |
| HOLDING | EBOOL | R | The axis is holding in wait position | %Ir.m.c.8 |
| RESUMING | EBOOL | R | The axis is moving after a hold | %Ir.m.c.9 |
| DRIVE_ENABLED | EBOOL | R | The servodrive is enabled | %Ir.m.c.10 |
| DRIVE_DIAG | EBOOL | R | The drive is performing a class 3 diagnostic | %Ir.m.c.11 |
| DRIVE_WARNING | EBOOL | R | The drive is performing a class 2 diagnostic | %Ir.m.c.12 |
| DRIVE_FLT | EBOOL | R | The drive is performing a class 1 diagnostic | %Ir.m.c.13 |
| DRIVE_DISABLED | EBOOL | R | The drive is disabled | %Ir.m.c.14 |
| AXIS_SUMMARY_FLT | EBOOL | R | Drive fault | %Ir.m.c.15 |
| AXIS_COM_OK | EBOOL | R | Communication between the drive and the module is OK | %Ir.m.c.16 |
| AXIS_IS_LINKED | EBOOL | R | The axis belongs to a set of axes | %Ir.m.c.17 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| AXIS_IN_CMD | EBOOL | R | The axis is active and can be controlled | %Ir.m.c.18 |
| AXIS_AT_TARGET | EBOOL | R | The axis is within the in-position band for the target position | %Ir.m.c.20 |
| AXIS_POS_LIMIT | EBOOL | R | The axis has reached the positive limit | %Ir.m.c.21 |
| AXIS_NEG_LIMIT | EBOOL | R | The axis has reached the negative limit | %Ir.m.c.22 |
| AXIS_WARNING | EBOOL | R | MotionWarning status returned by the drive | %Ir.m.c.23 |
| BIAS_REMAIN | EBOOL | R | Offset added to the command position | %Ir.m.c.24 |
| AXIS_MANUAL_MODE | EBOOL | R | Axis operating in Manual Mode | %Ir.m.c.25 |
| DRIVE_REALTIME_BIT1 | EBOOL | R | Drive real time bit 1 | %Ir.m.c.26 |
| DRIVE_REALTIME_BIT2 | EBOOL | R | Drive real time bit 2 | %Ir.m.c.27 |
| AXIS_HOLD | EBOOL | R | The axis is stopped and waiting for a command | %Ir.m.c.28 |
| AXIS_HALT | EBOOL | R | The axis has stopped | %Ir.m.c.29 |
| AXIS_FASTSTOP | EBOOL | R | The axis has faststopped | %Ir.m.c.30 |
| AXIS_READY | EBOOL | R | The axis is ready to respond to a command | %Ir.m.c.31 |
| CONF_OK | EBOOL | R | The channel is configured | %Ir.m.c.32 |

## List of Implicit Exchange Output Objects

The following table introduces the implicit exchange output objects of the `T_CSY_IND` IODDT that apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CONTROL_ACQUIRE | EBOOL | RW | control acquisition | %Qr.m.c.2 |
| CONTROL_JOG_POS | EBOOL | RW | Manual Mode: visible movement command in positive direction along axis | %Qr.m.c.4 |
| CONTROL_JOG_NEG | EBOOL | RW | Manual Mode: visible movement command in negative direction along axis | %Qr.m.c.5 |
| REAL_TIME_CTRL_BIT1 | EBOOL | RW | Drive bit | %Qr.m.c.6 |
| REAL_TIME_CTRL_BIT2 | EBOOL | RW | Drive bit | %Qr.m.c.7 |
| OPERATION_MODE_1 | EBOOL | RW | Selection of operating mode | %Qr.m.c.8 |
| OPERATION_MODE_2 | EBOOL | RW | Selection of operating mode | %Qr.m.c.9 |
| CONTROL_ENABLE | EBOOL | RW | Control enable | %Qr.m.c.10 |
| CONTROL_FOLLOW | EBOOL | RW | Follow control for an axis or a set of follower axes | %Qr.m.c.11 |
| CONTROL_RESUME | EBOOL | RW | Resumes control after a stop | %Qr.m.c.12 |
| CONTROL_INC_POS | EBOOL | RW | Manual Mode: incremental movement command in positive direction along axis | %Qr.m.c.13 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CONTROL_INC_NEG | EBOOL | RW | Manual Mode: incremental movement command in negative direction along axis | %Qr.m.c.14 |
| CONTROL_CLEAR_FLT | EBOOL | RW | Fault clear control | %Qr.m.c.15 |
| ALLOW_ACQUIRE | EBOOL | RW | Acquisition enable control | %Qr.m.c.18 |
| ALLOW_ENABLE | EBOOL | RW | Disable axis control | %Qr.m.c.26 |
| ALLOW_FOLLOW | EBOOL | RW | Control to cancel following for an axis or a set of follower axes | %Qr.m.c.27 |
| ALLOW_RESUME | EBOOL | RW | Authorizes a movement to continue after a stop using the HOLD command | %Qr.m.c.28 |
| ALLOW_MOVE | EBOOL | RW | Authorizes a movement to continue after a stop using the HALT command | %Qr.m.c.29 |
| ALLOW_NOT_FASTSTOP | EBOOL | RW | Control after a Faststop | %Qr.m.c.30 |
| ALLOW_NOT_FLT | EBOOL | RW | Enable fault control | %Qr.m.c.31 |

## Actual Position

The following table introduces the implicit exchange input real of the `T_CSY_IND` IODDT which applies to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| POSITION | REAL | R | Actual position | %IFr.m.c.0 |

## Parameter Report Word

The following table introduces the implicit exchange input word for the `T_CSY_IND` IODDT that applies to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| PARAM_RPT | INT | R | Parameters report signaling a programming fault. The least significant byte contains the error code and the most significant byte contains the address in the registers of the field that triggered the error. | %IWr.m.c.2 |

## Simulated Position

The following table introduces the implicit exchange double output word of the `T_CSY_IND` IODDT which applies to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| REMOTE_POSITION | DINT | RW | Real and imaginary axis: position increment in manual mode.<br>Remote axis: simulated position | %QDr.m.c.0 |

## Details Concerning T_CSY_IND-Type IODDT Explicit Exchange Objects

### At a Glance

This part introduces IODDT explicit exchange objects of the `T_CSY_IND` type that apply to the TSX CSY 85 module for channels 1 to16. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_IND`.

### Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- Not all bits are used.

### Indicators of Explicit Exchange Execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: `EXCH_STS` (`%MWr.m.c.0`). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_IN_PROGR | BOOL | R | Read channel status words in progress | %MWr.m.c.0.0 |
| CMD_IN_PROGR | BOOL | R | Command parameters exchange in progress | %MWr.m.c.0.1 |
| ADJ_IN_PROGR | BOOL | R | Adjust parameters exchange in progress | %MWr.m.c.0.2 |
| TRF_IN_PROGR | BOOL | R | TRF_RECIPE in progress | %MWr.m.c.0.3 |
| RECONF_IN_PROGR | BOOL | R | Module reconfiguring | %MWr.m.c.0.15 |

### Explicit Exchange Report: EXCH_RPT

The following table introduces the report bits: `EXCH_RPT` (`%MWr.m.c.1`). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_ERR | BOOL | R | Channel status words read fault (1 = failure) | %MWr.m.c.1.0 |
| CMD_ERR | BOOL | R | Fault during exchange of command parameters (1 = failure) | %MWr.m.c.1.1 |
| ADJ_ERR | BOOL | R | Fault during an exchange of adjustment parameters (1 = failure) | %MWr.m.c.1.2 |
| TRF_ERR | BOOL | R | Error while sending a TRF_RECIPE on the channel | %MWr.m.c.1.3 |
| RECONF_ERR | BOOL | R | Fault during channel reconfiguration (1 = failure) | %MWr.m.c.1.15 |

## Channel Fault Word

The following table introduces the channel fault bits: CH_FLT. These variables are updated by a READ_STS (IODDT_VAR1).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| EXT_FLT0 | BOOL | R | External fault 0: drive fault | %MWr.m.c.2.0 |
| EXT_FLT1 | BOOL | R | External fault 1: communication fault with the axis | %MWr.m.c.2.1 |
| EXT_FLT2 | BOOL | R | External fault 2 | %MWr.m.c.2.3 |
| INT_FLT | BOOL | R | Internal fault | %MWr.m.c.2.4 |
| CONF_FLT | BOOL | R | Configuration fault: different hardware and software configurations | %MWr.m.c.2.5 |
| COM_FLT | BOOL | R | Communication fault | %MWr.m.c.2.6 |
| APPLI_FLT | BOOL | R | Application fault: configuration, adjustment or command fault | %MWr.m.c.2.7 |
| PROCESS_CONF | BOOL | R | Creation of move in progress object | %MWr.m.c.2.11 |
| PROCESS_CONF_FAILED | BOOL | R | Configuration fault (except for channel 0) | %MWr.m.c.2.12 |

## Objects of the TRF_RECIPE Function

The following table introduces the objects associated with the TRF_RECIPE function. These objects are automatically updated by the system each time the TRF_RECIPE function is used.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_TRF | INT | R | Read error for the TRF_RECIPE function | %MWr.m.c.3 |
| RETURN_TRF_1 | DINT | R | Return 1 of the TRF_RECIPE function | %MDr.m.c.4 |
| RETURN_TRF_2 | REAL | R | Return 2 of the TRF_RECIPE function | %MFr.m.c.6 |
| RETURN_TRF_3 | REAL | R | Return 3 of the TRF_RECIPE function | %MFr.m.c.8 |
| ACTION_TRF | INT | R | Action to be carried out by the TRF_RECIPE function | %MWr.m.c.10 |
| PARAM_TRF_1 | DINT | R | Parameter 1 of the TRF_RECIPE function | %MDr.m.c.11 |
| PARAM_TRF_2 | DINT | R | Parameter 2 of the TRF_RECIPE function | %MDr.m.c.13 |
| PARAM_TRF_3 | REAL | R | Parameter 3 of the TRF_RECIPE function | %MFr.m.c.15 |
| PARAM_TRF_4 | REAL | R | Parameter 4 of the TRF_RECIPE function | %MFr.m.c.17 |

## WRITE_CMD Interface Words

The following table explains the meanings of the variables associated with the WRITE_CMD whose action was specified in the word ACTION_CMD. These variables are updated by a WRITE_CMD (IODDT_VAR1).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_CMD | INT | RW | Error during WRITE_CMD | %MWr.m.c.19 |
| RETURN_CMD_1 | DINT | RW | Return 1 of the function | %MDr.m.c.20 |
| RETURN_CMD_2 | REAL | RW | Return 2 of the function | %MFr.m.c.22 |
| RETURN_CMD_3 | REAL | RW | Return 3 of the function | %MFr.m.c.24 |
| ACTION_CMD | INT | RW | Action to be carried out | %MWr.m.c.26 |
| PARAM_CMD_1 | DINT | RW | Parameter 1 | %MDr.m.c.27 |
| PARAM_CMD_2 | DINT | RW | Parameter 2 | %MDr.m.c.29 |
| PARAM_CMD_3 | REAL | RW | Parameter 3 | %MFr.m.c.31 |
| PARAM_CMD_4 | REAL | RW | Parameter 4 | %MFr.m.c.33 |

## READ_PARAM, WRITE_PARAM Interface Words

The table below explains the meanings of the parameters which can be accessed using the READ_PARAM and WRITE_PARAM functions for channels 1 to 16. These variables are updated using READ_PARAM (IODDT_VAR1) or WRITE_PARAM (IODDT_VAR1). You can also use the SAVE_PARAM and RESTORE_PARAM functions.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| FUNCTION_VALIDATION | INT | RW | Word containing the selective validation bits | %MWr.m.c.35 |
| ACCEL | REAL | RW | Acceleration value | %MFr.m.c.36 |
| DECEL | REAL | RW | Deceleration value | %MFr.m.c.38 |
| ACCEL_TYPE | INT | RW | Acceleration type | %MWr.m.c.40 |
| IN_POSITION_BAND | REAL | RW | Value of the in-position band | %MFr.m.c.41 |
| ENABLE_POSITION_BAND | REAL | RW | Value of the monitoring window | %MFr.m.c.43 |
| ROLLOVER_MAX | REAL | RW | Maximum rollover | %MFr.m.c.45 |
| ROLLOVER_MIN | REAL | RW | Minimum rollover | %MFr.m.c.47 |
| ACCEL_MAX | REAL | RW | Maximum acceleration | %MFr.m.c.49 |
| DECEL_MAX | REAL | RW | Maximum deceleration | %MFr.m.c.51 |
| SPEED_MAX | REAL | RW | Maximum speed | %MFr.m.c.53 |
| POSITION_MAX | REAL | RW | Maximum position | %MFr.m.c.55 |
| POSITION_MIN | REAL | RW | Minimum position | %MFr.m.c.57 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| SCALE_NUMERATOR | REAL | RW | Scale factor enumerator (configuration of an independent axis and GetGearRatio function) | %MFr.m.c.59 |
| SCALE_DENOMINATOR | REAL | RW | Scale factor denominator (configuration of an independent axis and GetGearRatio function) | %MFr.m.c.61 |
| ACCEL_UNIT | INT | RW | Acceleration unit. | %MWr.m.c.63 |
| SPEED_UNIT | INT | RW | Velocity unit | %MWr.m.c.64 |
| POSITION_UNIT | INT | RW | Position unit | %MWr.m.c.65 |

## Details Concerning T_CSY_FOLLOW-Type IODDT Implicit Exchange Objects

### At a Glance

The `T_CSY_FOLLOW` IODDT possesses implicit exchange objects, which are described below. This type of IODDT is applied to the `TSX_CSY_85` module for channels 21 to 24.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_FOLLOW`.

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

### List of Implicit Exchange Input Objects

The following table introduces the implicit exchange input objects of the `T_CSY_FOLLOW` IODDT which apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CH_ERROR | EBOOL | R | Channel error bit. | %Ir.m.c.ERR |
| RAMPING | EBOOL | R | Indicates whether the axis is accelerating or decelerating | %Ir.m.c.0 |
| STEADY | EBOOL | R | The speed is steady | %Ir.m.c.1 |
| STOPPING | EBOOL | R | The movement is decelerating to a stop | %Ir.m.c.2 |
| PROFILE_END | EBOOL | R | The last profile command has been sent to the module | %Ir.m.c.3 |
| IN_POSITION | EBOOL | R | The axis is within the in-position band | %Ir.m.c.4 |
| AXIS_HOMING | EBOOL | R | The axis is homing. For an imaginary axis, this bit is inactive | %Ir.m.c.5 |
| AXIS_HOMED | EBOOL | R | The axis position reading is referenced off the home position | %Ir.m.c.6 |
| AXIS_NOT_FOLLOWING | EBOOL | R | The drive is not recognizing module commands | %Ir.m.c.7 |
| HOLDING | EBOOL | R | The axis is holding in wait position | %Ir.m.c.8 |
| RESUMING | EBOOL | R | The axis is moving after a hold | %Ir.m.c.9 |
| DRIVE_ENABLED | EBOOL | R | The servodrive is enabled | %Ir.m.c.10 |
| DRIVE_DIAG | EBOOL | R | The drive is performing a class 3 diagnostic | %Ir.m.c.11 |
| DRIVE_WARNING | EBOOL | R | The drive is performing a class 2 diagnostic | %Ir.m.c.12 |
| DRIVE_FLT | EBOOL | R | The drive is performing a class 1 diagnostic | %Ir.m.c.13 |
| DRIVE_DISABLED | EBOOL | R | The drive is disabled | %Ir.m.c.14 |
| AXIS_SUMMARY_FLT | EBOOL | R | Drive fault | %Ir.m.c.15 |
| AXIS_COM_OK | EBOOL | R | Communication between the drive and the module is OK | %Ir.m.c.16 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| AXIS_IS_LINKED | EBOOL | R | The axis belongs to a set of axes | %Ir.m.c.17 |
| AXIS_IN_CMD | EBOOL | R | The axis is active and can be controlled | %Ir.m.c.18 |
| AXIS_AT_TARGET | EBOOL | R | The axis is within the in-position band for the target position | %Ir.m.c.20 |
| AXIS_POS_LIMIT | EBOOL | R | The axis has reached the positive limit | %Ir.m.c.21 |
| AXIS_NEG_LIMIT | EBOOL | R | The axis has reached the negative limit | %Ir.m.c.22 |
| AXIS_WARNING | EBOOL | R | MotionWarning status returned by the drive | %Ir.m.c.23 |
| AXIS_HOLD | EBOOL | R | The axis is stopped and waiting for a command | %Ir.m.c.28 |
| AXIS_HALT | EBOOL | R | The axis has stopped | %Ir.m.c.29 |
| AXIS_FASTSTOP | EBOOL | R | The axis has faststopped | %Ir.m.c.30 |
| AXIS_READY | EBOOL | R | The axis is ready to respond to a command | %Ir.m.c.31 |
| CONF_OK | EBOOL | R | The channel is configured | %Ir.m.c.32 |

## List of Implicit Exchange Output Objects

The following table introduces the implicit exchange output objects of the `T_CSY_FOLLOW` IODDT that apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CONTROL_ACQUIRE | EBOOL | RW | control acquisition | %Qr.m.c.2 |
| CONTROL_ENABLE | EBOOL | RW | Control enable | %Qr.m.c.10 |
| CONTROL_FOLLOW | EBOOL | RW | Follow control for an axis or a set of follower axes | %Qr.m.c.11 |
| CONTROL_RESUME | EBOOL | RW | Resumes control after a stop | %Qr.m.c.12 |
| CONTROL_CLEAR_FLT | EBOOL | RW | Fault clear control | %Qr.m.c.15 |
| ALLOW_ACQUIRE | EBOOL | RW | Acquisition enable control | %Qr.m.c.18 |
| ALLOW_ENABLE | EBOOL | RW | Disable axis control | %Qr.m.c.26 |
| ALLOW_FOLLOW | EBOOL | RW | Control to cancel following for an axis or a set of follower axes | %Qr.m.c.27 |
| ALLOW_RESUME | EBOOL | RW | Authorizes a movement to continue after a stop using the HOLD command | %Qr.m.c.28 |
| ALLOW_MOVE | EBOOL | RW | Authorizes a movement to continue after a stop using the HALT command | %Qr.m.c.29 |
| ALLOW_NOT_FASTSTOP | EBOOL | RW | Control after a Faststop | %Qr.m.c.30 |
| ALLOW_NOT_FLT | EBOOL | RW | Enable fault control | %Qr.m.c.31 |

## Actual Position

The following table introduces the implicit exchange input objects of the `T_CSY_FOLLOW` IODDT that apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| POSITION | REAL | R | Actual position | %IFr.m.c.0 |

## Parameter Report Word

The following table introduces the implicit exchanges entry word for the `T_CSY_FOLLOW` IODDT that applies to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| PARAM_RPT | INT | R | Parameters report signaling a programming fault. The least significant byte contains the error code and the most significant byte contains the address in the registers of the field that triggered the error. | %IWr.m.c.2 |

## Details Concerning T_CSY_FOLLOW-Type IODDT Explicit Exchange Objects

### At a Glance

This part introduces the explicit exchange objects of the `T_CSY_FOLLOW` IODDT that apply to the TSX CSY 85 module for channels 21 to 24. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_FOLLOW`.

### Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- Not all bits are used.

### Indicators of Explicit Exchange Execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: `EXCH_STS` (`%MWr.m.c.0`). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_IN_PROGR | BOOL | R | Read channel status words in progress | %MWr.m.c.0.0 |
| CMD_IN_PROGR | BOOL | R | Command parameters exchange in progress | %MWr.m.c.0.1 |
| ADJ_IN_PROGR | BOOL | R | Adjust parameters exchange in progress | %MWr.m.c.0.2 |
| TRF_IN_PROG | BOOL | R | TRF_RECIPE in progress | %MWr.m.c.0.3 |
| RECONF_IN_PROGR | BOOL | R | Module reconfiguring | %MWr.m.c.0.15 |

### Explicit Exchange Report: EXCH_RPT

The following table introduces the report bits: `EXCH_RPT` (%MWr.m.c.1). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_ERR | BOOL | R | Channel status words read fault (1 = failure) | %MWr.m.c.1.0 |
| CMD_ERR | BOOL | R | Fault during exchange of command parameters (1 = failure) | %MWr.m.c.1.1 |
| ADJ_ERR | BOOL | R | Fault during an exchange of adjustment parameters (1 = failure) | %MWr.m.c.1.2 |
| TRF_ERR | BOOL | R | Error while sending a TRF_RECIPE on the channel | %MWr.m.c.1.3 |
| RECONF_ERR | BOOL | R | Fault during channel reconfiguration (1 = failure) | %MWr.m.c.1.15 |

## Channel Fault Word

The following table introduces the channel fault bits: `CH_FLT`. These variables are updated by a `READ_STS` (`IODDT_VAR1`).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| EXT_FLT0 | BOOL | R | External fault 0: drive fault | %MWr.m.c.2.0 |
| EXT_FLT1 | BOOL | R | External fault 1: communication fault with the axis | %MWr.m.c.2.1 |
| EXT_FLT2 | BOOL | R | External fault 2 | %MWr.m.c.2.3 |
| INT_FLT | BOOL | R | Internal fault | %MWr.m.c.2.4 |
| CONF_FLT | BOOL | R | Configuration fault: different hardware and software configurations | %MWr.m.c.2.5 |
| COM_FLT | BOOL | R | Communication fault | %MWr.m.c.2.6 |
| APPLI_FLT | BOOL | R | Application fault: configuration, adjustment or command fault | %MWr.m.c.2.7 |
| PROCESS_CONF | BOOL | R | Creation of move in progress object | %MWr.m.c.2.11 |
| PROCESS_CONF_FAILED | BOOL | R | Configuration fault (except for channel 0) | %MWr.m.c.2.12 |

## Objects of the TRF_RECIPE Function

The following table introduces the objects associated with the `TRF_RECIPE` function. These objects are automatically updated by the system each time the `TRF_RECIPE` function is used.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_TRF | INT | R | Read error for the `TRF_RECIPE` function | %MWr.m.c.3 |
| RETURN_TRF_1 | DINT | R | Return 1 of the `TRF_RECIPE` function | %MDr.m.c.4 |
| RETURN_TRF_2 | REAL | R | Return 2 of the `TRF_RECIPE` function | %MFr.m.c.6 |
| RETURN_TRF_3 | REAL | R | Return 3 of the `TRF_RECIPE` function | %MFr.m.c.8 |
| ACTION_TRF | INT | R | Action to be carried out by the `TRF_RECIPE` function | %MWr.m.c.10 |
| PARAM_TRF_1 | DINT | R | Parameter 1 of the `TRF_RECIPE` function | %MDr.m.c.11 |
| PARAM_TRF_2 | DINT | R | Parameter 2 of the `TRF_RECIPE` function | %MDr.m.c.13 |
| PARAM_TRF_3 | REAL | R | Parameter 3 of the `TRF_RECIPE` function | %MFr.m.c.15 |
| PARAM_TRF_4 | REAL | R | Parameter 4 of the `TRF_RECIPE` function | %MFr.m.c.17 |

## WRITE_CMD Interface Words

The following table explains the meanings of the variables associated with the `WRITE_CMD` whose action was specified in the word `ACTION_CMD`. These variables are updated by a `WRITE_CMD` (`IODDT_VAR1`).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_CMD | INT | RW | Error during WRITE_CMD | %MWr.m.c.19 |
| RETURN_CMD_1 | DINT | RW | Return 1 of the function | %MDr.m.c.20 |
| RETURN_CMD_2 | REAL | RW | Return 2 of the function | %MFr.m.c.22 |
| RETURN_CMD_3 | REAL | RW | Return 3 of the function | %MFr.m.c.24 |
| ACTION_CMD | INT | RW | Action to be carried out | %MWr.m.c.26 |
| PARAM_CMD_1 | DINT | RW | Parameter 1 | %MDr.m.c.27 |
| PARAM_CMD_2 | DINT | RW | Parameter 2 | %MDr.m.c.29 |
| PARAM_CMD_3 | REAL | RW | Parameter 3 | %MFr.m.c.31 |
| PARAM_CMD_4 | REAL | RW | Parameter 4 | %MFr.m.c.33 |

## READ_PARAM, WRITE_PARAM Interface Words

The table below explains the meanings of the parameters which can be accessed using the READ_PARAM and WRITE_PARAM functions for channels 21 to 24. These variables are updated using `READ_PARAM` (`IODDT_VAR1`) or `WRITE_PARAM` (`IODDT_VAR1`). You can also use the `SAVE_PARAM` and `RESTORE_PARAM` functions.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| MASTER_CHANNEL | INT | RW | Number of the master axis (1 to 16, N is not accessible | %MWr.m.c.35 |
| SLAVE_CHANNEL_1 | INT | RW | Number of slave axis 1 | %MWr.m.c.36 |
| FOLL_DESCRIPTION_1 | INT | RW | Description of slave axis 1. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions:<br>bits 8, 9 and 10 set to zero = immediate startup<br>bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction<br>bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction<br>bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position<br>bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position | %MWr.m.c.37 |
| FOLL_WHERE_1 | BOOL | R | 0 = controller | %MWr.m.c.37.0 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| FOLL_TYPE_1 | BOOL | R | 0 = ratio mode<br>1 = Cam mode | %MWr.m.c.37.1 |
| FOLL_POSITION_1 | BOOL | R | 0= following the actual position<br>1 = following the commanded position | %MWr.m.c.37.2 |
| FOLL_FOL_ON_HALT_1 | BOOL | R | 1 = stop the follower axis if master/slave link is removed | %MWr.m.c.37.3 |
| FOLL_HALT_MASTER_1 | BOOL | R | 1 = stop the master in the event of a following error | %MWr.m.c.37.6 |
| FOLL_BIAS_REMAIN_1 | BOOL | R | 1 = dynamic offset on position of master | %MWr.m.c.37.7 |
| NUMERATOR_1 | REAL | RW | numerator for slave axis 1 | %MFr.m.c.38 |
| DENOMINATOR_1 | REAL | RW | Denominator for slave axis 1 | %MFr.m.c.40 |
| TRIGGER_POSITION_1 | REAL | RW | Value of trigger for slave axis 1 | %MFr.m.c.42 |
| SLAVE_CHANNEL_2 | INT | RW | Number of slave axis 2 | %MWr.m.c.44 |
| FOLL_DESCRIPTION_2 | INT | RW | Description of slave axis 2. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions:<br>bits 8, 9 and 10 set to zero = immediate startup<br>bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction<br>bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction<br>bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position<br>bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position | %MWr.m.c.45 |
| FOLL_WHERE_2 | BOOL | R | 0 = controller | %MWr.m.c.45.0 |
| FOLL_TYPE_2 | BOOL | R | 0 = ratio mode<br>1 = Cam mode | %MWr.m.c.45.1 |
| FOLL_POSITION_2 | BOOL | R | 0= following the actual position<br>1 = following the commanded position | %MWr.m.c.45.2 |
| FOLL_FOL_ON_HALT_2 | BOOL | R | 1 = stop the follower axis if master/slave link is removed | %MWr.m.c.45.3 |
| FOLL_HALT_MASTER_2 | BOOL | R | 1 = stop the master in the event of a following error | %MWr.m.c.45.6 |
| FOLL_BIAS_REMAIN_2 | BOOL | R | 1 = dynamic offset on position of master | %MWr.m.c.45.7 |
| NUMERATOR_2 | REAL | RW | numerator for slave axis 2 | %MFr.m.c.46 |
| DENOMINATOR_2 | REAL | RW | Denominator for slave axis 2 | %MFr.m.c.48 |
| TRIGGER_POSITION_2 | REAL | RW | Value of trigger for slave axis 2 | %MFr.m.c.50 |
| SLAVE_CHANNEL_3 | INT | RW | Number of slave axis 3 | %MWr.m.c.52 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| FOLL_DESCRIPTION_3 | INT | RW | Description of slave axis 3. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions:<br>bits 8, 9 and 10 set to zero = immediate startup<br>bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction<br>bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction<br>bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position<br>bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position | %MWr.m.c.53 |
| FOLL_WHERE_3 | BOOL | R | 0 = controller | %MWr.m.c.53.0 |
| FOLL_TYPE_3 | BOOL | R | 0 = ratio mode<br>1 = Cam mode | %MWr.m.c.53.1 |
| FOLL_POSITION_3 | BOOL | R | 0= following the actual position<br>1 = following the commanded position | %MWr.m.c.53.2 |
| FOLL_FOL_ON_HALT_3 | BOOL | R | 1 = stop the follower axis if master/slave link is removed | %MWr.m.c.53.3 |
| FOLL_HALT_MASTER_3 | BOOL | R | 1 = stop the master in the event of a following error | %MWr.m.c.53.6 |
| FOLL_BIAS_REMAIN_3 | BOOL | R | 1 = dynamic offset on position of master | %MWr.m.c.53.7 |
| NUMERATOR_3 | REAL | RW | numerator for slave axis 3 | %MFr.m.c.54 |
| DENOMINATOR_3 | REAL | RW | Denominator for slave axis 3 | %MFr.m.c.56 |
| TRIGGER_POSITION_3 | REAL | RW | Value of trigger for slave axis 3 | %MFr.m.c.58 |
| SLAVE_CHANNEL_4 | INT | RW | Number of slave axis 4 | %MWr.m.c.60 |
| FOLL_DESCRIPTION_4 | INT | RW | Description of slave axis 4. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions:<br>bits 8, 9 and 10 set to zero = immediate startup<br>bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction<br>bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction<br>bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position<br>bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position | %MWr.m.c.61 |
| FOLL_WHERE_4 | BOOL | R | 0 = controller | %MWr.m.c.61.0 |
| FOLL_TYPE_4 | BOOL | R | 0 = ratio mode<br>1 = Cam mode | %MWr.m.c.61.1 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| FOLL_POSITION_4 | BOOL | R | 0= following the actual position<br>1 = following the commanded position | %MWr.m.c.61.2 |
| FOLL_FOL_ON_HALT_4 | BOOL | R | 1 = stop the follower axis if master/slave link is removed | %MWr.m.c.61.3 |
| FOLL_HALT_MASTER_4 | BOOL | R | 1 = stop the master in the event of a following error | %MWr.m.c.61.6 |
| FOLL_BIAS_REMAIN_4 | BOOL | R | 1 = dynamic offset on position of master | %MWr.m.c.61.7 |
| NUMERATOR_4 | REAL | RW | numerator for slave axis 4 | %MFr.m.c.62 |
| DENOMINATOR_4 | REAL | RW | Denominator for slave axis 4 | %MFr.m.c.64 |
| TRIGGER_POSITION_4 | REAL | RW | Value of trigger for slave axis 4 | %MFr.m.c.66 |
| SLAVE_CHANNEL_5 | INT | RW | Number of slave axis 5 | %MWr.m.c.68 |
| FOLL_DESCRIPTION_5 | INT | RW | Description of slave axis 5. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions:<br>bits 8, 9 and 10 set to zero = immediate startup<br>bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction<br>bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction<br>bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position<br>bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position | %MWr.m.c.69 |
| FOLL_WHERE_5 | BOOL | R | 0 = controller | %MWr.m.c.69.0 |
| FOLL_TYPE_5 | BOOL | R | 0 = ratio mode<br>1 = Cam mode | %MWr.m.c.69.1 |
| FOLL_POSITION_5 | BOOL | R | 0= following the actual position<br>1 = following the commanded position | %MWr.m.c.69.2 |
| FOLL_FOL_ON_HALT_5 | BOOL | R | 1 = stop the follower axis if master/slave link is removed | %MWr.m.c.69.3 |
| FOLL_HALT_MASTER_5 | BOOL | R | 1 = stop the master in the event of a following error | %MWr.m.c.69.6 |
| FOLL_BIAS_REMAIN_5 | BOOL | R | 1 = dynamic offset on position of master | %MWr.m.c.69.7 |
| NUMERATOR_6 | REAL | RW | numerator for slave axis 6 | %MFr.m.c.70 |
| DENOMINATOR_6 | REAL | RW | Denominator for slave axis 6 | %MFr.m.c.72 |
| TRIGGER_POSITION_6 | REAL | RW | Value of trigger for slave axis 6 | %MFr.m.c.74 |
| SLAVE_CHANNEL_6 | INT | RW | Number of slave axis 6 | %MWr.m.c.76 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| FOLL_DESCRIPTION_6 | INT | RW | Description of slave axis 6. This word is composed of significant bits described below and has names of variables, as well as three unnamed bits that affect startup conditions:<br>bits 8, 9 and 10 set to zero = immediate startup<br>bit 8 set to 1 and 9 and 10 set to zero = master position reaches trigger in negative direction<br>bit 9 set to 1 and bits 8 and 10 set to zero = master position reaches trigger in positive direction<br>bits 8 and 9 set to 1 and bit 10 set to zero = master threshold > position<br>bits 8 and 9 set to zero and bit 10 set to 1 = master threshold < position | %MWr.m.c.77 |
| FOLL_WHERE_6 | BOOL | R | 0 = controller | %MWr.m.c.77.0 |
| FOLL_TYPE_6 | BOOL | R | 0 = ratio mode<br>1 = Cam mode | %MWr.m.c.77.1 |
| FOLL_POSITION_6 | BOOL | R | 0= following the actual position<br>1 = following the commanded position | %MWr.m.c.77.2 |
| FOLL_FOL_ON_HALT_6 | BOOL | R | 1 = stop the follower axis if master/slave link is removed | %MWr.m.c.77.3 |
| FOLL_HALT_MASTER_6 | BOOL | R | 1 = stop the master in the event of a following error | %MWr.m.c.77.6 |
| FOLL_BIAS_REMAIN_6 | BOOL | R | 1 = dynamic offset on position of master | %MWr.m.c.77.7 |
| NUMERATOR_6 | REAL | RW | numerator for slave axis 6 | %MFr.m.c.78 |
| DENOMINATOR_6 | REAL | RW | Denominator for slave axis 6 | %MFr.m.c.80 |
| TRIGGER_POSITION_6 | REAL | RW | Value of trigger for slave axis 6 | %MFr.m.c.82 |

## Details Concerning T_CSY_COORD-Type IODDT Implicit Exchange Objects

### At a Glance

The `T_CSY_COORD` IODDT possesses implicit exchange objects, which are described below. This type of IODDT is applied to the `TSX_CSY_85` module for channels 17 to 20.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_COORD`.

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

### List of Implicit Exchange Input Objects

The following table introduces the implicit exchange input objects of the `T_CSY_COORD` IODDT which apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CH_ERROR | EBOOL | R | Channel error bit. | %Ir.m.c.ERR |
| RAMPING | EBOOL | R | Indicates whether the axis is accelerating or decelerating | %Ir.m.c.0 |
| STEADY | EBOOL | R | The speed is steady | %Ir.m.c.1 |
| STOPPING | EBOOL | R | The movement is decelerating to a stop | %Ir.m.c.2 |
| PROFILE_END | EBOOL | R | The last profile command has been sent to the module | %Ir.m.c.3 |
| IN_POSITION | EBOOL | R | The axis is within the in-position band | %Ir.m.c.4 |
| AXIS_HOMING | EBOOL | R | The axis is homing. For an imaginary axis, this bit is inactive | %Ir.m.c.5 |
| AXIS_HOMED | EBOOL | R | The axis position reading is referenced off the home position | %Ir.m.c.6 |
| AXIS_NOT_FOLLOWING | EBOOL | R | The drive is not recognizing module commands | %Ir.m.c.7 |
| HOLDING | EBOOL | R | The axis is holding in wait position | %Ir.m.c.8 |
| RESUMING | EBOOL | R | The axis is moving after a hold | %Ir.m.c.9 |
| DRIVE_ENABLED | EBOOL | R | The servodrive is enabled | %Ir.m.c.10 |
| DRIVE_DIAG | EBOOL | R | The drive is performing a class 3 diagnostic | %Ir.m.c.11 |
| DRIVE_WARNING | EBOOL | R | The drive is performing a class 2 diagnostic | %Ir.m.c.12 |
| DRIVE_FLT | EBOOL | R | The drive is performing a class 1 diagnostic | %Ir.m.c.13 |
| DRIVE_DISABLED | EBOOL | R | The drive is disabled | %Ir.m.c.14 |
| AXIS_SUMMARY_FLT | EBOOL | R | Drive fault | %Ir.m.c.15 |
| AXIS_COM_OK | EBOOL | R | Communication between the drive and the module is OK | %Ir.m.c.16 |
| AXIS_IS_LINKED | EBOOL | R | The axis belongs to a set of axes | %Ir.m.c.17 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| AXIS_IN_CMD | EBOOL | R | The axis is active and can be controlled | %Ir.m.c.18 |
| AXIS_AT_TARGET | EBOOL | R | The axis is within the in-position band for the target position | %Ir.m.c.20 |
| AXIS_POS_LIMIT | EBOOL | R | The axis has reached the positive limit | %Ir.m.c.21 |
| AXIS_NEG_LIMIT | EBOOL | R | The axis has reached the negative limit | %Ir.m.c.22 |
| AXIS_WARNING | EBOOL | R | MotionWarning status returned by the drive | %Ir.m.c.23 |
| AXIS_HOLD | EBOOL | R | The axis is stopped and waiting for a command | %Ir.m.c.28 |
| AXIS_HALT | EBOOL | R | The axis has stopped | %Ir.m.c.29 |
| AXIS_FASTSTOP | EBOOL | R | The axis has faststopped | %Ir.m.c.30 |
| AXIS_READY | EBOOL | R | The axis is ready to respond to a command | %Ir.m.c.31 |
| CONF_OK | EBOOL | R | The channel is configured | %Ir.m.c.32 |

## List of Implicit Exchange Output Objects

The following table introduces the implicit exchange output objects of the `T_CSY_COORD` IODDT that apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CONTROL_ACQUIRE | EBOOL | RW | control acquisition | %Qr.m.c.2 |
| CONTROL_ENABLE | EBOOL | RW | Control enable | %Qr.m.c.10 |
| CONTROL_FOLLOW | EBOOL | RW | Follow control for an axis or a set of follower axes | %Qr.m.c.11 |
| CONTROL_RESUME | EBOOL | RW | Resumes control after a stop | %Qr.m.c.12 |
| CONTROL_CLEAR_FLT | EBOOL | RW | Fault clear control | %Qr.m.c.15 |
| ALLOW_ACQUIRE | EBOOL | RW | Acquisition enable control | %Qr.m.c.18 |
| ALLOW_ENABLE | EBOOL | RW | Disable axis control | %Qr.m.c.26 |
| ALLOW_FOLLOW | EBOOL | RW | Control to cancel following for an axis or a set of follower axes | %Qr.m.c.27 |
| ALLOW_RESUME | EBOOL | RW | Authorizes a movement to continue after a stop using the HOLD command | %Qr.m.c.28 |
| ALLOW_MOVE | EBOOL | RW | Authorizes a movement to continue after a stop using the HALT command | %Qr.m.c.29 |
| ALLOW_NOT_FASTSTOP | EBOOL | RW | Control after a Faststop | %Qr.m.c.30 |
| ALLOW_NOT_FLT | EBOOL | RW | Enable fault control | %Qr.m.c.31 |

## Parameter Report Word

The following table introduces the implicit exchange input word for the `T_CSY_COORD` IODDT that applies to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| PARAM_RPT | INT | R | Parameters report signaling a programming fault. The least significant byte contains the error code and the most significant byte contains the address in the registers of the field that triggered the error. | %IWr.m.c.2 |

## Details Concerning T_CSY_COORD-Type IODDT Explicit Exchange Objects

### At a Glance

This part introduces the explicit exchange objects of the `T_CSY_COORD` IODDT that apply to the TSX CSY 85 module for channels 17 to 20. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_COORD`.

### Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- Not all bits are used.

### Indicators of Explicit Exchange Execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: `EXCH_STS` (%MWr.m.c.0). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_IN_PROGR | BOOL | R | Read channel status words in progress | %MWr.m.c.0.0 |
| CMD_IN_PROGR | BOOL | R | Command parameters exchange in progress | %MWr.m.c.0.1 |
| ADJ_IN_PROGR | BOOL | R | Adjust parameters exchange in progress | %MWr.m.c.0.2 |
| TRF_IN_PROGR | BOOL | R | TRF_RECIPE in progress | %MWr.m.c.0.3 |
| RECONF_IN_PROGR | BOOL | R | Module reconfiguring | %MWr.m.c.0.15 |

### Explicit Exchange Report: EXCH_RPT

The following table introduces the report bits: `EXCH_RPT` (%MWr.m.c.1). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_ERR | BOOL | R | Channel status words read fault (1 = failure) | %MWr.m.c.1.0 |
| CMD_ERR | BOOL | R | Fault during exchange of command parameters (1 = failure) | %MWr.m.c.1.1 |
| ADJ_ERR | BOOL | R | Fault during an exchange of adjustment parameters (1 = failure) | %MWr.m.c.1.2 |
| TRF_ERR | BOOL | R | Error while sending a TRF_RECIPE on the channel | %MWr.m.c.1.3 |
| RECONF_ERR | BOOL | R | Fault during channel reconfiguration (1 = failure) | %MWr.m.c.1.15 |

## Channel Fault Word

The following table introduces the channel fault bits: CH_FLT. These variables are updated by a READ_STS (IODDT_VAR1).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| EXT_FLT0 | BOOL | R | External fault 0: drive fault | %MWr.m.c.2.0 |
| EXT_FLT1 | BOOL | R | External fault 1: communication fault with the axis | %MWr.m.c.2.1 |
| EXT_FLT2 | BOOL | R | External fault 2 | %MWr.m.c.2.3 |
| INT_FLT | BOOL | R | Internal fault | %MWr.m.c.2.4 |
| CONF_FLT | BOOL | R | Configuration fault: different hardware and software configurations | %MWr.m.c.2.5 |
| COM_FLT | BOOL | R | Communication fault | %MWr.m.c.2.6 |
| APPLI_FLT | BOOL | R | Application fault: configuration, adjustment or command fault | %MWr.m.c.2.7 |
| PROCESS_CONF | BOOL | R | Creation of move in progress object | %MWr.m.c.2.11 |
| PROCESS_CONF_FAILED | BOOL | R | Configuration fault (except for channel 0) | %MWr.m.c.2.12 |

## WRITE_CMD Interface Words

The following table explains the meanings of the variables associated with the WRITE_CMD whose action was specified in the word ACTION_CMD. These variables are updated by a WRITE_CMD (IODDT_VAR1).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_CMD | INT | RW | Error during WRITE_CMD | %MWr.m.c.19 |
| RETURN_CMD_1 | DINT | RW | Return 1 of the function | %MDr.m.c.20 |
| RETURN_CMD_2 | REAL | RW | Return 2 of the function | %MFr.m.c.22 |
| RETURN_CMD_3 | REAL | RW | Return 3 of the function | %MFr.m.c.24 |
| ACTION_CMD | INT | RW | Action to be carried out | %MWr.m.c.26 |
| PARAM_CMD_1 | DINT | RW | Parameter 1 | %MDr.m.c.27 |
| PARAM_CMD_2 | DINT | RW | Parameter 2 | %MDr.m.c.29 |
| PARAM_CMD_3 | REAL | RW | Parameter 3 | %MFr.m.c.31 |
| PARAM_CMD_4 | REAL | RW | Parameter 4 | %MFr.m.c.33 |
| PARAM_CMD_5 | REAL | RW | Parameter 5 | %MFr.m.c.35 |
| PARAM_CMD_6 | REAL | RW | Parameter 6 | %MFr.m.c.37 |
| PARAM_CMD_7 | REAL | RW | Parameter 7 | %MFr.m.c.39 |
| PARAM_CMD_8 | REAL | RW | Parameter 8 | %MFr.m.c.41 |
| PARAM_CMD_9 | REAL | RW | Parameter 9 | %MFr.m.c.43 |

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| PARAM_CMD_10 | REAL | RW | Parameter 10 | %MFr.m.c.45 |
| PARAM_CMD_11 | REAL | RW | Parameter 11 | %MFr.m.c.47 |
| PARAM_CMD_12 | REAL | RW | Parameter 12 | %MFr.m.c.49 |
| PARAM_CMD_13 | REAL | RW | Parameter 13 | %MFr.m.c.51 |
| PARAM_CMD_14 | REAL | RW | Parameter 14 | %MFr.m.c.53 |
| PARAM_CMD_15 | REAL | RW | Parameter 15 | %MFr.m.c.55 |
| PARAM_CMD_16 | REAL | RW | Parameter 16 | %MFr.m.c.57 |
| PARAM_CMD_17 | REAL | RW | Parameter 17 | %MFr.m.c.59 |
| PARAM_CMD_18 | REAL | RW | Parameter 18 | %MFr.m.c.61 |
| AXIS_MEMBER_1 | INT | RW | Coordinated axis member 1 | %MWr.m.c.63 |
| AXIS_MEMBER_2 | INT | RW | Coordinated axis member 2 | %MWr.m.c.64 |
| AXIS_MEMBER_3 | INT | RW | Coordinated axis member 3 | %MWr.m.c.65 |
| AXIS_MEMBER_4 | INT | RW | Coordinated axis member 4 | %MWr.m.c.66 |
| AXIS_MEMBER_5 | INT | RW | Coordinated axis member 5 | %MWr.m.c.67 |
| AXIS_MEMBER_6 | INT | RW | Coordinated axis member 6 | %MWr.m.c.68 |
| AXIS_MEMBER_7 | INT | RW | Coordinated axis member 7 | %MWr.m.c.69 |
| AXIS_MEMBER_8 | INT | RW | Coordinated axis member 8 | %MWr.m.c.70 |

## Details Concerning T_CSY_CAM-Type IODDT Implicit Exchange Objects

### At a Glance

The `T_CSY_CAM` IODDT possesses implicit exchange objects, which are described below. This type of IODDT is applied to the `TSX_CSY_85` module for channels 25 to 31.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_CAM`.

The meaning of bits is generally given for status 1 of this bit. In specific cases, the two bit statuses are explained.

### List of Implicit Exchange Input Objects

The following table introduces the implicit exchange input objects for the `T_CSY_CAM` IODDT which apply to the TSX_CSY_85 module.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| CH_ERROR | EBOOL | R | Channel error bit. | %Ir.m.c.ERR |

## Details Concerning T_CSY_CAM-Type IODDT Explicit Exchange Objects

### At a Glance

This part introduces the explicit exchange objects of the `T_CSY_CAM` IODDT that apply to the TSX CSY 85 module for channels 25 to 31. It groups word-type objects whose bits have a particular meaning. These objects are presented in detail below.

Example of declaration of a variable: `IODDT_VAR1` of the type `T_CSY_CAM`.

### Notes

- The meaning of bits is generally given for status 1 of this bit. In specific cases, each bit status is explained.
- Not all bits are used.

### Indicators of Explicit Exchange Execution: EXCH_STS

The following table introduces the control bits for explicit exchanges: `EXCH_STS` (`%MWr.m.c.0`). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_IN_PROGR | BOOL | R | Read channel status words in progress | %MWr.m.c.0.0 |
| CMD_IN_PROGR | BOOL | R | Command parameters exchange in progress | %MWr.m.c.0.1 |
| ADJ_IN_PROGR | BOOL | R | Adjust parameters exchange in progress | %MWr.m.c.0.2 |
| TRF_IN_PROGR | BOOL | R | TRF_RECIPE function in progress | %MWr.m.c.0.3 |
| RECONF_IN_PROGR | BOOL | R | Module reconfiguring | %MWr.m.c.0.15 |

### Explicit Exchange Report: EXCH_RPT

The following table introduces the report bits: `EXCH_RPT` (`%MWr.m.c.1`). These variables are automatically updated by the system upon each explicit exchange.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| STS_ERR | BOOL | R | Channel status words read fault (1 = failure) | %MWr.m.c.1.0 |
| CMD_ERR | BOOL | R | Fault during exchange of command parameters (1 = failure) | %MWr.m.c.1.1 |
| ADJ_ERR | BOOL | R | Fault during an exchange of adjustment parameters (1 = failure) | %MWr.m.c.1.2 |
| TRF_ERR | BOOL | R | Fault while TRF_RECIPE function is in progress | %MWr.m.c.1.3 |
| RECONF_ERR | BOOL | R | Fault during channel reconfiguration (1 = failure) | %MWr.m.c.1.15 |

## Channel Fault Word

The following table introduces the channel fault bits: `CH_FLT`. These variables are updated by a `READ_STS` (`IODDT_VAR1`).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| EXT_FLT0 | BOOL | R | External fault 0: drive fault | %MWr.m.c.2.0 |
| EXT_FLT1 | BOOL | R | External fault 1: communication fault with the axis | %MWr.m.c.2.1 |
| EXT_FLT2 | BOOL | R | External fault 2 | %MWr.m.c.2.3 |
| INT_FLT | BOOL | R | Internal fault | %MWr.m.c.2.4 |
| CONF_FLT | BOOL | R | Configuration fault: different hardware and software configurations | %MWr.m.c.2.5 |
| COM_FLT | BOOL | R | Communication fault | %MWr.m.c.2.6 |
| APPLI_FLT | BOOL | R | Application fault: configuration, adjustment or command fault | %MWr.m.c.2.7 |
| PROCESS_CONF | BOOL | R | Creation of move in progress object | %MWr.m.c.2.11 |
| PROCESS_CONF_FAILED | BOOL | R | Configuration fault (except for channel 0) | %MWr.m.c.2.12 |

## Objects of the TRF_RECIPE Function

The following table introduces the objects associated with the `TRF_RECIPE` function. These objects are automatically updated by the system each time the `TRF_RECIPE` function is used.

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_TRF | INT | R | Read error for the `TRF_RECIPE` function | %MWr.m.c.3 |
| RETURN_TRF_1 | DINT | R | Return 1 of the `TRF_RECIPE` function | %MDr.m.c.4 |
| RETURN_TRF_2 | REAL | R | Return 2 of the `TRF_RECIPE` function | %MFr.m.c.6 |
| RETURN_TRF_3 | REAL | R | Return 3 of the `TRF_RECIPE` function | %MFr.m.c.8 |
| ACTION_TRF | INT | R | Action to be carried out by the `TRF_RECIPE` function | %MWr.m.c.10 |
| PARAM_TRF_1 | DINT | R | Parameter 1 of the `TRF_RECIPE` function | %MDr.m.c.11 |
| PARAM_TRF_2 | DINT | R | Parameter 2 of the `TRF_RECIPE` function | %MDr.m.c.13 |
| PARAM_TRF_3 | REAL | R | Parameter 3 of the `TRF_RECIPE` function | %MFr.m.c.15 |
| PARAM_TRF_4 | REAL | R | Parameter 4 of the `TRF_RECIPE` function | %MFr.m.c.17 |

## WRITE_CMD Interface Words

The following table explains the meanings of the variables associated with the `WRITE_CMD` whose action was specified in the word `ACTION_CMD`. These variables are updated by a `WRITE_CMD` (`IODDT_VAR1`).

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| ERROR_CMD | INT | RW | Error during WRITE_CMD | %MWr.m.c.19 |
| RETURN_CMD_1 | DINT | RW | Return 1 of the function | %MDr.m.c.20 |
| RETURN_CMD_2 | REAL | RW | Return 2 of the function | %MFr.m.c.22 |
| RETURN_CMD_3 | REAL | RW | Return 3 of the function | %MFr.m.c.24 |
| ACTION_CMD | INT | RW | Action to be carried out | %MWr.m.c.26 |
| PARAM_CMD_1 | DINT | RW | Parameter 1 | %MDr.m.c.27 |
| PARAM_CMD_2 | DINT | RW | Parameter 2 | %MDr.m.c.29 |
| PARAM_CMD_3 | REAL | RW | Parameter 3 | %MFr.m.c.31 |
| PARAM_CMD_4 | REAL | RW | Parameter 4 | %MFr.m.c.33 |

# Section 5.3
# The IODDT Type T_GEN_MOD Applicable to All Modules

## Details of the Language Objects of the T_GEN_MOD-Type IODDT

### Introduction

Modules of Premium PLCs have an associated IODDT of type `T_GEN_MOD`.

### Observations

- In general, the meaning of the bits is given for bit status 1. In specific cases, an explanation is given for each status of the bit.
- Not all bits are used.

### List of Objects

The table below presents the objects of the IODDT:

| Standard symbol | Type | Access | Meaning | Address |
|---|---|---|---|---|
| MOD_ERROR | BOOL | R | Module error bit | %Ir.m.MOD.ERR |
| EXCH_STS | INT | R | Module exchange control word | %MWr.m.MOD.0 |
| STS_IN_PROGR | BOOL | R | Reading of status words of the module in progress | %MWr.m.MOD.0.0 |
| EXCH_RPT | INT | R | Exchange report word | %MWr.m.MOD.1 |
| STS_ERR | BOOL | R | Error detected while reading module status words | %MWr.m.MOD.1.0 |
| MOD_FLT | INT | R | Internal error word of the module | %MWr.m.MOD.2 |
| MOD_FAIL | BOOL | R | Internal error, inoperable module | %MWr.m.MOD.2.0 |
| CH_FLT | BOOL | R | Channel error detected | %MWr.m.MOD.2.1 |
| BLK | BOOL | R | Terminal block error | %MWr.m.MOD.2.2 |
| CONF_FLT | BOOL | R | Hardware or software configuration mismatch | %MWr.m.MOD.2.5 |
| NO_MOD | BOOL | R | Module missing or inoperative | %MWr.m.MOD.2.6 |
| EXT_MOD_FLT | BOOL | R | Internal error word of the module (Fipio extension only) | %MWr.m.MOD.2.7 |
| MOD_FAIL_EXT | BOOL | R | Module is unserviceable (Fipio extension only) | %MWr.m.MOD.2.8 |
| CH_FLT_EXT | BOOL | R | Channel error detected (Fipio extension only) | %MWr.m.MOD.2.9 |
| BLK_EXT | BOOL | R | Terminal block error detected (Fipio extension only) | %MWr.m.MOD.2.10 |
| CONF_FLT_EXT | BOOL | R | Hardware or software configuration mismatch (Fipio extension only) | %MWr.m.MOD.2.13 |
| NO_MOD_EXT | BOOL | R | Module missing or inoperative (Fipio extension only) | %MWr.m.MOD.2.14 |

# Appendices

# Appendix A
# Error Codes

## Error Codes

### At a Glance

Word `%MWr.m.c` (where i is between 21 and 24 for groups of channels) contains the codes for errors that occur following a TRF_RECIPE instruction.

Bit `%MWr.m.c:X3` sends back the status of the current command on channel i.

The interpolation algorithm of the TSX CSY 85 module manages error codes greater than 9500.

Errors are diagnosed as follows:

- Check that the command has been completed: `%MWr.m.c:X3` = 0,
- Check floating point `%MFr.m.c.8` (return_trf_3) to identify the point affected by the error.
- Check the error code in word `%MWr.m.c.3`,
- Use the table below to identify the type of error.
- Modify your program or module configuration to delete this error the next time the TRF_RECIPE instruction is executed.

### Error Codes

The table below describes the possible error codes:

| Code | Description |
|---|---|
| 9501 | For a type 1, 2 or 10 interpolation, one of the parameters ParF1 or ParF2 equals zero. |
| 9502 | The maximum number of points for a trajectory has been reached. The TSX CSY 85 module permits a maximum of 10,000 points. |
| 9503 | More axes have been defined than are permitted. |
| 9504 | Two consecutive identical points have been found in the table for interpolation types other than 12. |
| 9505 | The number of points defined for at least one cam is insufficient in relation to the number of points on the trajectory. |
| 9506 | Use of circular type interpolation although more than two axes have been defined (types 10, 11 and 12). |
| 9507 | A cam corresponding to one of the axes has not been configured. |
| 9508 | Circular link with angle of 180° (type 10) |
| 9509 | Circular link with angle of 0° (type 10) |

| Code | Description |
|---|---|
| 9510 | A trajectory has been defined with a number of points greater than the maximum number permitted:<br>● For firmware version 1.1 or earlier:<br>The maximum number of reference points allowed is 60.<br>● For firmware version 1.2 or later:<br>The maximum number of reference points allowed is 200. |
| 9511 | The radius is less than half the distance between points Pn-1 and Pn. |
| 9512 | Circle impossible. If the type is 11, the start point is the same as the end point. If the type is 12, the start point is the same as the end point and the same as the center of the circle. |
| 9513 | Radius equals 0 (type 11) |
| 9514 | Link too long: Next segment = 0 (types 1, 2 or 10) |
| 9515 | The number of points in the linear segment is set to 0 (types 0, 1 or 10). |
| 9516 | The number of points in the third-degree polynomial interpolation segment is set to 0 (type 1). |
| 9517 | The number of points in the circular interpolation segment is set to 0 (type 10). |
| 9518 | The number of points in the circular interpolation segment is set to 0 (type 11 or 12). |
| 9519 | The center position set in the table differs from the position calculated by the module by more than 50% of the radius of the circle (type 12). |
| 9520 | Group not configured |
| 9521 | At least one of the axes in the group has not been configured. |
| 9522 | The number of points in the fifth-degree polynomial interpolation segment is set to 0 (type 2). |
| 9523 | The number of points in the interpolation table equals zero (first word in the table). |
| 9524 | Not enough memory to calculate interpolation. |
| 9525 | As the length of the next segment is zero, the link cannot be made. |
| 9526 | The master table is empty, so the interpolation calculation has not been made. |
| 9527 | The number of words per point is incorrect in the interpolation table. |
| 9528 | The type of interpolation requested does not exist (parameter type other than 0, 1, 2, 10, 11 or 12). |
| | |
| 9002 | Error code already exists but may occur if the SERCOS ring has not been configured correctly. |

# Index