TML_LIB_CJ1

TECHNOSOFT

Motion Control Library for OMRON CJ1 Series

User Manual

TECHNOSOFT

TML_LIB
User Manual

P091.040.CJ1.UM.1007

Technosoft S.A.

Avenue des Alpes 20
CH-2000 NEUCHATEL
Switzerland
Tel.: +41 (0) 32 732 5500
Fax: +41 (0) 32 732 5504
contact@technosoftmotion.com
www.technosoftmotion.com

Read This First

Whilst Technosoft believes that the information and guidance given in this manual is correct, all parties must rely upon their own skill and judgment when making use of it. Technosoft does not assume any liability to anyone for any loss or damage caused by any error or omission in the work, whether such error or omission is the result of negligence or any other cause. Any and all such liability is disclaimed.

About This Manual

This book describes the motion library **TML_LIB_CJ1.** TML_LIB_CJ1 is a **IEC61131-3 compatible** collection of motion function blocks, which can be integrated in a CX Programmer application for the **OMRON SYSMAC CJ** PLC series**.** With TML_LIB_CJ1 motion library, you can quickly program the desired motion and control the Technosoft intelligent drives and motors (with the drive integrated in the motor case) from an OMRON SYSMAC CJ1 PLC environment. The TML_LIB_CJ1 uses the **User Defined CAN Unit – CJ1W-CORT21**, through which an OMRON PLC is connected with the Technosoft drives/motors via a CAN bus link.

Scope of This Manual

This manual applies to the following Technosoft intelligent drives and motors:

- **IDM240 / IDM640** (models IDM240-5EI, IDM240-5LI, IDM640-8EI and IDM640-8LI), with firmware **F000H** or later (revision letter must be equal or after H i.e. I, J, etc.)
- **IDS240 / IDS640** (all models), with firmware **F000H** or later
- **ISCM4805 / ISCM8005** (all models), with firmware **F000H** or later
- **IBL2403** (all models), with firmware **F020H** or later
- **IPS110** (all models), with firmware **F005H** or later
- **IM23x** (models IS and MA), with firmware **F900H** or later

IMPORTANT! *For correct operation, these drives/motors must be programmed with firmware revision H.* ***EasySetUp***[1] *- Technosoft IDE for drives/motors setup, includes a firmware programmer with which you can check your drive/motor firmware version and revision and if needed, update your drive/motor firmware to revision H.*

[1] **EasySetUp** is included in **TML_LIB_CJ1** installation package as a component of **EasyMotion Studio Demo version**. It can also be downloaded free of charge from Technosoft web page

Notational Conventions

This document uses the following conventions:

- **Drive/motor** - an *intelligent drive* or an *intelligent motor* having the drive part integrated in the motor case
- **TML** – Technosoft Motion Language
- **IU** – drive/motor internal units
- **ACR.5** – bit 5 of ACR data

Related Documentation

MotionChip™ II TML Programming (part no. P091.055.MCII.TML.UM.xxxx) describes in detail TML basic concepts, motion programming, functional description of TML instructions for high level or low level motion programming, communication channels and protocols. Also give a detailed description of each TML instruction including syntax, binary code and examples.

MotionChip II Configuration Setup (part no. P091.055.MCII.STP.UM.xxxx) describes the MotionChip II operation and how to setup its registers and parameters starting from the user application data. This is a technical reference manual for all the MotionChip II registers, parameters and variables.

Help of the EasyMotion Studio software platform – describes how to use the EasyMotion Studio, which support all new features added to revision H of firmware. It includes: motion system setup & tuning wizard, motion sequence programming wizard, testing and debugging tools like: data logging, watch, control panels, on-line viewers of TML registers, parameters and variables, etc.

If you want to …	Contact Technosoft at …
Visit Technosoft online	World Wide Web: http://www.technosoftmotion.com/
Receive general information or assistance	World Wide Web: http://www.technosoftmotion.com/ Email: contact@technosoftmotion.com
Ask questions about product operation or report suspected problems Make suggestions about or report errors in documentation	Fax: (41) 32 732 55 04 Email: hotline@technosoftmotion.com

Contents

1 Introduction

The programming of Technosoft intelligent drives/motors involves 2 steps:

1) Drive/motor setup
2) Motion programming

For **Step 1 – drive/motor setup**, Technosoft provides **EasySetUp**. EasySetUp is an integrated development environment for the setup of Technosoft drives/motors. The output of EasySetUp is a set of *setup data*, which can be downloaded to the drive/motor EEPROM or saved on your PC for later use. The setup data is copied at power-on into the RAM memory of the drive/motor and is used during runtime. The reciprocal is also possible i.e. to retrieve the complete setup data from a drive/motor EEPROM previously programmed. EasySetUp can be downloaded free of charge from Technosoft web page. It is also provided on the TML_LIB_CJ1 installation CD.

For **Step 2 – motion programming**, Technosoft offers multiple options, like:

1) Use the drives/motors embedded motion controller and do the motion programming in Technosoft Motion Language (TML). For this operation Technosoft provides **EasyMotion Studio**, an IDE for both drives setup and motion programming. The output of EasyMotion Studio is a set of setup data and a TML program to download and execute on the drive/motor.
2) Use a **.DLL** with high-level motion functions which can be integrated in a host application written in C/C++, Delphi Pascal, Visual Basic or Labview
3) Use a **IEC61131-3 compatible** library with motion function blocks which can be integrated in a PLC application based on one of the IEC 61136 standard languages
4) Combine option 1) with options 2) or 3) to really distribute the intelligence between the master/host and the drives/motors in complex multi-axis applications. Thus, instead of trying to command each step of an axis movement, you can program the drives/motors using TML to execute complex tasks and inform the master when these are done.

TML_LIB_CJ1 is part of option 3) – a IEC61131-3 compatible motion library developed for **OMRON SYSMAC CJ** family of PLCs. The link between the Technosoft drives/motors and the OMRON PLC is done via CAN-bus using as interface the **CJ1W-CORT21** user defined CAN unit. This solution embeds a Technosoft motion control unit in the OMRON environment, providing simultaneous access to precise motion control and a large number of distributed peripherals (Figure 1.1).

This manual presents how to install and use the components of the **TML_LIB_CJ1** library. These were built following the guidelines described in PLC standards for motion control.

The functionality and the interface of these additional functions comply with the technical specifications of the standard.

Remarks:

- *Option 4) requires using EasyMotion Studio instead of EasySetUp. With EasyMotion Studio you can create high-level motion functions in TML, to be called from your PLC*
- *EasyMotion Studio is also recommended if your application includes a homing as it comes with 32 predefined homing procedures to select from, with possibility to adapt them*

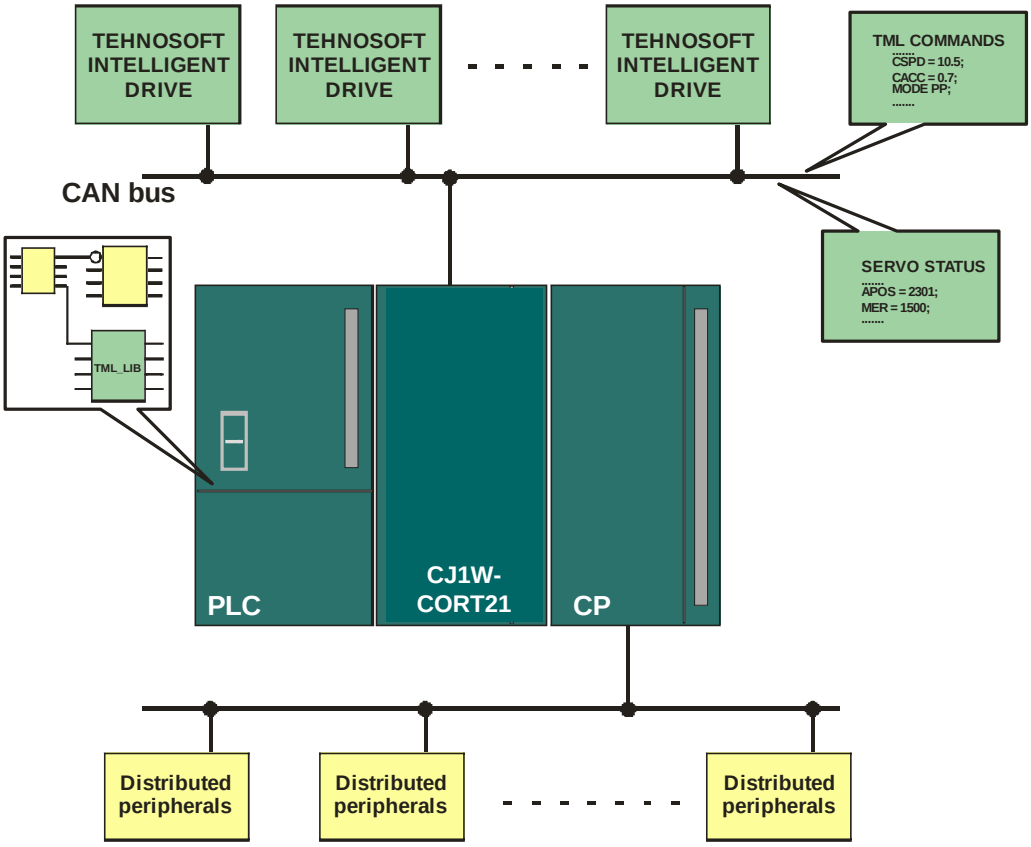


Figure 1.1. *Technosoft drives/motors connections with an OMRON CJ1 PLC*

2 Getting started

2.1 Hardware installation

Figure 2.1 shows how to connect via a CAN-bus link several drives/motors with the OMRON SYSMAC CJ series PLC + the CJ1W-CORT21 user defined CAN unit.

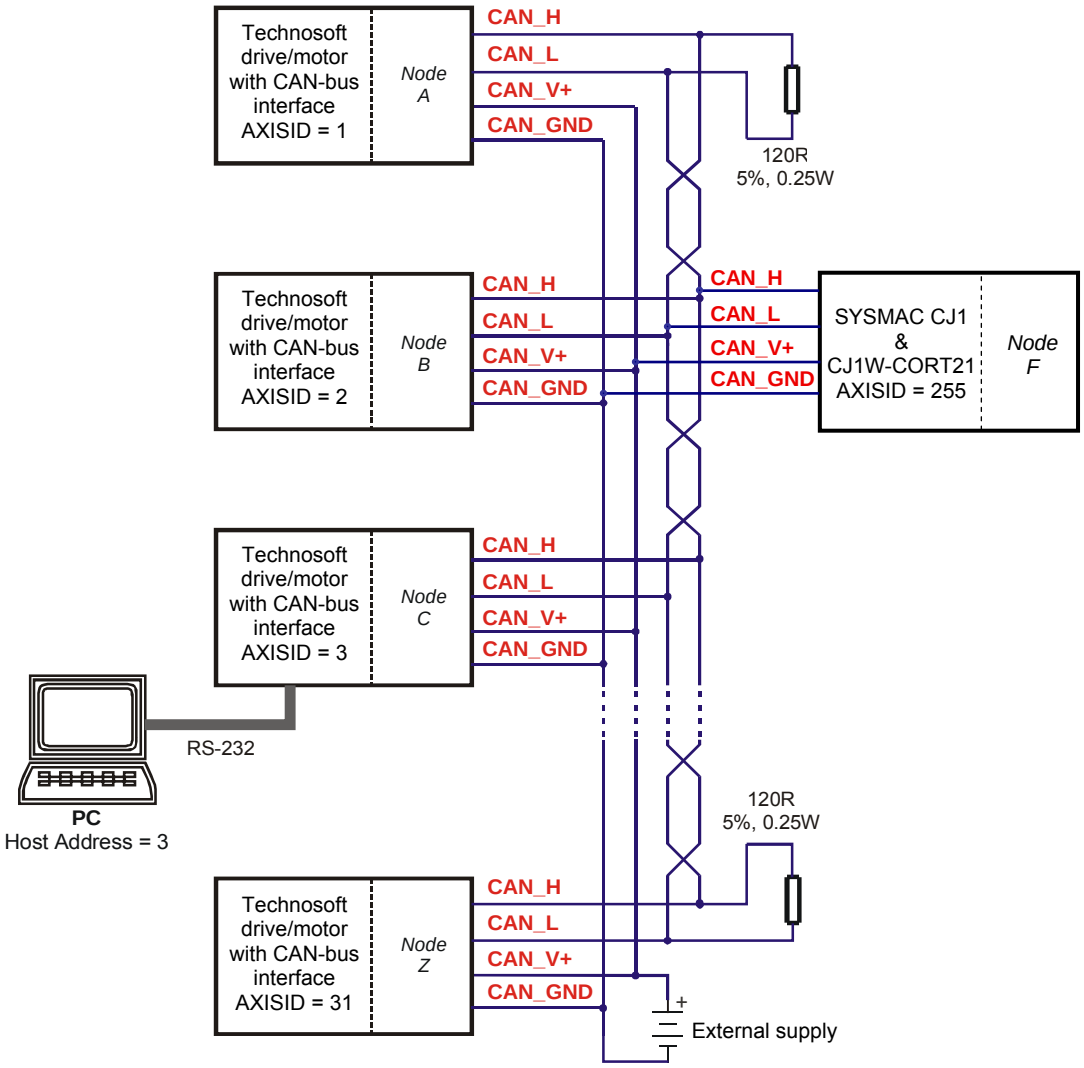


Figure 2.1 *Multiple-Axis CAN network*

Remarks:

- *The 120Ω terminating resistors between CAN-HI and CAN-LO are mandatory*
- *Some Technosoft products do not require an external supply for the CAN interface*

For the hardware installation of the Technosoft drives/motors see their user manual.

For the hardware installation of the OMRON SYSMAC CJ PLC and the CJ1W-CORT21 user defined CAN unit, see their user manual.

For drives/motors setup, you can connect your PC to any drive/motor using an RS232 serial link. Through this serial link you can access all the drives/motors from the network. Alternately, you can connect your PC directly on the CAN bus network if it is equipped with one of the CAN interfaces supported by EasySetUp.

2.2 Software installation

In order to perform successfully the following software installations, make sure that you have the "Administrator" rights.

2.2.1 Installing EasySetUp

On the TML_LIB_CJ1 installation CD you'll find the setup for EasyMotion Studio Demo version. This application includes a fully functional version of EasySetUp and a demo version of EasyMotion Studio. Start the setup and follow the installation instructions.

2.2.2 Installing TML_LIB_CJ1 library

Start the TML_LIB_CJ1 setup and follow the installation instructions.

2.3 Drive/motor setup

Before starting to send motion commands from your PLC, you need to do the drive/motor setup according with your application needs. For this operation you'll use **EasySetUp**.

EasySetUp is an integrated development environment for the setup of Technosoft drives and motors (with the drive integrated in the motor case). The output of **EasySetUp** is a set of setup data, which can be downloaded to the drive/motor EEPROM or saved on your PC for later use.

A setup contains all the information needed to configure and parameterize a Technosoft drive/motor. This information is preserved in the drive/motor EEPROM in the setup table. The setup table is copied at power-on into the RAM memory of the drive/motor and is used during runtime. The reciprocal is also possible i.e. to retrieve the complete setup data from a drive/motor EEPROM previously programmed

Steps to follow for commissioning a Technosoft drive/motor

Step 1. Start EasySetUp

From Windows Start menu execute: "Start | Programs | EasySetUp | EasySetUp" or "Start | Programs | EasyMotion Studio | EasySetUp" depending on which installation package you have used.

Step 2. Establish communication

EasySetUp starts with an empty window from where you can create a **New** setup, **Open** a previously created setup which was saved on your PC, or **Upload** the setup from the drive/motor.

Before selection one of the above options, you need to establish the communication with the drive/motor you want to commission. Use menu command **Communication | Setup** to check/change your PC communication settings. Press the **Help** button of the dialogue opened. Here you can find detailed information about how to setup your drive/motor and do the connections. Power on the drive/motor and then close the **Communication | Setup** dialogue with OK. If the communication is established, EasySetUp displays in the status bar (the bottom line) the text "**Online**" plus the axis ID of your drive/motor and its firmware version. Otherwise the text displayed is "**Offline**" and a communication error message tells you the error type. In this case, return to the Communication | Setup dialogue, press the Help button and check troubleshoots.

Remark: *When first started, EasySetUp tries to communicate with your drive/motor via RS-232 and COM1 (default communication settings). If your drive/motor is powered and connected to your PC port COM1 via an RS-232 cable, the communication can be automatically established.*

Step 3. Setup drive/motor

Press **New** button and select your drive/motor type. Depending on the product chosen, the selection may continue with the motor technology (for example: brushless motor, brushed motor) or the control mode (for example stepper – open-loop or stepper – closed-loop) and type of feedback device (for example: incremental encoder, SSI encoder)

This opens 2 setup dialogues: for **Motor Setup** and for **Drive setup** through which you can configure and parameterize a Technosoft drive/motor, plus several predefined control panels customized for the product selected.

In the **Motor setup** dialogue you can introduce the data of your motor and the associated sensors. Data introduction is accompanied by a series of tests having as goal to check the connections to the drive and/or to determine or validate a part of the motor and sensors parameters. In the **Drive setup** dialogue you can configure and parameterize the drive for your application. In each dialogue you will find a **Guideline Assistant**, which will guide you through the whole process of introducing and/or checking your data. Close the Drive setup dialogue with **OK** to keep all the changes regarding the motor and the drive setup.

Step 4. Download setup data to drive/motor

Press the **Download to Drive/Motor** button to download your setup data in the drive/motor EEPROM memory in the *setup table*. From now on, at each power-on, the setup data is copied into the drive/motor RAM memory that is used during runtime. It is also possible to **Save** the setup data on your PC and use it in other applications.

Step 5 Reset the drive/motor to activate the setup data

2.4 User defined CAN unit setup

Step 1. Unit No. The unit number uniquely identifies the CAN Unit when more than one CPU Bus Unit is mounted to the same PLC. The Unit No is set with the rotary switch labeled **Unit No**. The unit number setting determines the CIO and DM area words allocated to the Unit as software switches and the status area.

Step 2. CAN bus baud rate. Use the DIP switches on the front of the User Defined CAN Unit to set the CAN bus baud rate according with the value set for Technosoft drives/motors. Technosoft drives/motors support the following baud rates: 125, 250, 500, 800 and 1000 kb. The drive/motors CAN bus baud rate is set at power on using the following algorithm:

- With the value read from the EEPROM setup table
- If the setup table is invalid, with the last baud rate read from a valid setup table
- If there is no baud rate set by a valid setup table, with 500kb. For this baud rate the DIP switches positions must be 1-ON, 2-OFF, 3-ON and 4- OFF.

See "*CJ1W-CORT21 User Defined CAN Unit Operation Manual*" for a description of DIP switches positions for CAN Unit supported baud rates.

The User Defined CAN Unit is configured, through library's function blocks, to communicate using CAN protocol 2.0B and thus the messages sent/received will have 29-bit identifier.

2.5 Build a CX Programmer project with TML_LIB_CJ1

The TML_LIB_CJ1 library is provided as a collection of STL source files. For each library component you will find a source file with extension .cxf identifiable by the file name. The next steps detail how to include the TML_LIB_CJ1 components in a CX Programmer project.

1. **Create the project.** Launch **CX Programmer** and create a new project. For details read the CX Programmer online help.
2. **Configure the memory buffers required by TML_LIB_CJ1 and User Defined CAN Unit.** The TML_LIB requires a memory buffer where to store the axes information. The user specifies the memory type and start address of the buffer through a globally defined symbol named **AddressInfoAxis**. Each axis requires **24 words** and thus the length of the buffer is obtained by multiplying the number of axed with 24.

   Before a CAN message is sent from the CAN Unit, the CAN data is stored in a dedicated buffer. The message is sent at the end of each PLC cycle if the data buffer has an identifier associated and the corresponding trigger bit is set. This mode of operation requires 2 buffers, a send buffer where to store the CAN data to be sent and a buffer for the send trigger bits. For CAN messages received by the CAN Unit there are also required 2 buffers, a receive buffer to store the CAN data received and a buffer for receive triggers, that signals received messages.

   The start addresses of the buffer are set through globally defined symbols. **The addresses of the symbols must be in the data memory and must ensure that the buffers do not overlap**. For each axis you will have the following requirements:

   - **35 words** for send buffer. The send buffer length is $\text{No\_axes} \times 35$ No words
   - **7 bits** for send trigger buffer. The send trigger buffer length is $(\text{No\_axes} \times 7)/16$ rounded up to whole numbers
   - **10 words** for receive buffer. The receive buffer length is $\text{No\_axes} \times 2$ No words
   - **2 bits** for receive trigger buffer. The send trigger buffer length is $(\text{No\_axes} \times 2)/16$ rounded up to whole numbers
3. **Define global symbols.** In order to work properly the TML_LIB requires several symbols defined globally. Table 2.1 lists the symbols name, type, address and memory type.

   The CANModuleEnable, CANModuleEnabled and CANMessageReceived global symbols address depend on the CAN Unit allocated memory in the CIO area. For example if the UnitNo = 2 then the start address of the allocated CIO area is CIO1500+ (25 x UnitNo) = CIO1550. Then you will have for CANModuleEnable the address CIO1550.04, for

CANModuleEnabled the address CIO1553.02 and for CANModuleEnabled the address CIO1553.03.

***Table 1.1** Globally defined symbols*

Name	Type	Address	Memory type	Description
AddressInfoAxis	WORD	User defined	User defined	Specified the start address of the buffer with axes information
AddressSendBuffer	WORD	User defined	DM	Specified the start address of the buffer with CAN message data to be sent. **Must be in data memory**
AddressSendTrigger	WORD	User defined	DM	Specified the start address of the buffer with send trigger bits. **Must be in data memory**
AddressReceiveBuffer	WORD	User defined	DM	Specified the start address of the buffer with CAN message data received. **Must be in data memory**
AddressReceiveTrigger	WORD	User defined	DM	Specified the start address of the buffer with receive triggers. **Must be in data memory**
CANModuleEnable	BOOL	Function of unit number	CIO	Enable CAN communication. Represents the 4th bit from the first
CANModuleEnabled	BOOL	Function of unit number	CIO	Signals CAN communications enabled
CANMessageReceived	BOOL	Function of unit number	CIO	Signals a CAN message received and accepted by the CAN unit
CommEnableFlag	BOOL	A202.00	A	Signals the possibility to execute FINS commands
PLCAxisID	INT	User defined	User defined	Stores the PLC Axis ID

4. Import TML_LIB_CJ1 function blocks. **From the project tree select Function block** tree item, then click right mouse button and select the option **Insert Function | From File…** In the **Select Function Block Library File** dialogue select the folder where you unzipped the **TML_LIB_CJ1** and select the appropriate function.

***Remark:** Function TS_CommunicationInit followed by TS_AxisState are mandatory. The TS_CommunicationInit will configure the CAN Unit and TS_AxisState will configure the Technosoft drives/motors to send CAN messages automatically.*

5. **Create your application.** Build the application motion using the functions supplied with TML_LIB_CJ1.

3 TML_LIB_CJ1 description

3.1 Basic concepts

The TML_LIB_CJ1 functions and function blocks were built following the guidelines described in PLC standards for motion control.

Each Technosoft drive/motor is represented at PLC level using the term **Axis**. Its behavior is defined through up to 8 states. Figure presents the axis behavior implemented through TML_LIB_CJ1. At any moment the axis must be in only one state. The states are stored on the PLC, for each axis, in the TML_DB. Table 3.1 shows the axis states and their corresponding values.

Table 3.1 *Axis states*

State	State code	State byte value
Disable	0	1
Standstill	1	2
Discrete Motion	2	4
Continuous Motion	3	8
Synchronized Motion	4	16
Stopping	5	32
ErrorStop	6	64
Homing	7	128

The implemented function blocks are classified into 2 categories:

- Administrative (no driving motion) e.g. MC_Power, TS_WriteIntegerParameter, etc.
- Motion related e.g. MC_GearIn, MC_MoveVelocity, etc.

The axis state is changed by function blocks that send motion commands towards Technosoft drive/motor. Administrative functions have no influence on axis state

Remark: *Function TS_CommunicationInit followed by TS_AxisState must be called before any other library function calls since they initialize the CAN bus communication.*

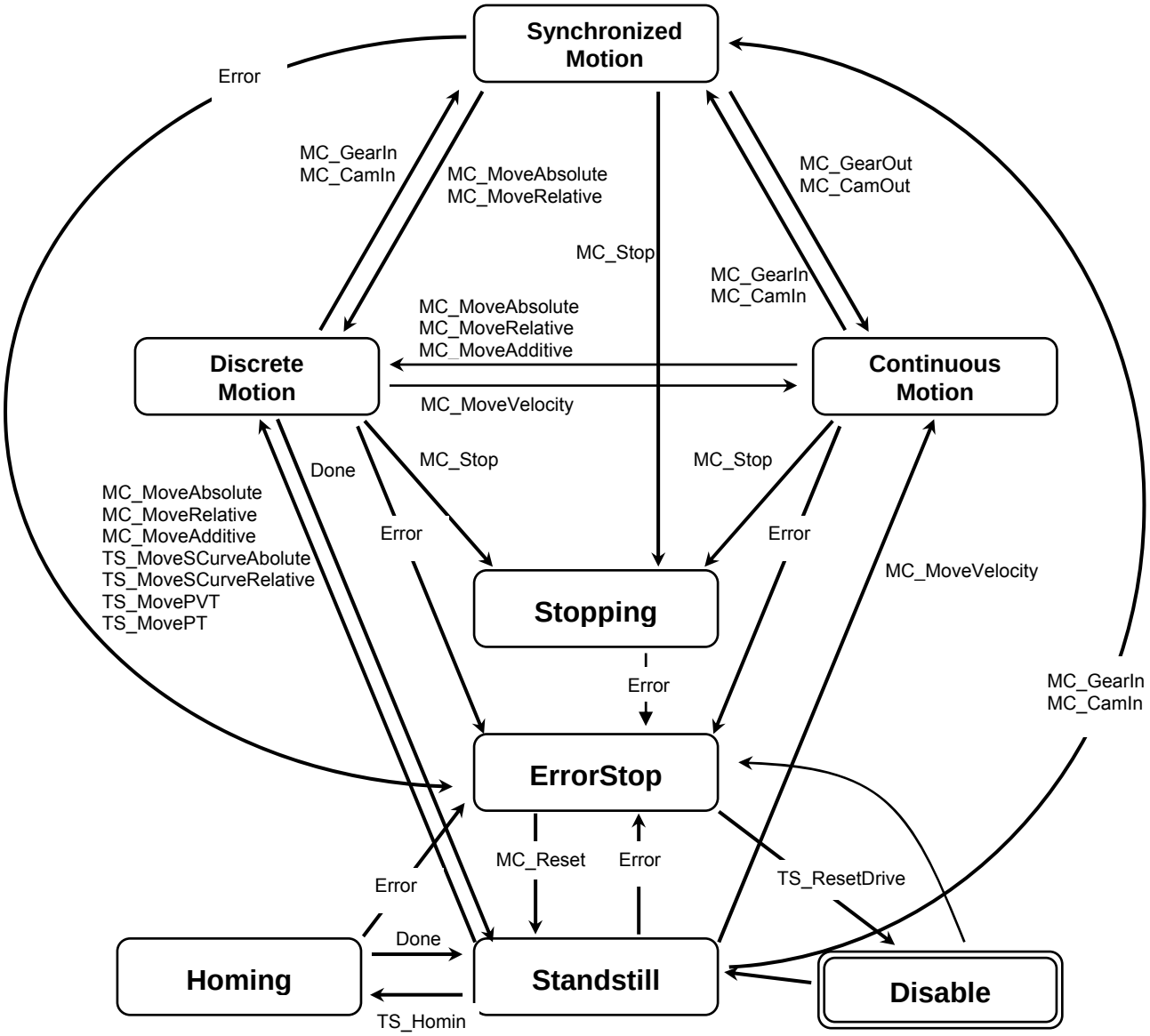


Figure 3.1. *Axis behaviour*

3.2 TML_LIB_CJ1 requirements

In order to use TML_LIB_CJ1 for OMRON SYSMAC CJ1 series you must have the following minimal hardware configuration:

- One or more Technosoft drives/motors, connected through a CAN-bus network
- An OMRON SYSMAC CJ1 PLC with function blocks support. See CX Programmer online help for a list with CPU types that support function blocks.
- An User defined CAN unit

The software required in order to implement an application on the previous configuration consists of:

- **EasySetUp** for setup of Technosoft drives/motors
- CX Programmer for hardware configuration and programming of the PLC

3.3 Internal units and scaling factors

Technosoft drives/motors work with parameters and variables represented in internal units (IU). The parameters and variables may represent various signals: position, speed, current, voltage, etc. Each type of signal has its own internal representation in IU and a specific scaling factor. In order to easily identify each type of IU, these have been named after the associated signals. For example the **position units** are the internal units for position, the **speed units** are the internal units for speed, etc.

The scaling factor of each internal unit shows the correspondence with the international standard units (SI). The scaling factors are dependent on the product, motor and sensor type. Put in other words, the scaling factors depend on the setup configuration.

In order to find the internal units and the scaling factors for a specific case, select the application in the project window and then execute menu command **Help | Setup Data Management | Internal Units and Scaling Factors**.

Important: *The* ***Help*** **| Setup Data Management |** ***Internal Units and Scaling Factors*** *command provides customized information, function of the application setup. If you change the drive, the motor technology or the feedback device, check again the scaling factors with this command. It may show you other relations!*

3.4 TML data

The TML uses the following data types:

- `int` 16-bit signed integer
- `uint` 16-bit unsigned integer
- `fixed` 32-bit fixed-point data with the 16MSB for the integer part and the 16LSB for the factionary part.
- `long` 32-bit signed integer
- `ulong` 32-bit unsigned integer

Since the motion defining parameters, i.e. position (distance), speed, acceleration and jerk have different data types in TML, for consistency, they are represented at the PLC level using only data type **real**, the conversion to TML data types is performed automatically by he function block.

3.5 Axis ID Identification

The data exchanged on the CAN bus is done using messages. Each message contains one TML instruction to be executed by the receiver of the message. Apart from the binary code of the TML instruction attached, any message includes information about its destination: an axis (drive/motor) or group of axes. This information is grouped in the **Axis/Group ID Code**. Each drive/motor has its own 8-bit Axis ID and Group ID.

Remarks:

1. *The Axis ID of a drive/motor must be **unique** and is set during the drive/motor setup phase with **EasySetUp**. The TML_LIB_CJ1 doesn't contain a function or function block for changing the value of Axis ID.*
2. *A drive/motor belongs, by default, to the group ID = 1. You can set/change the group ID of a drive/motor with function block TS_SetGroupID.*
3. *The Axis ID and Group ID of a drive/motor are stored in TML variable **AAR**(uint@0x030C). Use TS_ReadIntegerParameter to read the value of the Axis ID and Group ID.*

The Group ID represents a way to identify a group of axes, for a multicast transmission. This feature allows the PLC to send a command simultaneously to several axes, for example to start or stop the axes motion in the same time. When a function block sends a command to a group, all the axes members of this group will receive the command. For example, if the axis is member of group 1 and group 3, it will receive all the messages that in the group ID include group 1 and group 3.

Each axis can be programmed to be member of one or several of the 8 possible groups.

Table 3.2*Definition of the groups*

Group No.	Group ID value
1	1 (0000 0001b)
2	2 (0000 0010b)
3	4 (0000 0100b)
4	8 (0000 1000b)
5	16 (0001 0000b)
6	32 (0010 0000b)
7	64 (0100 0000b)
8	128 (1000 0000b)

By default all function blocks send the commands to one axis. Several function blocks can also send commands to a group of axes. The destination of the command (single axis or group of axes) is set through function block's input **ControlWord**. If ControlWord.15 is set when the function block is executed the commands are sent to the axes having the Group ID equal with the value read from input **AxisID**.

Remark: *When a function block is configured for group commands it will use the buffer allocated to the first axis defined.*

3.6 Functions descriptions

The section presents function blocks implemented in the **TML_LIB_CJ1** library. The following convention in naming the functions applies to library's elements: function name starting with "MC" refers to a function that complies with PLC standard, for example MC_MoveAbsolute. The functions name starting with "TS" refer to a function specific to Technosoft drives/motors, for example TS_MovePVT.

The implemented function blocks are classified as:

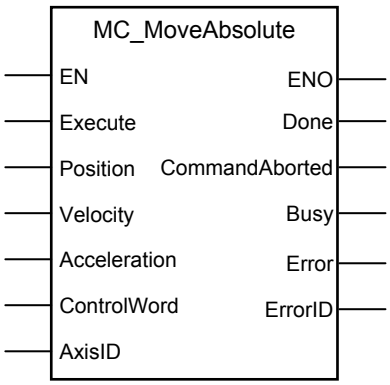
- Motion related
  - MC_MoveAbsolute
  - MC_MoveRelative
  - MC_MoveAdditive
  - TS_MoveSCurveAbsolute
  - TS_MoveSCurveRelative
  - MC_MoveVelocity
  - TS_ExternalAnalogue
  - TS_ExternalDigital
  - TS_ExternalOnline
  - TS_SetPVT
  - TS_PVTPoint
  - TS_SetPT
  - TS_PTPoint
  - MC_GearIn
  - MC_GearOut
  - MC_CamIn
  - MC_CamOut
  - MC_Stop
  - TS_Homing
- Administrative (no driving motion).
  - MC_ReadActualPosition
  - MC_ReadStatus
  - MC_Power
  - MC_CamTableSelect
  - TS_SetMaster
  - TS_WriteLongParameter
  - TS_WriteFixedParameter
  - TS_WriteIntegerParameter
  - TS_ReadLongParameter
  - TS_ReadFixedParamter
  - TS_ReadIntegerParameter
  - TS_SetGroupID
  - TS_ResetFault
  - TS_ResetDrive
  - TS_AxisStatus
  - TS_SendCommand
  - TS_SaveParameters

For each function block you will find the following information:

- The symbol in LAD representation
- Parameters description with their name and associated data type
- A functional description
- Remarks regarding drive/motor setup
- An example that illustrates how to use the function or function block

3.6.1 MC_MoveAbsolute

Symbol:



Parameters description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Send motion commands at rising edge
	Position	REAL	Position to reach expressed in TML position internal units
	Velocity	REAL	The slew speed expressed in TML speed internal units
	Acceleration	REAL	Acceleration increment expressed in TML acceleration internal units.
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Commanded position reached
	CommandAborted	BOOL	The function block is aborted by another motion function block
	Busy	BOOL	Signal the function block is waiting for motion complete
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block programs an absolute positioning with trapezoidal speed profile. You specify the position to reach plus the velocity (maximum travel) and the acceleration/deceleration rate. The velocity and acceleration must be positive. Negative values are taken in modulus.

On detecting a rising edge at the **Execute** input, the function block starts sending motion commands and sets the **Busy** output. The **Busy** output remains set until the drive/motor signals target reached, moment when the function block sets the **Done** output and resets **Busy**.

The **CommandAborted** output is set if another motion function block sends motion commands to the same drive/motor. If a drive's error register information is received during the execution of the function block, the output **Error** is set and its value is passed to **ErrorID**.

All outputs remain set until **Execute** input is reset, but at least for one block call.

During motion execution the axis is transferred into **DiscreteMotion** state, when the target is reached (**Done** output is set) the axis is transferred to **Standstill** state.

Once set, the motion parameters are memorized on the drive/motor. If you intend to use values previously defined (by a different motion function block or different instance of the same function block) for the acceleration rate, the velocity or the position to reach, you don't need to send their values again in the following trapezoidal profiles. Through **ControlWord** input you can select the motion parameters sent by the function block to the drive/motor.

If ControlWord.15 is set then the value read from **AxisID** input is interpreted as a group ID. The function block will send the motion commands to drives/motors members of the group. In this case the done output is set after the motion commands are sent.

Table 3.3 *ControlWord bits description*

Bit	Value	Description
0	0	Send position value
	1	Don't send the position value
1	0	Send the speed value
	1	Don't send the speed value
2	0	Send the acceleration value
	1	Don't send the acceleration value
3	0	Target Update Mode 1 (TUM1). Generates new trajectory starting from the actual values of position and speed reference
	1	Target Update Mode 0 (TUM0). Generates new trajectory starting from the actual values of load/motor position and speed
4-14	0	Reserved for new features
15	0	The motion commands are sent to a one drive/motor
	1	The motion commands are sent to a group of drives/motors.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

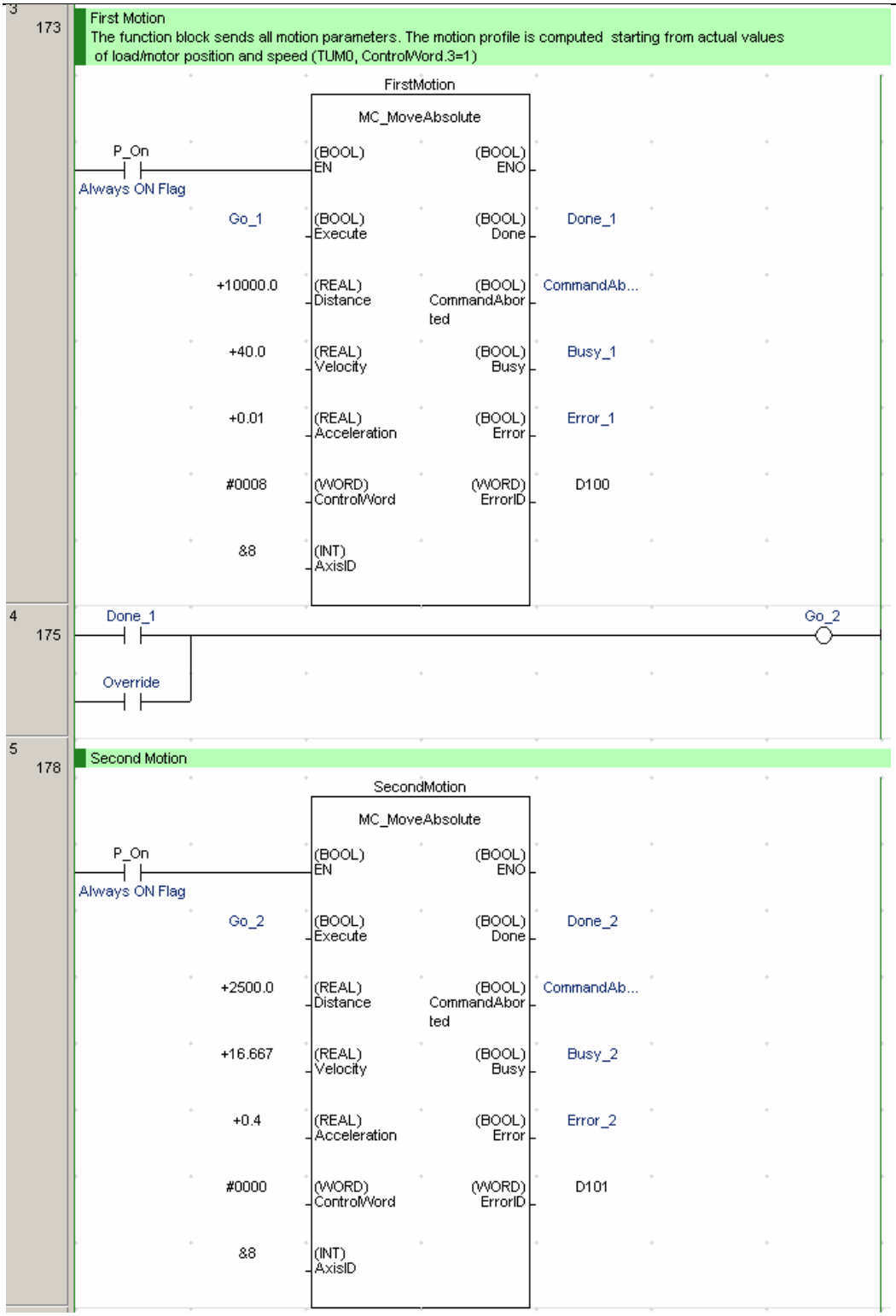
Remarks:

1. *The function block requires drive/motor position loop to be closed. During the drive/motor setup select **Position** at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between discrete motion (position control) and continuous motion (speed control) you must also close the speed loop and perform the tuning of the speed controller*.

Example:

The example shows how to concatenate two absolute positioning commands for the same axis. The first motion is triggered when "Go_1" becomes TRUE. The second motion can be triggered by "Done_1", when the first motion is completed, or by "Override" before the first motion

completes. If the second motion is triggered with “Override” it will abort the first motion. In Figure 3.1 is displayed the signals state for the two cases.


3
173
First Motion
The function block sends all motion parameters. The motion profile is computed starting from actual values of load/motor position and speed (TUM0, ControlWord.3=1)
FirstMotion
MC_MoveAbsolute
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_1
(BOOL) Execute
(BOOL) Done
Done_1
+10000.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+40.0
(REAL) Velocity
(BOOL) Busy
Busy_1
+0.01
(REAL) Acceleration
(BOOL) Error
Error_1
#0008
(WORD) ControlWord
(WORD) ErrorID
D100
&8
(INT) AxisID
4
175
Done_1
Override
Go_2
5
178
Second Motion
SecondMotion
MC_MoveAbsolute
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_2
(BOOL) Execute
(BOOL) Done
Done_2
+2500.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+16.667
(REAL) Velocity
(BOOL) Busy
Busy_2
+0.4
(REAL) Acceleration
(BOOL) Error
Error_2
#0000
(WORD) ControlWord
(WORD) ErrorID
D101
&8
(INT) AxisID

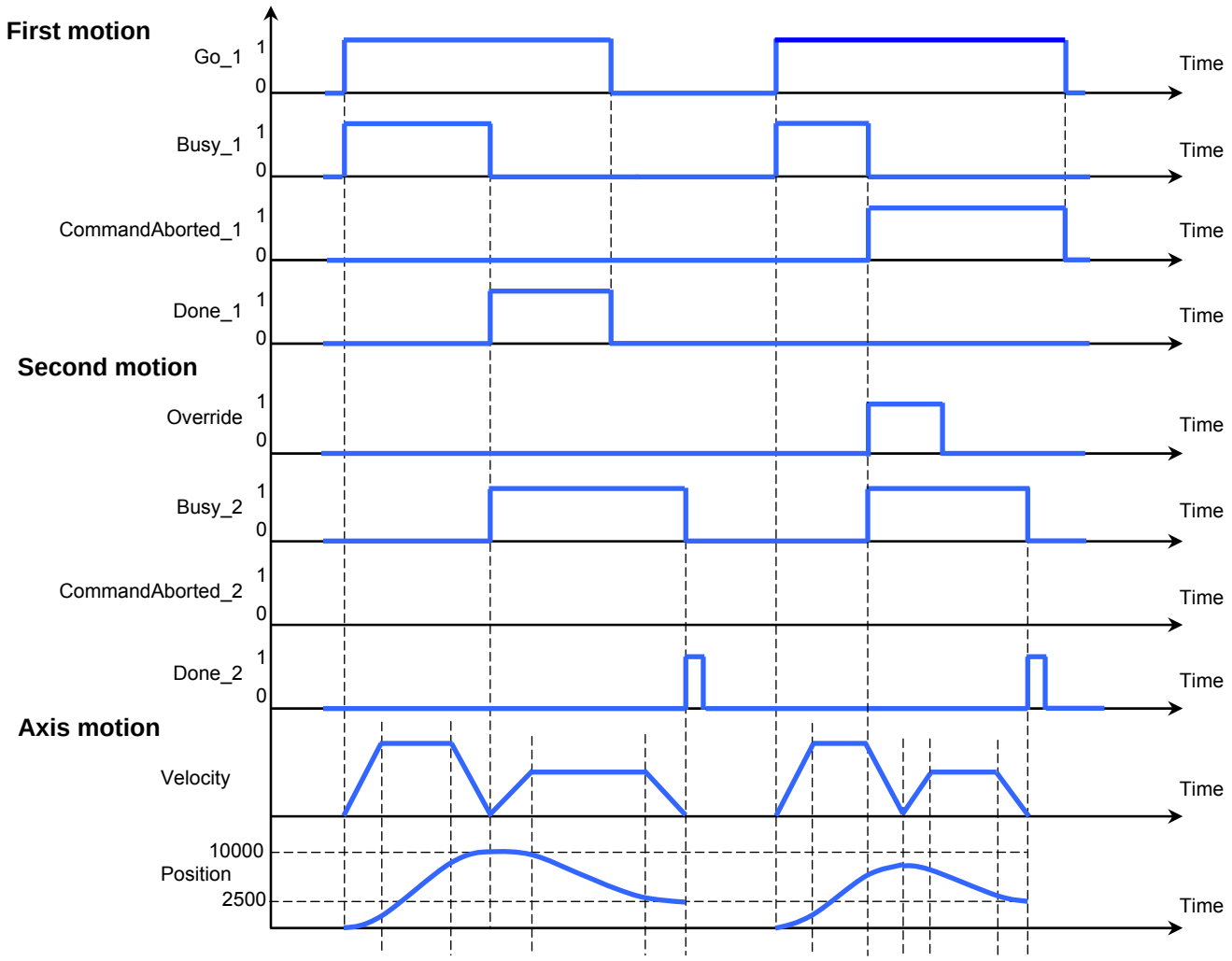
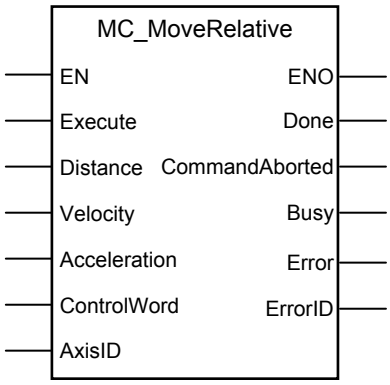


Figure 3.1. *Time diagram for 2 consecutive absolute motions*

3.6.2 MC_MoveRelative

Symbol:



Parameters description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function block execution
	Execute	BOOL	Send motion commands at rising edge
	Distance	REAL	Position increment expressed in TML position internal units
	Velocity	REAL	The slew speed expressed in TML speed internal units
	Acceleration	REAL	Acceleration rate expressed in TML acceleration internal units.
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Signal the motion complete
	CommandAborted	BOOL	The function block is aborted by another motion function block
	Busy	BOOL	Signal the function block is waiting for motion complete
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block programs a relative positioning with trapezoidal speed profile. You specify the position increment plus the velocity (maximum travel speed) and the acceleration/deceleration rate. The values of velocity and acceleration must be positive. Negative values are taken in modulus.

On detecting a rising edge at the **Execute** input, the function block starts sending motion commands and sets the **Busy** output. The **Busy** output remains set until the drive/motor signals target reached, moment when the function block sets the **Done** output and resets **Busy**.

The **CommandAborted** output is set if another motion function block sends motion commands to the same drive/motor. If a drive's error register information is received during the execution of the function block, the output **Error** is set and its value is passed to **ErrorID**.

All outputs remain set until **Execute** input is reset, but at least for one block call.

During motion execution the axis is transferred to **DiscreteMotion** state, when the target is reached (**Done** output is set) the axis is transferred to **Standstill** state.

Once set, the motion parameters are memorized on the drive/motor. If you intend to use values previously defined (by a different motion function block or different instance of the same function block) for the acceleration rate, the velocity or the position to reach, you don't need to send their values again in the following trapezoidal profiles. Through **ControlWord** input you can select the motion parameters sent by the function block to the drive/motor.

If ControlWord.15 is set then the value read from **AxisID** input is interpreted as a group ID. The function block will send the motion commands to drives/motors members of the group. In this case the done output is set after the motion commands are sent.

Table 3.4 *ControlWord bits description*

Bit	Value	Description
0	0	Send position value
	1	Don't send the position value
1	0	Send the speed value
	1	Don't send the speed value
2	0	Send the acceleration value
	1	Don't send the acceleration value
3	0	Target Update Mode 1 (TUM1). Generates new trajectory starting from the actual values of position and speed reference
	1	Target Update Mode 0 (TUM0). Generates new trajectory starting from the actual values of load/motor position and speed
4-14	0	Reserved for new features
15	0	The motion commands are sent to a single drive/motor
	1	The motion commands are sent to a group of drives/motors.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Remarks:

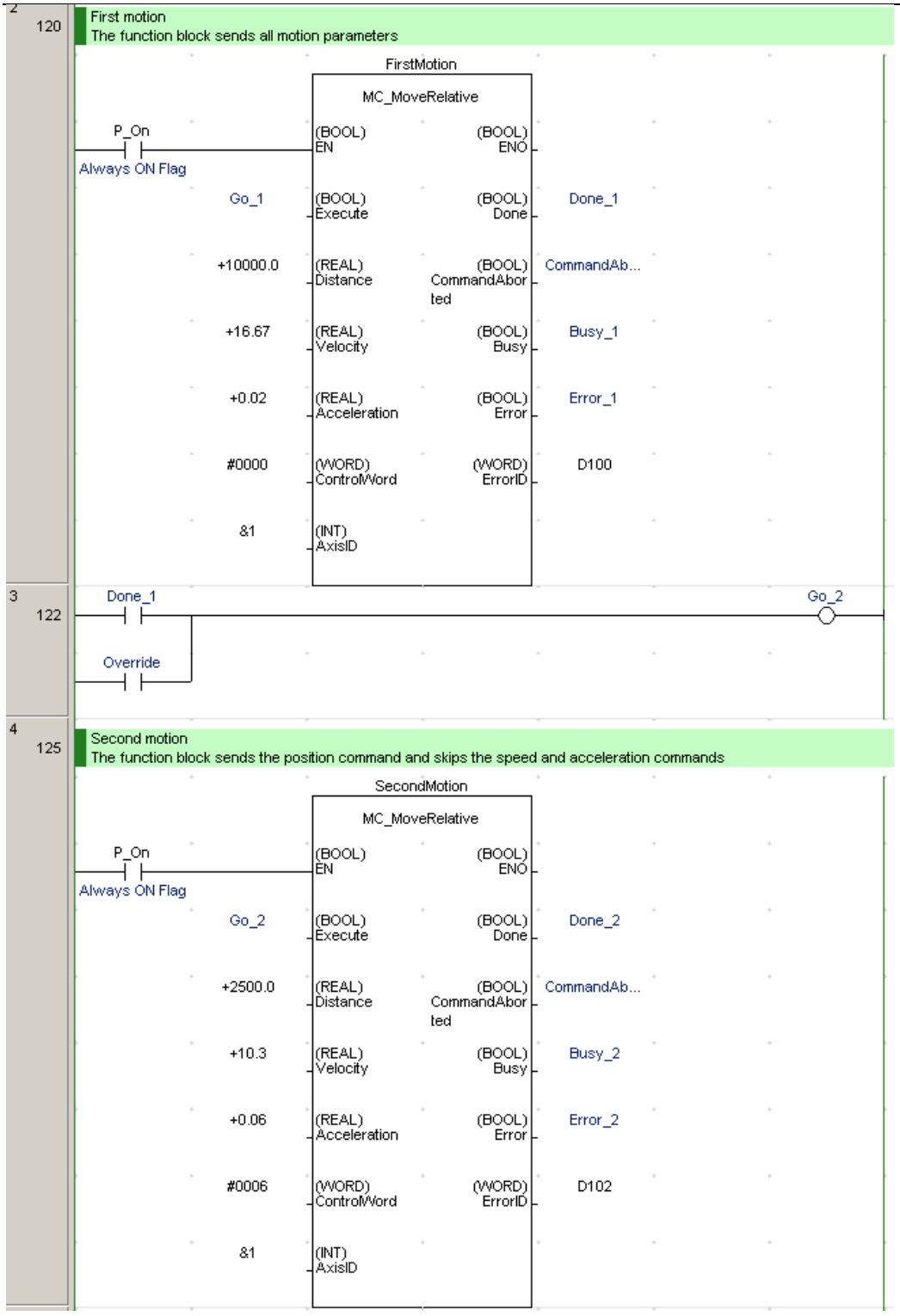
1. *The function block requires drive/motor position loop to be closed. During the drive/motor setup select **Position** at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between discrete motion (position control) and continuous motion (speed control) you must also close the speed loop and perform the tuning of the speed controller*.

Example:

The example shows how to concatenate two relative positioning commands for the same axis. The first motion is triggered when "Go_1" becomes TRUE. The second motion can be triggered by "Done_1", when the first motion is completed, or by "Override" before the first motion

completes. If the second motion is triggered with “Override” it will abort the first motion. Figure 3.2displays the signals state for the two cases.

The second motion sends only the position increment command.


2
120
First motion
The function block sends all motion parameters
FirstMotion
MC_MoveRelative
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_1
(BOOL) Execute
(BOOL) Done
Done_1
+10000.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+16.67
(REAL) Velocity
(BOOL) Busy
Busy_1
+0.02
(REAL) Acceleration
(BOOL) Error
Error_1
#0000
(WORD) ControlWord
(WORD) ErrorID
D100
&1
(INT) AxisID
3
122
Done_1
Override
Go_2
4
125
Second motion
The function block sends the position command and skips the speed and acceleration commands
SecondMotion
MC_MoveRelative
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_2
(BOOL) Execute
(BOOL) Done
Done_2
+2500.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+10.3
(REAL) Velocity
(BOOL) Busy
Busy_2
+0.06
(REAL) Acceleration
(BOOL) Error
Error_2
#0006
(WORD) ControlWord
(WORD) ErrorID
D102
&1
(INT) AxisID

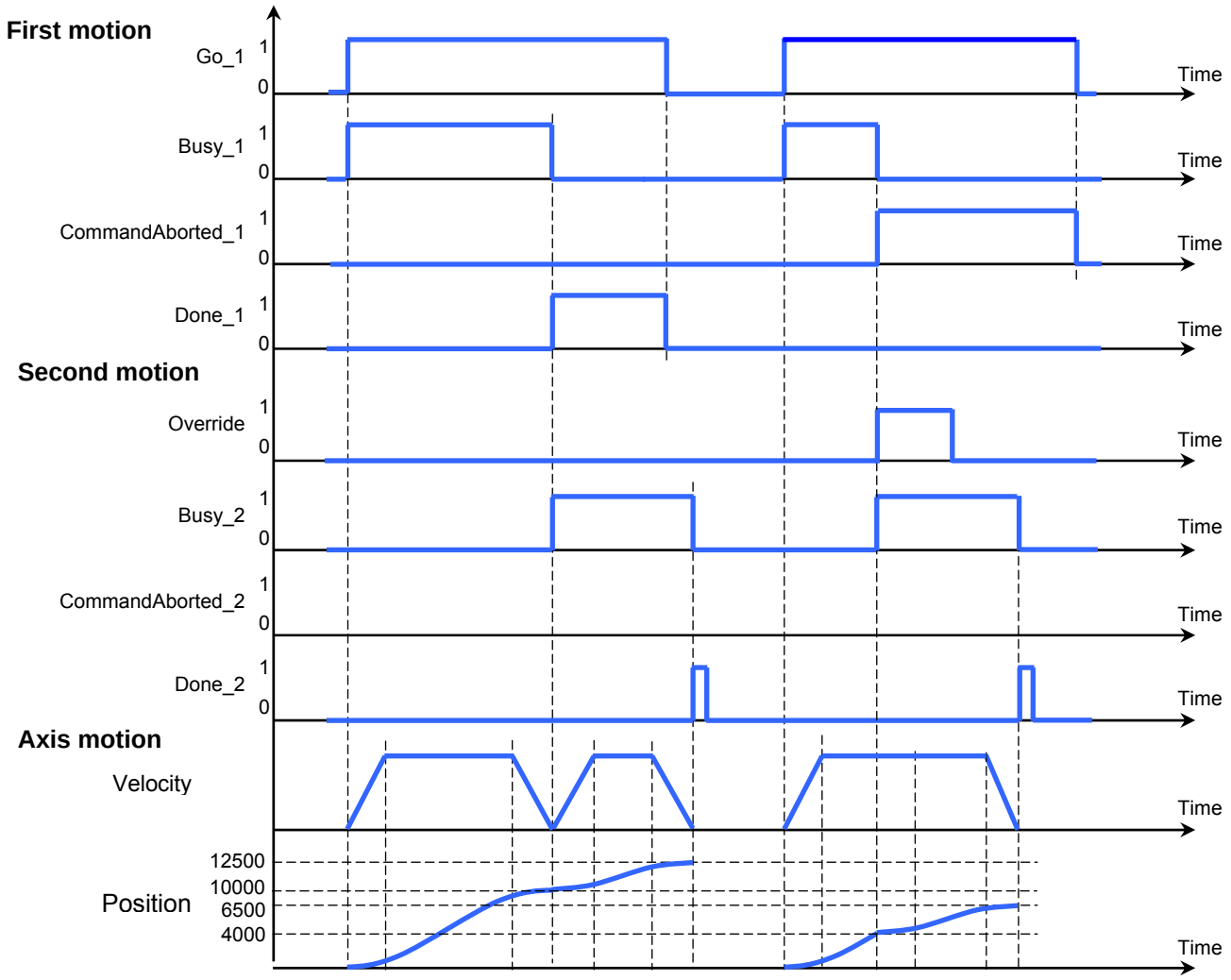
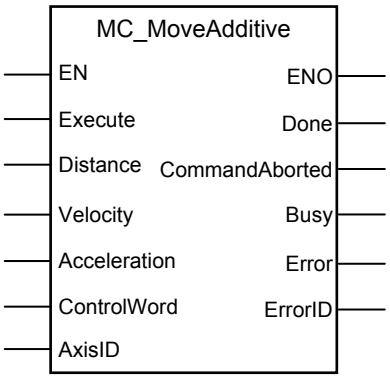


Figure 3.2*Time diagram for 2 consecutive relative motions*

3.6.3 MC_MoveAdditive

Symbol:



Parameters description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Send motion commands at rising edge
	Distance	REAL	Position increment expressed in TML position internal units
	Velocity	REAL	The slew speed expressed in TML speed internal units
	Acceleration	REAL	Acceleration rate expressed in TML acceleration internal units.
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Signal the motion complete
	CommandAborted	BOOL	The function block is aborted by another motion function block
	Busy	BOOL	Signal the function block is waiting for motion complete
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block programs a relative positioning with trapezoidal speed profile. The position to reach is computed by adding the position increment to the previous position to reach, independently of the moment when the command was issued. You specify the position increment plus the velocity (maximum travel speed) and the acceleration/deceleration rate. The values of velocity and acceleration must be positive. Negative values are taken in modulus on the drive/motor.

Once set, the motion parameters are memorized on the drive/motor. If you intend to use values previously defined (by a different motion function block or different instance of the same function block) for the acceleration rate, the velocity or the position to reach, you don't need to send their values again in the following trapezoidal profiles. Through **ControlWord** input you can select the motion parameters sent by the function block to the drive/motor.

If ControlWord.15 is set then the value read from **AxisID** input is interpreted as a group ID. The function block will send the motion commands to drives/motors members of the group. In this case the done output is set after the motion commands are sent.

Table 3.5 *ControlWord bits description*

Bit	Value	Description
0	0	Send position value
	1	Don't send the position value
1	0	Send the speed value
	1	Don't send the speed value
2	0	Send the acceleration value
	1	Don't send the acceleration value
3	0	Target Update Mode 1 (TUM1). Generates new trajectory starting from the actual values of position and speed reference
	1	Target Update Mode 0 (TUM0). Generates new trajectory starting from the actual values of load/motor position and speed
4-14	0	Reserved for new features
15	0	The motion commands are sent to a single drive/motor
	1	The motion commands are sent to a group of drives/motors.

Remarks:

1. *The function block requires drive/motor position loop to be closed. During the drive/motor setup select **Position** at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between discrete motion (position control) and continuous motion (speed control) you must also close the speed loop and perform the tuning of the speed controller*.

On detecting a rising edge at the **Execute** input, the function block starts sending motion commands and sets the **Busy** output. The **Busy** output remains set until the drive/motor signals target reached, moment when the function block sets the **Done** output and resets **Busy**.

The **CommandAborted** output is set if another motion function block sends motion commands to the same drive/motor. If a drive's error register information is received during the execution of the function block, the output **Error** is set and its value is passed to **ErrorID**.

All outputs remain set until **Execute** input is reset, but at least for one block call.

During motion execution the axis is transferred to **DiscreteMotion** state, when the target is reached (**Done** output is set) the axis is transferred to **Standstill** state.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:

The example implements two consecutive motions for the drive/motor with axis ID = 1. The first motion is an absolute positioning at 6000 IU implemented with function block FB101 MC_MoveAbsolute. The second motion is a relative positioning additive to the first motion implemented with FB MC_MoveAdditive.

The first motion is triggered when “Go_1” becomes TRUE. The second motion can be triggered by “Done_1”, when the first motion is completed, or by “Override” before the first motion completes. If the second motion is triggered with “Override” it will abort the first motion. In Figure 3.3 is displayed the signals state for the two cases.

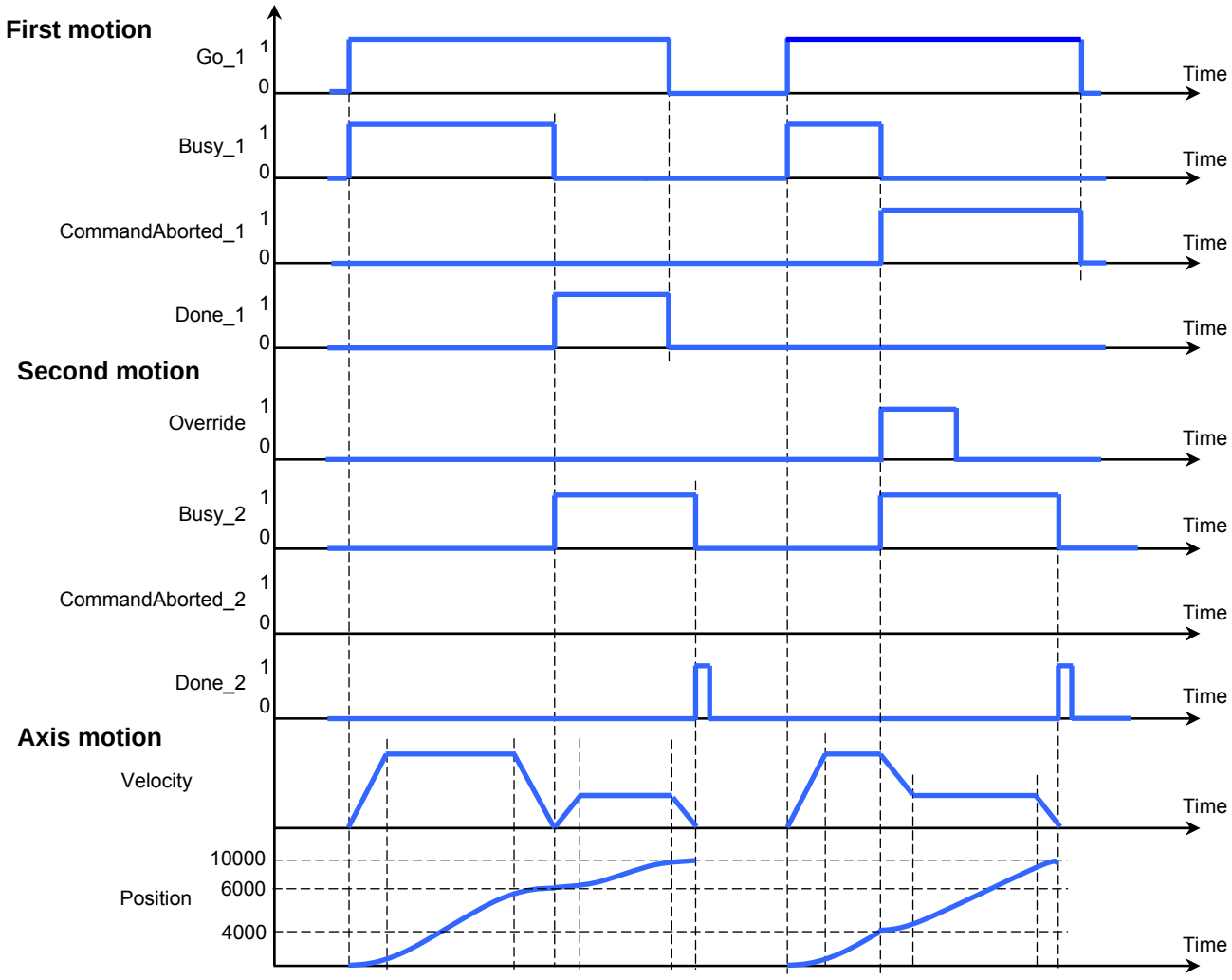
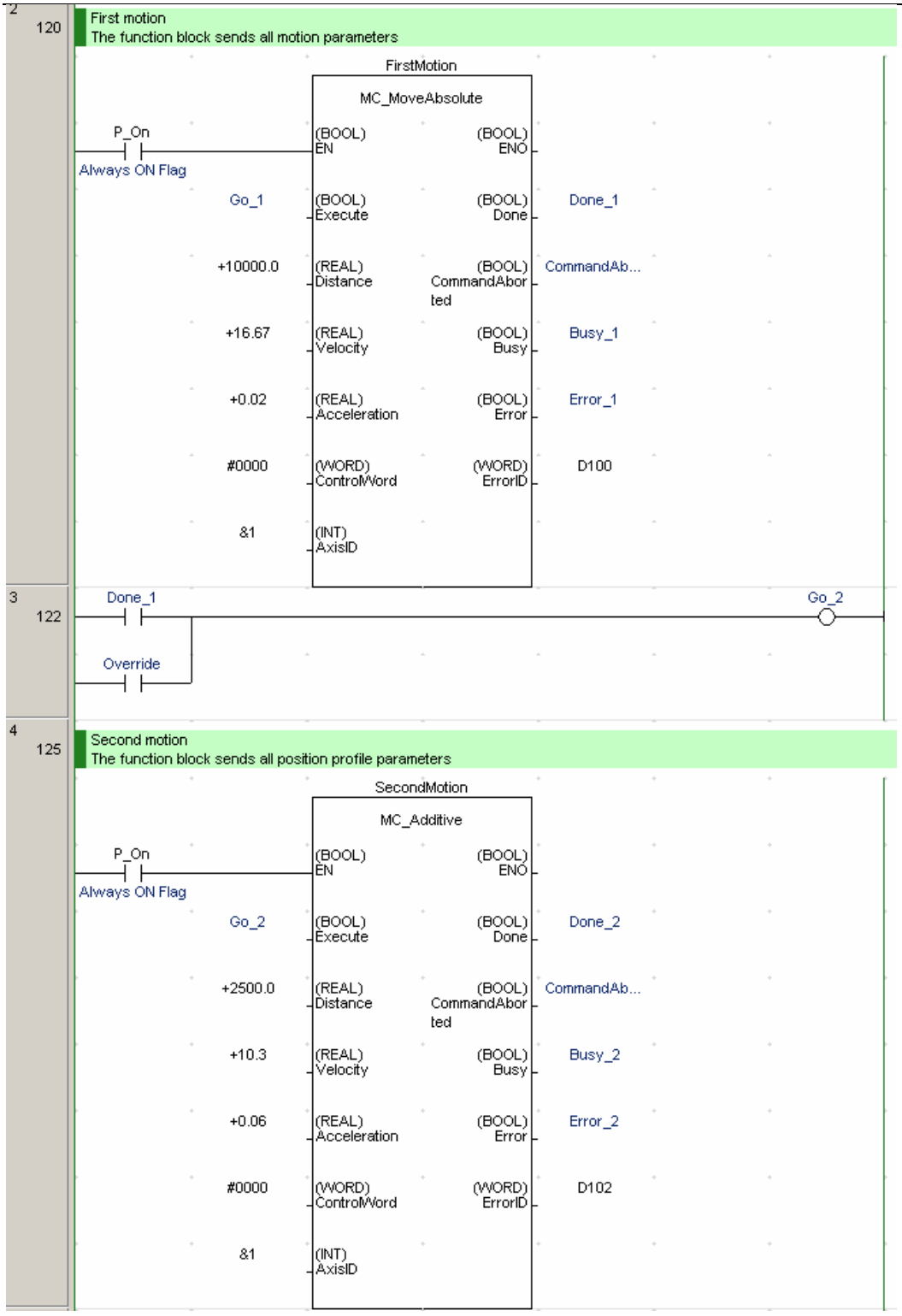
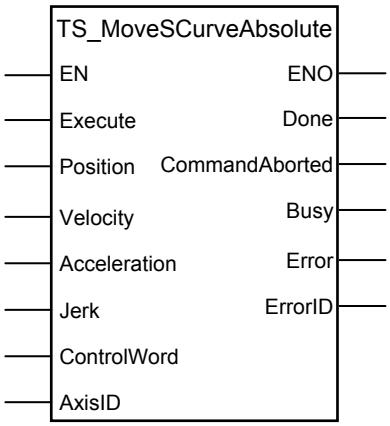


Figure 3.3. *Time diagram for an absolute motion concatenated with an additive motion*


2
120
First motion
The function block sends all motion parameters
FirstMotion
MC_MoveAbsolute
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_1
(BOOL) Execute
(BOOL) Done
Done_1
+10000.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+16.67
(REAL) Velocity
(BOOL) Busy
Busy_1
+0.02
(REAL) Acceleration
(BOOL) Error
Error_1
#0000
(WORD) ControlWord
(WORD) ErrorID
D100
&1
(INT) AxisID
3
122
Done_1
Override
Go_2
4
125
Second motion
The function block sends all position profile parameters
SecondMotion
MC_Additive
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_2
(BOOL) Execute
(BOOL) Done
Done_2
+2500.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+10.3
(REAL) Velocity
(BOOL) Busy
Busy_2
+0.06
(REAL) Acceleration
(BOOL) Error
Error_2
#0000
(WORD) ControlWord
(WORD) ErrorID
D102
&1
(INT) AxisID

3.6.4 TS_MoveSCurveAbsolute

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function block execution
	Execute	BOOL	Send motion commands at rising edge
	Position	REAL	Position to reach expressed in TML position internal units
	Velocity	REAL	The slew speed expressed in TML speed internal units
	Acceleration	REAL	Acceleration rate expressed in TML acceleration internal units.
	Jerk	REAL	Acceleration maximum jerk expressed in TML jerk internal units
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Signal the motion complete
	CommandAborted	BOOL	The function block is aborted by another motion function block
	Busy	BOOL	Signal the function block is waiting for motion complete
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block programs an absolute positioning with an S-curve shape of the speed. This shape is due to the jerk limitation, leading to a trapezoidal or triangular profile for the acceleration and an S-curve profile for the speed. You specify the position to reach plus the velocity (maximum travel speed), the maximum acceleration/deceleration rate and the jerk rate.

The function block can be executed only from **Standstill** state. During motion the parameters should not be changed. Therefore when executing successive motions with TS_MoveSCurveAbsolute function blocks, you should wait for the previous motion to end before setting the new motion parameters and starting next motion.

When the motion is stopped with function block MC_Stop, the deceleration phase can be done in 2 ways:

- Smooth, using an S-curve speed profile, when ControlWord.3 = 0, or
- Fast using a trapezoidal speed profile, when ControlWord.3 = 1

On detecting a rising edge at the **Execute** input, the function block starts sending motion commands and sets the **Busy** output. The **Busy** output remains set until the drive/motor signals target reached, moment when the function block sets the **Done** output and resets **Busy**.

The **CommandAborted** output is set if another motion function block sends motion commands to the same drive/motor. If a drive's error register information is received during the execution of the function block, the output **Error** is set and its value is passed to **ErrorID**.

All outputs remain set until **Execute** input is reset, but at least for one block call.

During motion execution the axis is transferred to **DiscreteMotion** state, when the target position is reached (**Done** output is set) the axis is transferred to **Standstill** state.

Once set, the motion parameters are memorized on the drive/motor. If you intend to use values previously defined (by a different motion function block or different instance of the same function block) for the acceleration rate, the velocity or the position to reach, you don't need to send their values again in the following trapezoidal profiles. Through **ControlWord** input you can select the motion parameters sent by the function block to the drive/motor.

If ControlWord.15 is set then the axis ID read from **AxisID** input is interpreted as a group ID. The function block will send the motion commands to drives/motors members of the group. In this case the done output is set after the motion commands are sent.

Table 3.6 *ControlWord bits description*

Bit	Value	Description
0	0	Send position value
	1	Don't send the position value
1	0	Send the speed value
	1	Don't send the speed value
2	0	Send the acceleration value
	1	Don't send the acceleration value
3	0	Smooth stop, using an S-curve speed profile, when function block MC_Stop is called
	1	Fast stop, using a trapezoidal speed profile, when function block MC_Stop is called
4-14	0	Reserved for new features
15	0	The motion commands are sent to a single drive/motor
	1	The motion commands are sent to a group of drives/motors.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Remarks:

1. *The function block requires drive/motor position loop to be closed. During the drive/motor setup select **Position** at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between discrete motion (position control) and continuous motion (speed control) you must also close the speed loop and perform the tuning of the speed controller*.
3. *For stopping with trapezoidal profile, the deceleration value is read from TML parameter CDEC (fixed@0x0859). Use TS_WriteFixedParameter in order to change its value. The default value is 0.5 IU.*

Example:

The example implements two consecutive motions for the drive/motor with axis ID =1. The first motion is a relative positioning at 10000 IU implemented with function block MC_MoveRelative. The second motion is an absolute positioning at 6000 IU implemented with FB TS_MoveSCurveAbsolute.

The first motion is triggered when “Go_1” becomes TRUE. The second motion can be triggered only when “Done_1” is set, when the first motion is completed and “Override” is set too. In Figure 3.4 is displayed the signals state for the two cases.

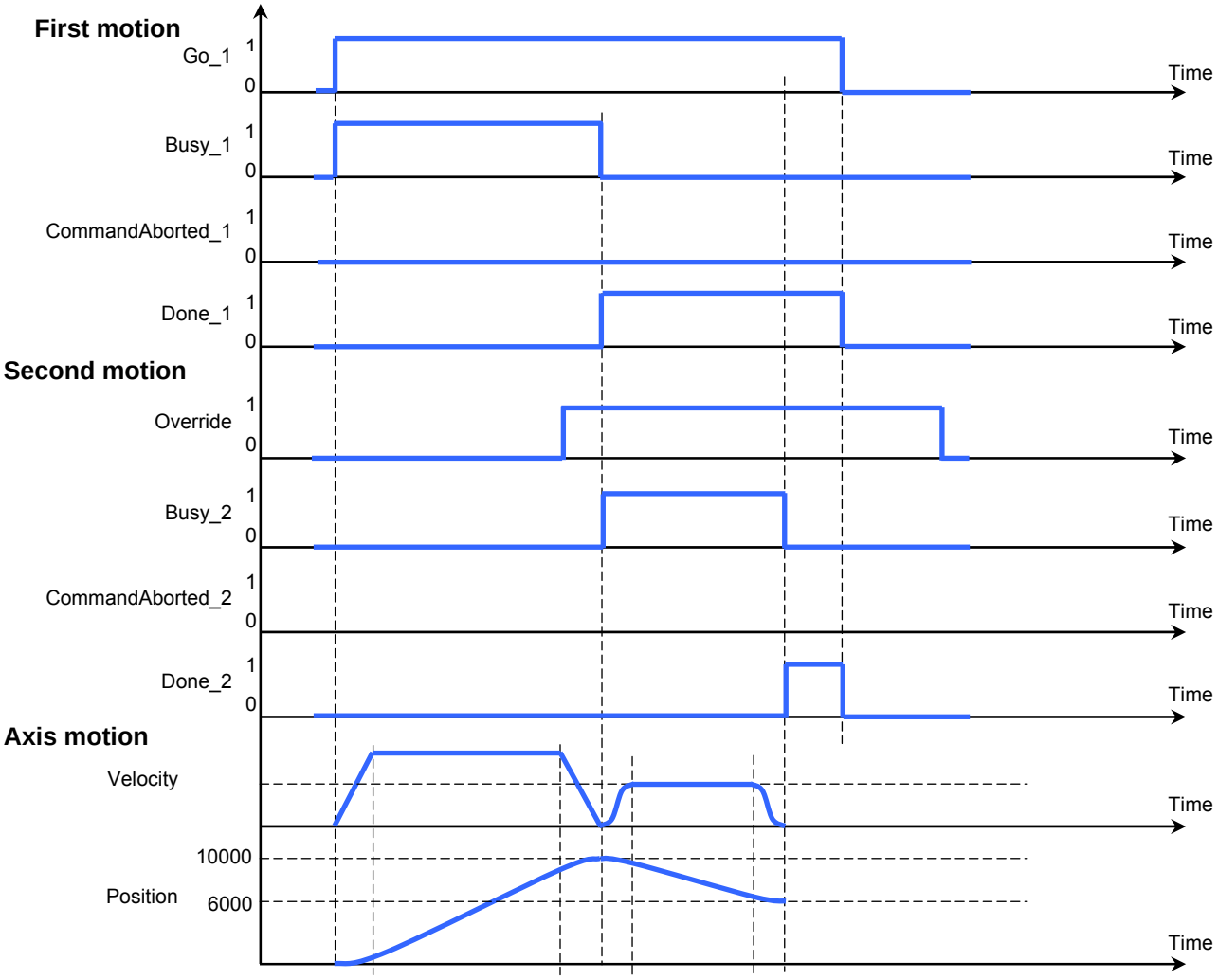
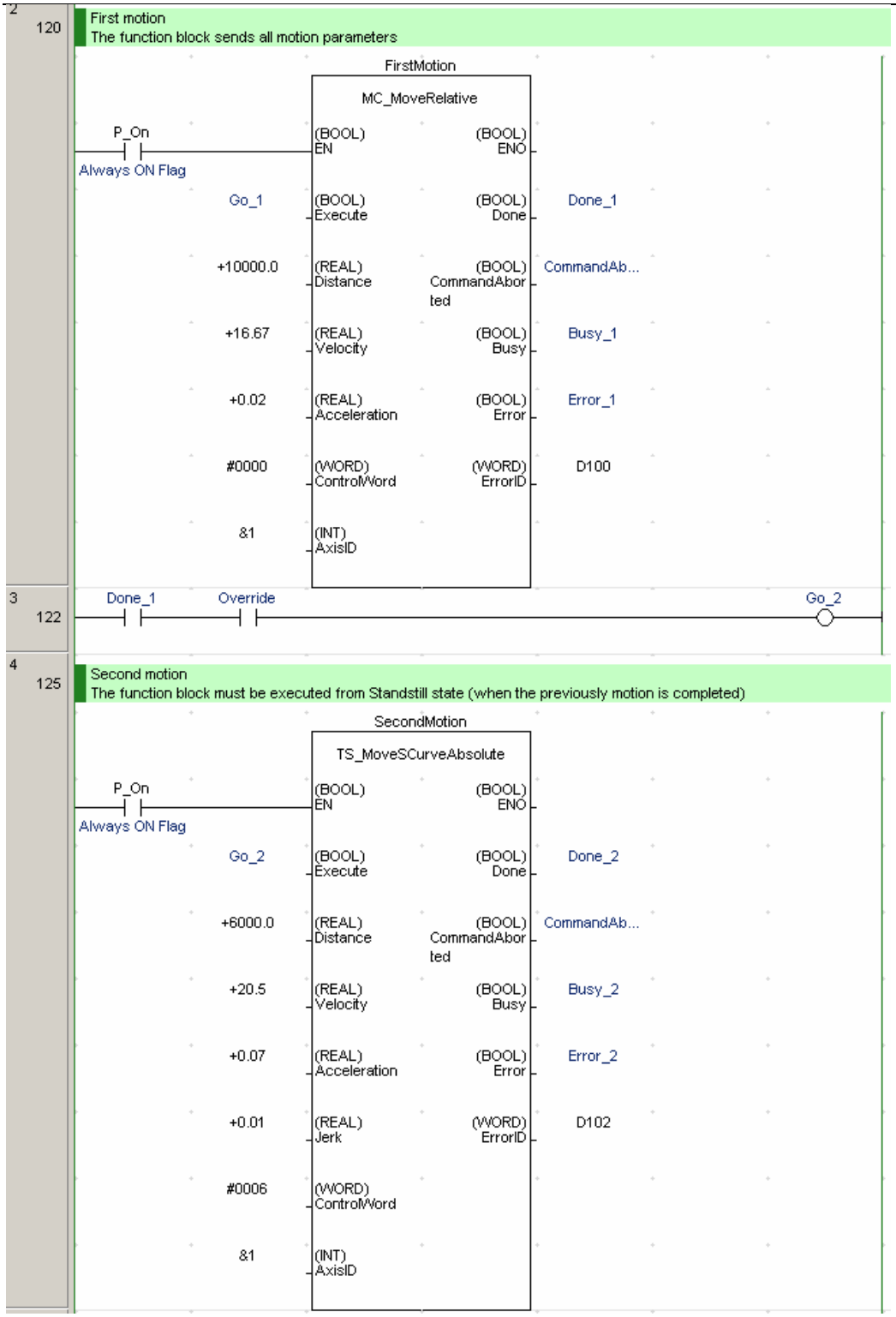
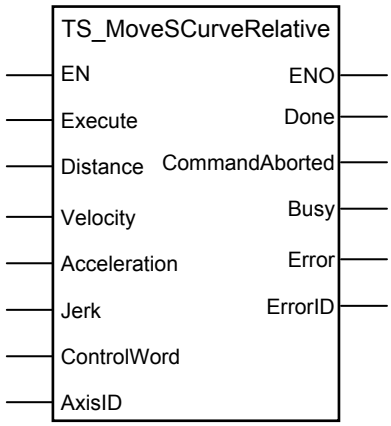


Figure 3.4. *Time diagram for 2 concatenated motions*


2
120
First motion
The function block sends all motion parameters
FirstMotion
MC_MoveRelative
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_1
(BOOL) Execute
(BOOL) Done
Done_1
+10000.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+16.67
(REAL) Velocity
(BOOL) Busy
Busy_1
+0.02
(REAL) Acceleration
(BOOL) Error
Error_1
#0000
(WORD) ControlWord
(WORD) ErrorID
D100
&1
(INT) AxisID
3
122
Done_1
Override
Go_2
4
125
Second motion
The function block must be executed from Standstill state (when the previously motion is completed)
SecondMotion
TS_MoveSCurveAbsolute
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_2
(BOOL) Execute
(BOOL) Done
Done_2
+6000.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+20.5
(REAL) Velocity
(BOOL) Busy
Busy_2
+0.07
(REAL) Acceleration
(BOOL) Error
Error_2
+0.01
(REAL) Jerk
(WORD) ErrorID
D102
#0006
(WORD) ControlWord
&1
(INT) AxisID

3.6.5 TS_MoveSCurveRelative

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Send motion commands at rising edge
	Distance	REAL	Position increment expressed in TML position internal units
	Velocity	REAL	The slew speed expressed in TML speed internal units
	Acceleration	REAL	Acceleration rate expressed in TML acceleration internal units.
	Jerk	REAL	Acceleration's maximum jerk expressed in TML jerk internal units
	ControlWord	BOOL	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	The motion is complete
	CommandAborted	BOOL	The function block is aborted by another motion function block
	Busy	BOOL	Signal the function block is waiting for motion complete
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block programs a relative positioning with an S-curve shape of the speed. This shape is due to the jerk limitation, leading to a trapezoidal or triangular profile for the acceleration and an S-curve profile for the speed. You specify the position to reach plus the velocity (maximum travel speed), the maximum acceleration/deceleration rate and the jerk rate.

The function block can be executed only from **Standstill** state. During motion the parameters should not be changed. Therefore when executing successive motions with TS_MoveSCurveAbsolute function blocks, you should wait for the previous motion to end before setting the new motion parameters and starting next motion.

When the motion is stopped with function block MC_Stop, the deceleration phase can be done in 2 ways:

- Smooth, using an S-curve speed profile, when ControlWord.3 = 0, or
- Fast using a trapezoidal speed profile, when ControlWord.3 = 1

On detecting a rising edge at the **Execute** input, the function block starts sending motion commands and sets the **Busy** output. The **Busy** output remains set until the drive/motor signals target reached, moment when the function block sets the **Done** output and resets **Busy**.

The **CommandAborted** output is set if another motion function block sends motion commands to the same drive/motor. If a drive's error register information is received during the execution of the function block, the output **Error** is set and its value is passed to **ErrorID**.

All outputs remain set until **Execute** input is reset, but at least for one block call.

During motion execution the axis is transferred to **DiscreteMotion** state, when the target position is reached (**Done** output is set) the axis is transferred to **Standstill** state.

Once set, the motion parameters are memorized on the drive/motor. If you intend to use values previously defined (by a different motion function block or different instance of the same function block) for the acceleration rate, the velocity or the position to reach, you don't need to send their values again in the following trapezoidal profiles. Through **ControlWord** input you can select the motion parameters sent by the function block to the drive/motor.

If ControlWord.15 is set then the value read from **AxisID** input is interpreted as a group ID. The function block will send the motion commands to drives/motors members of the group. In this case the done output is set after the motion commands are sent.

Table 3.7 *ControlWord bits description*

Bit	Value	Description
0	0	Send position value
	1	Don't send the position value
1	0	Send the speed value
	1	Don't send the speed value
2	0	Send the acceleration value
	1	Don't send the acceleration value
3	0	Smooth stop
	1	Faster stop
4-14	0	Reserved for new features
15	0	The motion commands are sent to a single drive/motor
	1	The motion commands are sent to a group of drives/motors.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

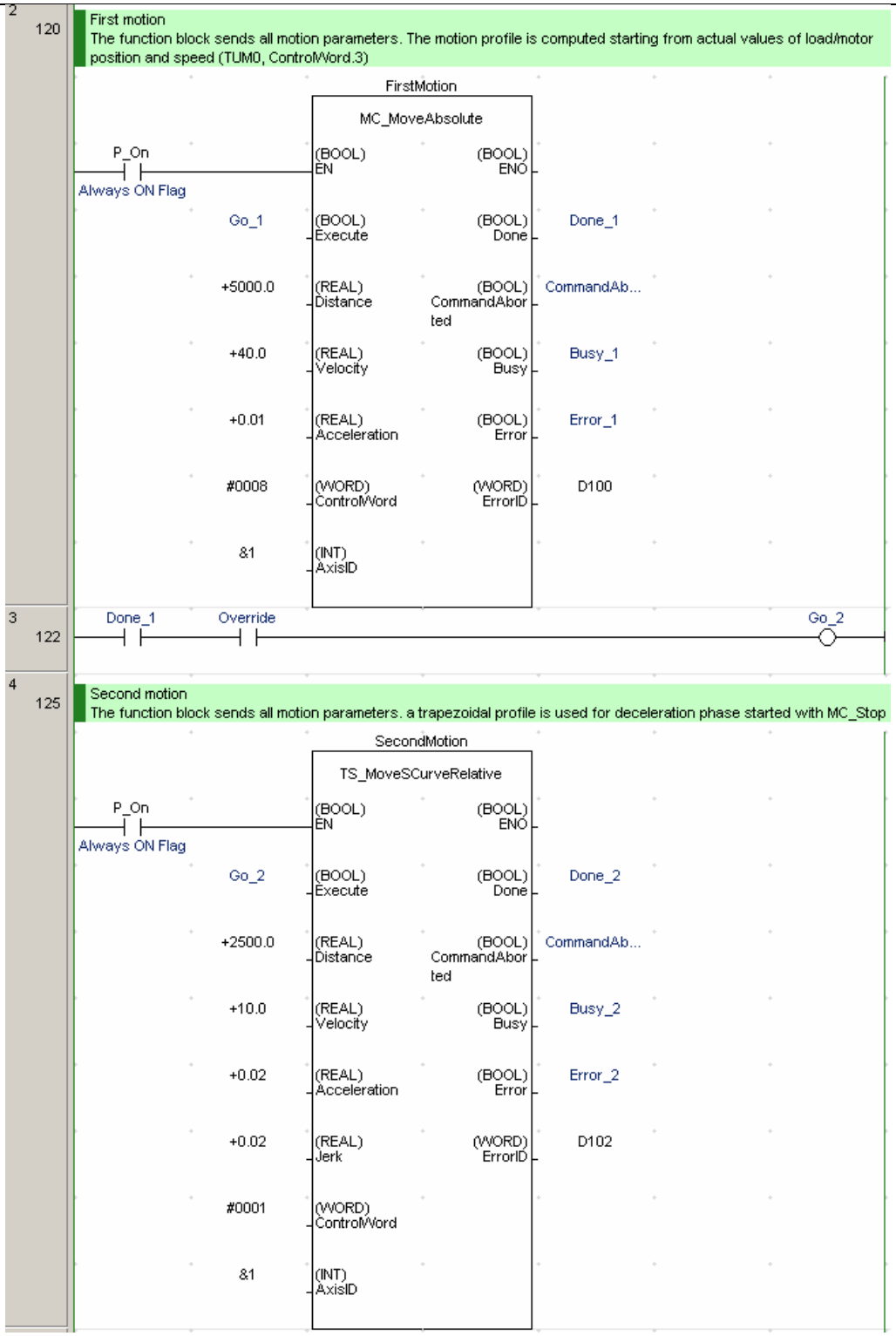
Remarks:

1. *The function block requires the drive/motor position loop to be closed. During the drive/motor setup select* ***Position*** *at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between discrete motion (position control) and continuous motion (speed control) you must also close the speed loop and perform the tuning of the speed controller*.
3. *For stopping with trapezoidal profile the deceleration value is read from TML parameter CDEC (fixed@0x0859). Use TS_WriteFixedParameter in order to change its value. The default value is 0.5 IU.*

Example:

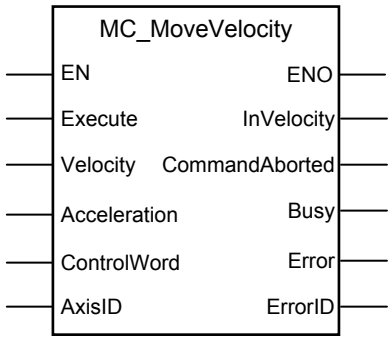
The example implements two consecutive motions for the drive/motor with axis ID =1. The first motion is an absolute positioning at 5000 IU implemented with function block MC_MoveAbsolute. The second motion is a relative positioning with 2500 IU increment, implemented with TS_MoveSCurveRelative.

The first motion is triggered when “Go_1” becomes TRUE. The second motion can be triggered only when “Done_1” is set, when the first motion is completed and “Override” is set too. If function block MC_Stop is issued during second motion the drive/motor stops using a trapezoidal profile with deceleration read from CDEC TML parameter.


2
120
First motion
The function block sends all motion parameters. The motion profile is computed starting from actual values of load/motor position and speed (TUM0, ControlWord.3)
FirstMotion
MC_MoveAbsolute
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_1
(BOOL) Execute
(BOOL) Done
Done_1
+5000.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+40.0
(REAL) Velocity
(BOOL) Busy
Busy_1
+0.01
(REAL) Acceleration
(BOOL) Error
Error_1
#0008
(WORD) ControlWord
(WORD) ErrorID
D100
&1
(INT) AxisID
3
122
Done_1
Override
Go_2
4
125
Second motion
The function block sends all motion parameters. a trapezoidal profile is used for deceleration phase started with MC_Stop
SecondMotion
TS_MoveSCurveRelative
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_2
(BOOL) Execute
(BOOL) Done
Done_2
+2500.0
(REAL) Distance
(BOOL) CommandAborted
CommandAb...
+10.0
(REAL) Velocity
(BOOL) Busy
Busy_2
+0.02
(REAL) Acceleration
(BOOL) Error
Error_2
+0.02
(REAL) Jerk
(WORD) ErrorID
D102
#0001
(WORD) ControlWord
&1
(INT) AxisID

3.6.6 MC_MoveVelocity

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Send motion commands at rising edge
	Velocity	REAL	The jog speed expressed in TML speed internal units
	Acceleration	REAL	Acceleration rate expressed in TML acceleration internal units.
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	InVelocity	BOOL	Signal that commanded velocity reached
	CommandAborted	BOOL	The function block is aborted by another motion function block
	Busy	BOOL	The function block is waiting for velocity to reach the target speed command
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block commands a trapezoidal speed profile.

At the rising edge of the **Execute** input, the motion command is sent and the output **Busy** is set. The axis is transferred to ContinuousMotion state. When the jog speed is reached, the **InVelocity** output is set and **Busy** is reset.

The **CommandAborted** is set if another function block starts sending motion commands. If a drive's motion error register information is received during the execution of the function block, the output **Error** is set and its value is passed to **ErrorID**.

All outputs remain set until **Execute** input is reset, but at least for one block call.

During motion execution the axis is transferred to **ContinuousMotion** state and keeps the state until another motion command is executed.

Once set, the motion parameters are memorized on the drive/motor. If you intend to use values previously defined (by a different motion function block or different instance of the same function block) for the acceleration rate, the velocity you don't need to send their values again in the following trapezoidal profiles. Through **ControlWord** input you can select the motion parameters sent by the function block to the drive/motor.

If ControlWord.15 is set then the value read from **AxisID** input is interpreted as a group ID. The function block will send the motion commands to drives/motors members of the group. In this case the done output is set after the motion commands are sent.

Table 3.8 *ControlWord bits description*

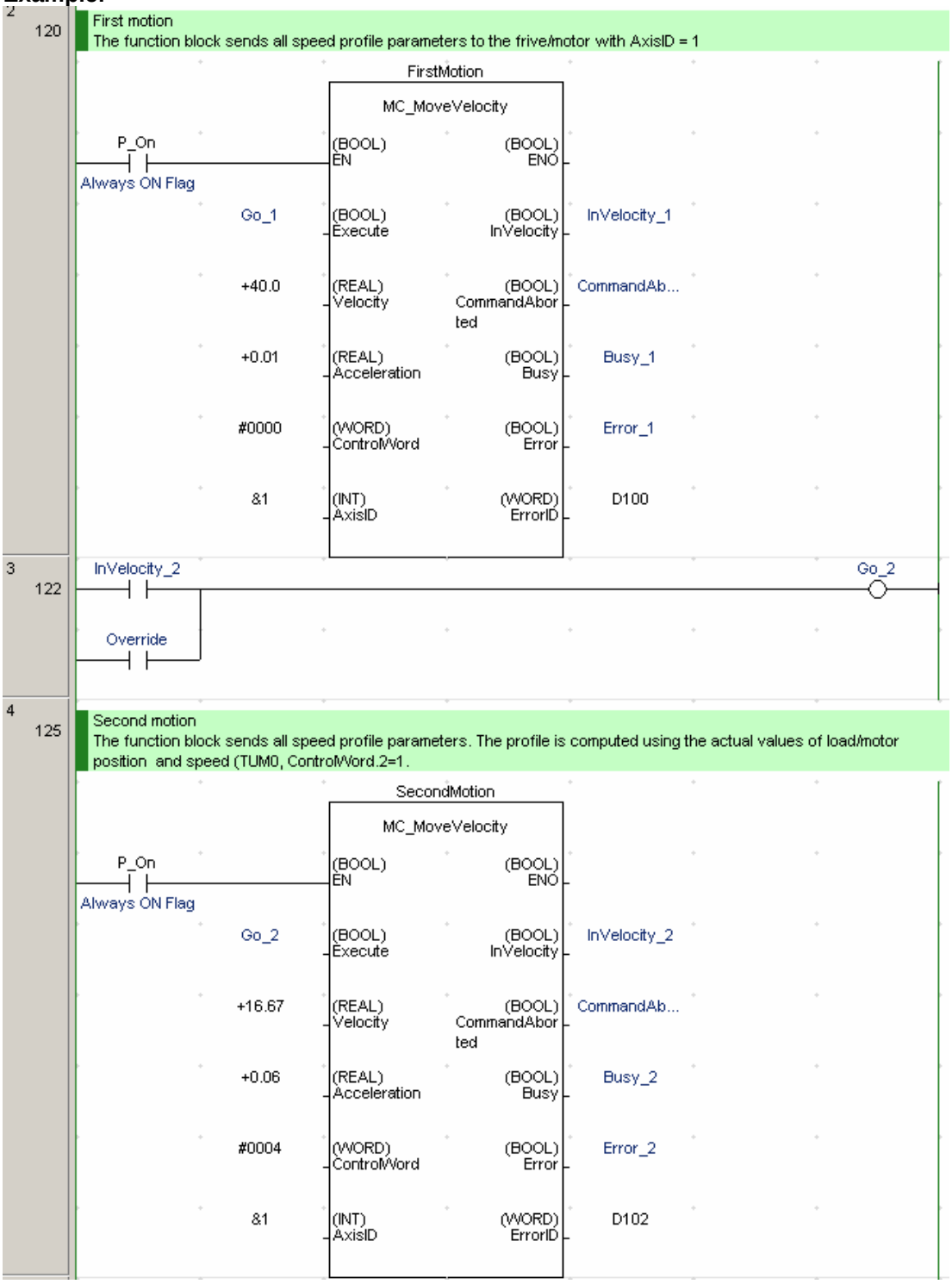
Bit	Value	Description
0	0	Send speed value
	1	Don't send the speed value
1	0	Send the acceleration value
	1	Don't send the acceleration value
2	0	Target Update Mode 1 (TUM1). Generates new trajectory starting from the actual values of position and speed reference
	1	Target Update Mode 0 (TUM0). Generates new trajectory starting from the actual values of load/motor position and speed
3-14	0	Reserved
15	0	The motion commands are sent to a single drive/motor
	1	The motion commands are sent to a group of drives/motors.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Remarks:

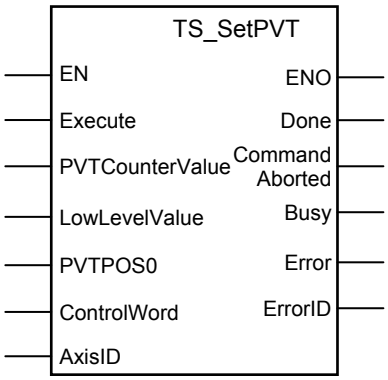
1. *The function block requires drive/motor speed loop to be closed. During the drive/motor setup, in the **Drive Setup** dialogue, select **Speed** at Control Mode and perform the speed controller tuning.*
2. *If the application requires switching between continuous motion (speed control) and discrete motion (position control) the position loop must be closed, also. In the **Drive setup** dialogue select **Position** at **Control Mode**, then enter the **Advanced** dialogue and close the speed loop. After the selection perform the tuning of the position controller and speed controller.*

Example:


2
120
First motion
The function block sends all speed profile parameters to the frive/motor with AxisID = 1
FirstMotion
MC_MoveVelocity
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_1
(BOOL) Execute
(BOOL) InVelocity
InVelocity_1
+40.0
(REAL) Velocity
(BOOL) CommandAborted
CommandAb...
+0.01
(REAL) Acceleration
(BOOL) Busy
Busy_1
#0000
(WORD) ControlWord
(BOOL) Error
Error_1
&1
(INT) AxisID
(WORD) ErrorID
D100
3
122
InVelocity_2
Go_2
Override
4
125
Second motion
The function block sends all speed profile parameters. The profile is computed using the actual values of load/motor position and speed (TUM0, ControlWord.2=1.
SecondMotion
MC_MoveVelocity
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Go_2
(BOOL) Execute
(BOOL) InVelocity
InVelocity_2
+16.67
(REAL) Velocity
(BOOL) CommandAborted
CommandAb...
+0.06
(REAL) Acceleration
(BOOL) Busy
Busy_2
#0004
(WORD) ControlWord
(BOOL) Error
Error_2
&1
(INT) AxisID
(WORD) ErrorID
D102

3.6.7 TS_SetPVT

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	At rising edge the command is sent
	PVTCounterValue	INT	The new counter value for the first PVT point
	LowLevelValue	INT	The new value for low buffer signaling
	PVTPOS0	REAL	The initial position for absolute PVT mode
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Signal the path described through PVT points ended correctly, the last point has velocity zero
	CommandAborted	BOOL	The motion command is aborted by another motion command
	Busy	BOOL	The function block is processing the commands
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function programs a drive/motor to work in PVT motion mode. In PVT motion mode the drive/motor performs a positioning path described through a series of points. Each point specifies the desired **P**osition, **V**elocity and **T**ime, i.e. contains a PVT data. Between the points the built-in reference generator performs a 3rd order interpolation.

Remark: *The function block just programs the drive/motor for PVT mode, the motion mode is activated from function block TS_PVTPoint, and therefore the TS_PVTPoint must follow TS_SetPVT call. Also, the function block TS_PVTPoint sends the PVT points to the drive/motor.*

A key factor for getting a correct positioning path in PVT mode is to set correctly the distance in time between the points. Typically this is 10-20ms, the shorter the better. If the distance in time between the PVT points is too big, the 3rd order interpolation may lead to important variations compared with the desired path.

The PVT motion mode can be started only when the previous motion is complete. However, you can switch at any moment to another motion mode.

The PVT mode can be relative (ControlWord.0 = 0) or absolute (ControlWord.0 = 1). In the absolute mode, each PVT point specifies the position to reach. The initial position may be either the current position reference TML variable **TPOS** (ControlWord.12=1) or a preset value read from the TML parameter **PVTPOS0** (ControlWord.12=0). In the relative mode, each PVT point specifies the position increment relative to the previous point. In both cases, the time is relative to the previous point i.e. represents the duration of a PVT segment. For the first PVT point, the time is measured from the starting of the PVT mode.

Each time when the drive receives a new PVT point, it is saved into the PVT buffer. The reference generator empties the buffer as the PVT points are executed. The PVT buffer is of type FIFO (first in, first out). The default length of the PVT buffer is 7 PVT points. Each entry in the buffer is made up of 9 words, so the default length of the PVT buffer in terms of how much memory space is reserved is 63 (3Fh) words. The drive/motor automatically sends messages to the PLC when the buffer is full, low or empty. The messages contain the PVT status (TML variable **PVTSTS**). The buffer full condition occurs when the number of PVT points in the buffer is equal with the buffer size. The buffer low condition occurs when the number of PVT points in the buffer is less or equal with a programmable value. When ControlWord.7=1 the buffer low level is programmed with the value read from **LowLevelValue** input. The buffer empty condition occurs when the buffer is empty and the execution of the last PVT point is over. When the PVT buffer becomes empty the drive/motor:

- Remains in PVT mode if the velocity of last PVT point executed is zero and waits for new points to receive
- Enters in quick stop mode if the velocity of last PVT point executed is not zero

Therefore, a correct PVT sequence must always end with a last PVT point having velocity zero.

Remarks:

1. *The PVT and PT modes share the same buffer. Therefore the TML parameters and variables associated with the buffer management are the same.*
2. *Both the PVT buffer size and its start address are programmable via TML parameters (int@0x0864) and PVTBUFLEN (int@0x0865). Therefore if needed, the PVT buffer size can be substantially increased. Use TS_WriteIntegerParameter to change the PVT buffer parameters.*

Table 3.9. *ControWord bits description*

Bit	Value	Description
0	0	The PVT mode is relative
	1	The PVT mode is absolute
1	0	Target update mode 1. Generates new trajectory starting from the actual values of position and speed reference (i.e. don't update the reference values with load/motor position and speed)
	1	Target update mode 0. Generates new trajectory starting from the actual values of load/motor position and speed (i.e. update the reference values with load/motor position and speed)
2	0	Don't change the value of PVTPOS0
	1	Change PVTPOS0 with the value read from **PVTPOS0** input
3-6	0	Reserved

7	0	No change in the buffer low parameter
	1	Change the buffer low parameter with the value specified in **LowLevelValue**
8-11	0	Reserved
12	0	If PVT mode is set as absolute, the initial position is taken from TML parameter PVTPOS0 (default = 0). The initial position is used to compute the distance to move up to the first PVT point.
	1	If PVT mode is set as absolute, the initial position is taken from TML variable **TPOS**. The initial position is used to compute the distance to move up to the first PVT point.
13	0	No change to the internal integrity counter
	1	Change internal integrity counter with the value specified in **PVTCounterValue**
14	0	Integrity checking is active
	1	Integrity checking inactive
15	0	Nothing
	1	Clear PVT buffer

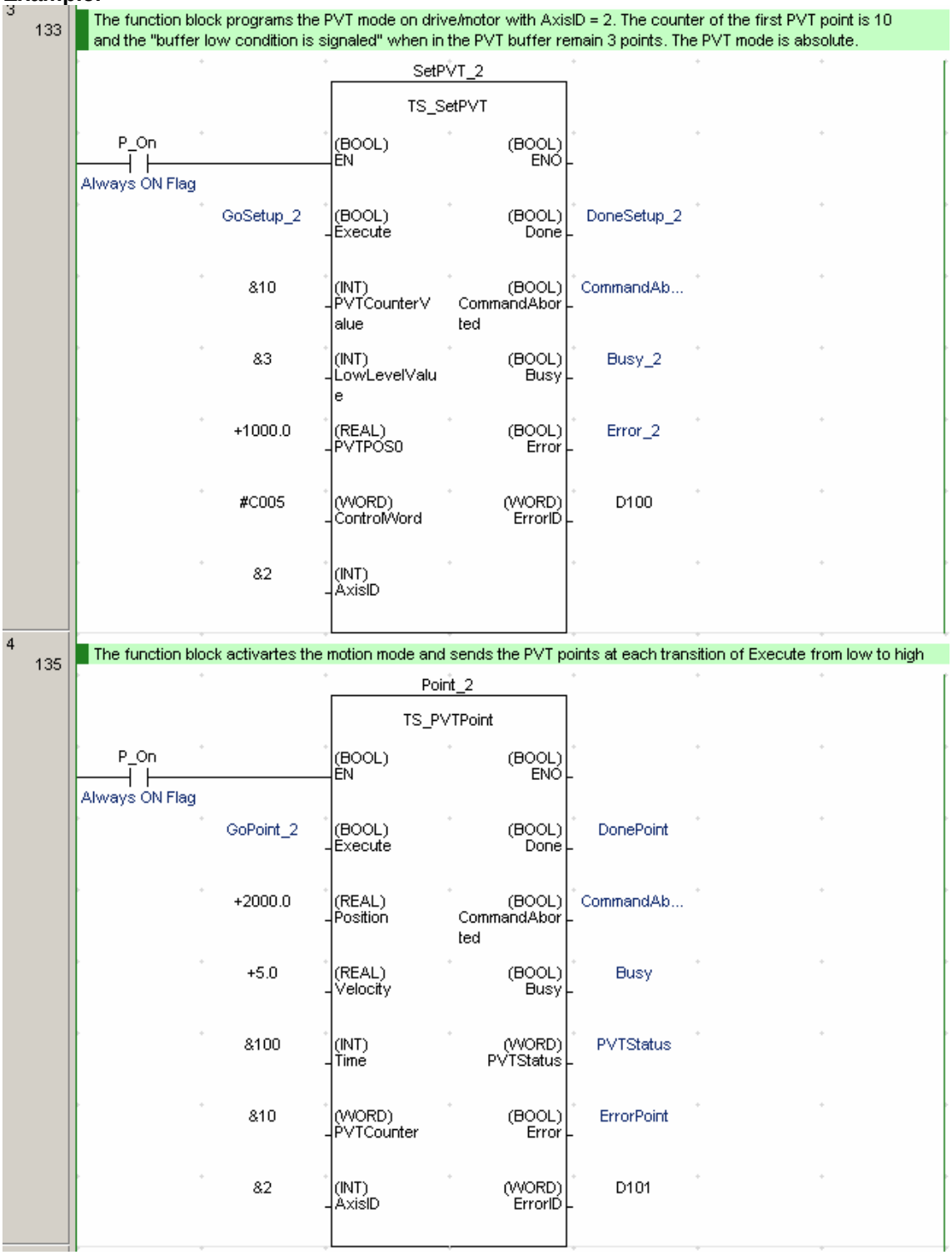
On detecting a rising edge at **Execute** input the function block reads the inputs and sends the motion commands to the drive/motor. The output **Busy** is set and remains set until the function block **TS_PVTPoint** signals “motion mode active” moment when the **Done** output is set and **Busy** reset.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Remarks:

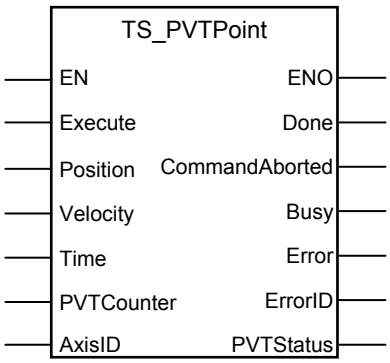
1. *The function block requires the drive/motor position loop to be closed. During the drive/motor setup select **Position** at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between discrete motion (position control) and continuous motion (speed control) the speed loop must be closed, also. In the **Drive setup** dialogue select **Position** at **Control Mode**, then enter the **Advanced** dialogue and close the speed loop. After the selection perform the tuning of the position controller and speed controller.*

Example:


3
133
The function block programs the PVT mode on drive/motor with AxisID = 2. The counter of the first PVT point is 10 and the "buffer low condition is signaled" when in the PVT buffer remain 3 points. The PVT mode is absolute.
SetPVT_2
TS_SetPVT
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
GoSetup_2
(BOOL) Execute
(BOOL) Done
DoneSetup_2
&10
(INT) PVTCounterValue
(BOOL) CommandAborted
CommandAb...
&3
(INT) LowLevelValue
(BOOL) Busy
Busy_2
+1000.0
(REAL) PVTPOS0
(BOOL) Error
Error_2
#C005
(WORD) ControlWord
(WORD) ErrorID
D100
&2
(INT) AxisID
4
135
The function block activartes the motion mode and sends the PVT points at each transition of Execute from low to high
Point_2
TS_PVTPoint
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
GoPoint_2
(BOOL) Execute
(BOOL) Done
DonePoint
+2000.0
(REAL) Position
(BOOL) CommandAborted
CommandAb...
+5.0
(REAL) Velocity
(BOOL) Busy
Busy
&100
(INT) Time
(WORD) PVTStatus
PVTStatus
&10
(WORD) PVTCounter
(BOOL) Error
ErrorPoint
&2
(INT) AxisID
(WORD) ErrorID
D101

3.6.8 TS_PVTPoint

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	At rising edge the command is sent
	Position	REAL	Represents the position to be reached at the end of the PVT segment
	Velocity	REAL	Is the velocity at the of the PVT segment
	Time	INT	Represents the time interval of the PVT segment expressed in TML time internal units
	PVTCounter	INT	The counter value for the current point. The maximum value for the counter is 127.
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Signal that commanded velocity is reached
	CommandAborted	BOOL	The motion command is aborted by another motion command
	Busy	BOOL	The function block is processing the command
	Error	BOOL	Signal if an error message was received from the drive/motor while **!Done**
	ErrorID	WORD	Information about the error occurred
	PVTStatus	WORD	The PVT status received from the drive/motor

Description: The function block activates the PVT motion mode and sends the PVT points to a drive/motor previously programmed with function block TS_SetPVT to work in PVT motion mode. In PVT motion mode the drive/motor performs a positioning path described through a series of points. Each point specifies the desired **P**osition, **V**elocity and **T**ime, i.e. contains a PVT data. Between the points the built-in reference generator performs a 3rd order interpolation.

At the first rising edge of the **Execute** input the function block reads the inputs, sends the PVT point to the drive/motor and activates the motion mode. The output **Busy** is set and remains set until the drive signals the motion is complete, moment when the **Done** output is set and **Busy** reset. If the last point sent has non-zero velocity the drive/motor signals the condition **quick stop** and the function block sets the output **Error**.

Every time the function block detects a rising edge at input **Execute** reads the inputs and sends a PVT point.

When the PLC receives a message from the drive/motor with PVT status the function transfers to output **PVTStatus** its value.

Table 3.10 *PVT motion mode status*

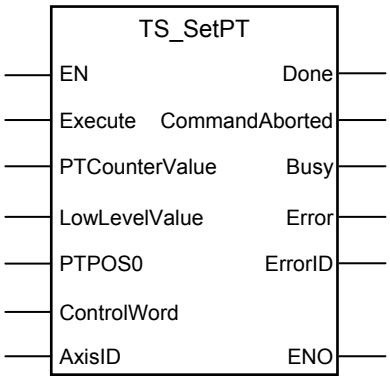
BIT	VALUE	DESCRIPTION
15	0	PVT buffer is not empty
	1	PVT buffer is empty – there is no PVT point in the buffer and the execution of the current PVT segment is over. If **PVTSENDOFF** = 0 (default), the drive/motor will send the **PVTSTS** each time this bit goes from 0 to 1
14	0	PVT buffer is not low
	1	PVT buffer is low – the number of PVT points from the buffer is equal or less than the low limit set using **SETPVT**. If **PVTSENDOFF** = 0 (default), the drive will send the **PVTSTS** each time this bit goes from 0 to 1
13	0	PVT buffer is not full
	1	PVT buffer is full – the number of PVT points from the buffer is equal with the buffer dimension. If **PVTSENDOFF** = 0 (default), the drive will send the **PVTSTS** each time this bit goes from 0 to 1
12	0	No integrity counter error
	1	Integrity counter error. If integrity counter error checking is enabled and **PVTSENDOFF** = 0 (default), the drive will send the **PVTSTS** each time this bit goes from 0 to 1
11	0	The drive has kept the PVT motion mode after a PVT buffer empty condition, because the velocity of the last PVT point was 0
	1	The drive has performed a Quick stop, following a PVT buffer empty condition, because the velocity of the last PVT point was different from 0
10	0	Normal operation. Data received are PVT points
	1	A PT point was received while PVT mode is active. The PT point was discharged. If **PVTSENDOFF** = 0 (default), the drive/motor will send the **PVTSTS** each time this bit goes from 0 to 1
9..7	0	Reserved
6..0	0..127	Current integrity counter value

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Remarks: *The function block just sends the PVT points, the motion mode is programmed with function block TS_SetPVT, and therefore the TS_PVTPoint must follow TS_SetPVT call.*

3.6.9 TS_SetPT

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	At rising edge the command is sent
	PTCounterValue	INT	The new counter value for the first PT point
	LowLevelValue	INT	The new value for low buffer signaling
	PTPOS0	REAL	The initial position for absolute PT mode
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Signal that PT motion mode programmed successfully
	CommandAborted	BOOL	The motion command is aborted by another motion command
	Busy	BOOL	The function block is processing the command
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block programs a drive/motor to work in PT motion mode. In PT motion mode the drive/motor performs a positioning path described through a series of points. Each point specifies the desired **P**osition and **T**ime, i.e. contains a PT data. Between the points the built-in reference generator performs a linear interpolation.

Remarks: *The function block just programs the drive/motor for PT mode, the motion mode is activated from function block TS_PTPoint, and therefore the TS_PTPoint must follow TS_SetPT call. Also, the function block TS_PTPoint sends the PT points to the drive/motor.*

The PT motion mode can be started only when the previous motion is complete. However, you can switch at any moment to another motion mode by calling a different function block with

The PT mode can be relative (ControlWord.0 = 0) or absolute (ControlWord.0 = 1). In the absolute mode, each PT point specifies the position to reach. The initial position may be either the

current position reference TML variable **TPOS** (ControlWord.12=1) or a preset value read from the TML parameter **PTPOS0** (ControlWord.12=0). In the relative mode, each PT point specifies the position increment relative to the previous point. In both cases, the time is relative to the previous point i.e. represents the duration of a PT segment. For the first PT point, the time is measured from the starting of the PT mode.

Each time when a new PT point is received it is saved into the PT buffer. The reference generator empties the buffer as the PT points are executed. The PT buffer is of type FIFO (first in, first out). The default length of the PT buffer is 7 PT points. Each entry in the buffer is made up of 9 words, so the default length of the PT buffer in terms of how much memory space is reserved is 63 (3Fh) words. The drive/motor automatically sends messages to the PLC when the buffer is full, low or empty. The messages contain the PT status (TML variable **PVTSTS**). The buffer full condition occurs when the number of PT points in the buffer is equal with the buffer size. The buffer low condition occurs when the number of PT points in the buffer is less or equal with a programmable value. When ControlWord.7=1 the buffer low level is programmed with the value read from **LowLevelValue** input. The buffer empty condition occurs when the buffer is empty and the execution of the last PVT point is over. When the PT buffer becomes empty the drive/motor hen the PT buffer becomes empty the drive/motor keeps the position reference unchanged.

Remark:

1. *The PVT and PT modes share the same buffer. Therefore the TML parameters and variables associated with the buffer management are the same.*
2. *Both the PT buffer size and its start address are programmable via TML parameters (int@0x0864) and PVTBUFLEN (int@0x0865). Therefore if needed, the PT buffer size can be substantially increased. Use TS_WriteIntegerParameter to change the PT buffer parameters.*

On detecting a rising edge at **Execute** input the function block reads the inputs and sends the motion commands to the drive/motor. The output **Busy** is set and remains set until the function block **TS_PTPoint** signals “motion mode active”, moment when the **Done** output is set and **Busy** reset.

Table 3.11. *ControWord bits description*

Bit	Value	Description
0	0	The PT mode is relative
	1	The PT mode is absolute
1	0	Target update mode 1. Generates new trajectory starting from the actual values of position and speed reference (i.e. don’t update the reference values with load/motor position and speed)
	1	Target update mode 0. Generates new trajectory starting from the actual values of load/motor position and speed (i.e. update the reference values with load/motor position and speed)
7	0	No change in the buffer low parameter
	1	Change the buffer low parameter with the value specified in **LowLevelvalue**
11-8	0	Reserved for new features
12	0	If PT mode is set as absolute, the initial position is taken from TML parameter PVTPOS0 (default = 0). The initial position is used to compute the distance to move up to the first PVT point.

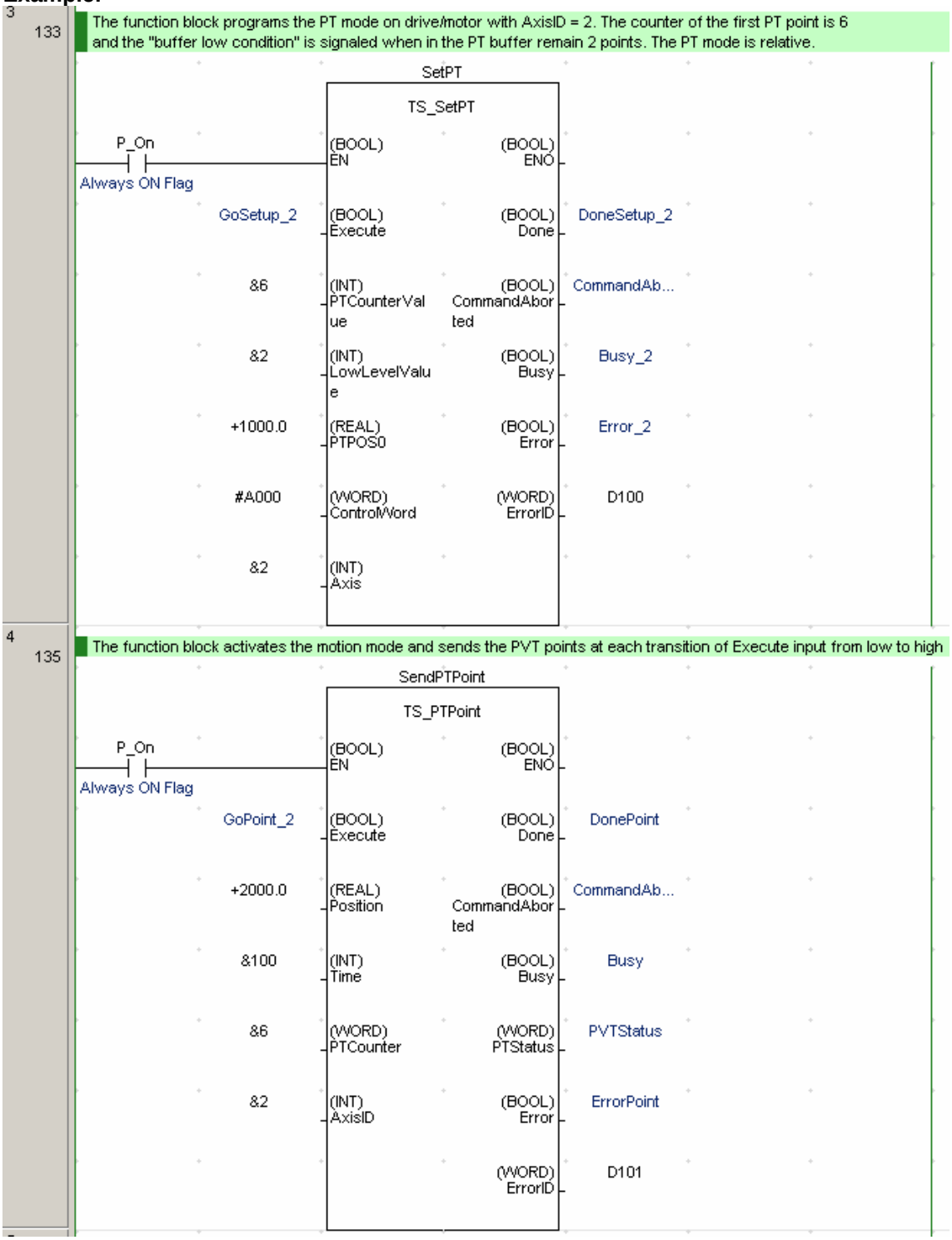
	1	If PT mode is set as absolute, the initial position is taken from TML variable **TPOS**. The initial position is used to compute the distance to move up to the first PT point.
13	0	No change to the internal integrity counter
	1	Change internal integrity counter with the value specified in **PTCounterValue**
14	0	Integrity checking is active
	1	Integrity checking inactive
15	0	Nothing
	1	Clear PT buffer

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Remarks:

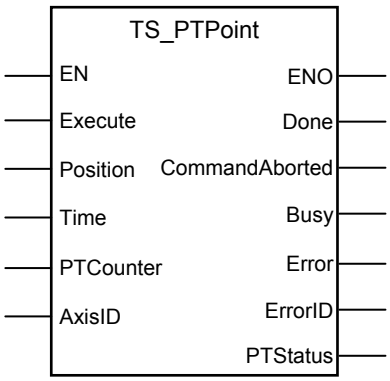
1. *The function block requires the drive/motor position loop to be closed. During the drive/motor setup select **Position** at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between discrete motion (position control) and continuous motion (speed control) the speed loop must be closed, also. In the **Drive setup** dialogue select **Position** at **Control Mode**, then enter the **Advanced** dialogue and close the speed loop. After the selection perform the tuning of the position controller and speed controller.*

Example:


3
133
The function block programs the PT mode on drive/motor with AxisID = 2. The counter of the first PT point is 6 and the "buffer low condition" is signaled when in the PT buffer remain 2 points. The PT mode is relative.
SetPT
TS_SetPT
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
GoSetup_2
(BOOL) Execute
(BOOL) Done
DoneSetup_2
&6
(INT) PTCounterValue
(BOOL) CommandAborted
CommandAb...
&2
(INT) LowLevelValue
(BOOL) Busy
Busy_2
+1000.0
(REAL) PTPOS0
(BOOL) Error
Error_2
#A000
(WORD) ControlWord
(WORD) ErrorID
D100
&2
(INT) Axis
4
135
The function block activates the motion mode and sends the PVT points at each transition of Execute input from low to high
SendPTPoint
TS_PTPoint
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
GoPoint_2
(BOOL) Execute
(BOOL) Done
DonePoint
+2000.0
(REAL) Position
(BOOL) CommandAborted
CommandAb...
&100
(INT) Time
(BOOL) Busy
Busy
&6
(WORD) PTCounter
(WORD) PTStatus
PVTStatus
&2
(INT) AxisID
(BOOL) Error
ErrorPoint
(WORD) ErrorID
D101

3.6.10 TS_PTPoint

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	At rising edge the command is sent
	Position	REAL	Represents the position to be reached at the end of the PT segment
	Time	INT	Represents the time interval of the PT segment
	PTCounter	INT	The counter value for the current point. The maximum value for the counter is 127.
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Signal the commanded motion is completed after the last point PT sent
	CommandAborted	BOOL	The motion command is aborted by another motion command
	Busy	BOOL	The function block is processing the command
	Error	BOOL	Signal if an error message was received from the drive/motor while **!Done**
	ErrorID	WORD	Information about the error occurred
	PTStatus	WORD	The PT status received from the drive/motor

Description: The function block activates the PT motion mode and sends the PT points to a drive/motor previously programmed with function block TS_SetPT to work in PT motion mode. In PT motion mode the drive/motor performs a positioning path described through a series of points. Each point specifies the desired **P**osition and **T**ime, i.e. contains a PT data. Between the points the built-in reference generator performs a linear interpolation.

On detecting a rising edge at **Execute** input the function block reads the inputs sends the first PT point to the drive/motor and activates the motion mode. The output **Busy** is set and remains set until the drive signals the motion is complete, moment when the **Done** output is set and **Busy** reset. If the last point sent to has non-zero velocity the drive/motor signals the condition quick stop and the function block sets the output **Error**.

The function block sends a PT point every time detects a rising edge at input **Execute**.

When the PLC receives a message from the drive/motor with PT status the function transfers to output **PTStatus** its value.

Table 3.12 *PT motion mode status*

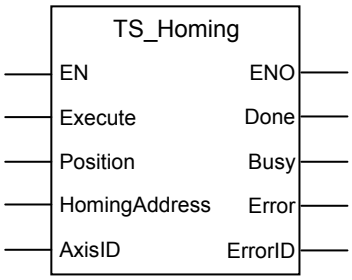
BIT	VALUE	DESCRIPTION
15	0	PT buffer is not empty
	1	PT buffer is empty – there is no PT point in the buffer and the execution of the current PT segment is over. If **PVTSENDOFF** = 0 (default), the drive/motor will send the **PVTSTS** each time this bit goes from 0 to 1
14	0	PT buffer is not low
	1	PT buffer is low – the number of PT points from the buffer is equal or less than the low limit set using **SETPT**. If **PVTSENDOFF** = 0 (default), the drive will send the **PVTSTS** each time this bit goes from 0 to 1
13	0	PT buffer is not full
	1	PT buffer is full – the number of PT points from the buffer is equal with the buffer dimension. If **PVTSENDOFF** = 0 (default), the drive will send the **PVTSTS** each time this bit goes from 0 to 1
12	0	No integrity counter error
	1	Integrity counter error. If the integrity counter error checking is enabled and **PVTSENDOFF** = 0 (default), the drive will send the **PVTSTS** each time this bit goes from 0 to 1
11	0	Reserved
10	0	Normal operation. Data received are PT points
	1	A PVT point was received while PT mode is active. The PVT point was discharged. If **PVTSENDOFF** = 0 (default), the drive/motor will send the **PVTSTS** each time this bit goes from 0 to 1
9..7	0	Reserved
6..0	0..127	Current integrity counter value

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Remarks: *The function block just sends the PT points, the motion mode is programmed with function block TS_SetPT, and therefore the TS_PTPoint must follow TS_SetPT call.*

3.6.11 TS_Homing

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Execute	BOOL	Send the stop command at rising edge
	Position	REAL	The position set at the end of the homing procedure
	HomingAddress	WORD	Program memory address of the homing procedure
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	State of function block execution
	Done	BOOL	The homing procedure ended successfully
	Busy	BOOL	The function is waiting for homing procedure to end
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block starts a homing procedure programmed on the drive/motor with **AxisID**.

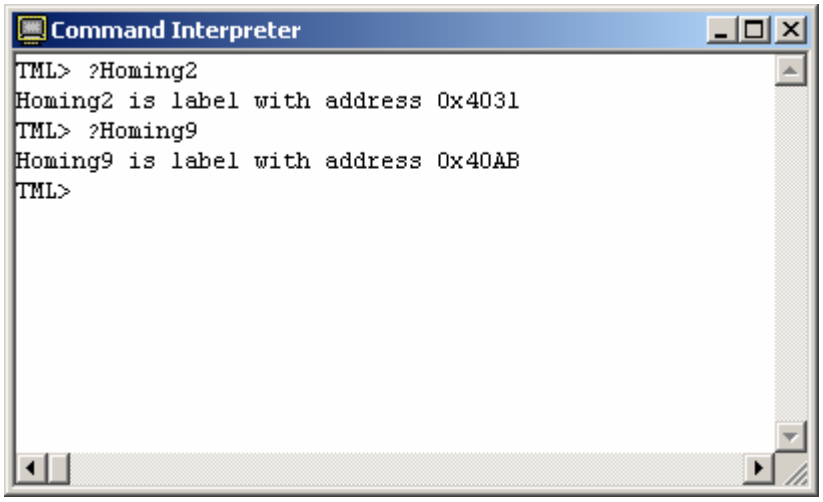
The search for the home position can be done in numerous ways. In order to offer maximum flexibility, the TML does not impose the homing procedures but lets you define your own, according with your application needs.

The homing procedures are programmed and downloaded on the drive/motor using **EasyMotion Studio** tool**.** Technosoft provides for each intelligent drive/motor a collection of up to 32 homing procedures. You may use any of these homing procedures as they are, or use them as a starting point for your own homing routines.

Steps to program the drive/motor for homing:

- Start **EasyMotion Studio** and load the project associated to your application
- Select from the list with all the defined homing procedures one or several to be used
- Select the menu command **Application | Motion | Build** to compile and link the TML program, the result is a file with out extension ready to be downloaded to the drive/motor.
- Establish the communication with your drive and download the TML program with menu command **Application | Motion | Download Program**

- Open the **Command interpreter** window and interrogate the drive/motor about the program memory address of the homing procedure using the command "**?Homing**XX" where **XX** is the homing procedure number.



- In the PLC program call TS_Homing with the homing procedure address and the home position.

On detecting a rising edge at the input **Execute,** the function block sends the home position and starts the homing procedure on the drive/motor. The **Busy** output is set and remains set until the drive/motor signals homing procedure completed. When the drive/motor message is received the function block sets the **Done** output and resets **Busy**. Outputs **Error** and **ErrorID** have no associated functionality.

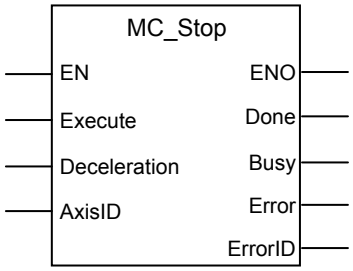
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:


137
The function block starts a homing procedure downloaded on drive/motor with AxisID = 2.
The homing procedure start address is 0x4031.
Homing
TS_Homing
P_On
Always ON Flag
(BOOL)
EN
(BOOL)
ENO
GoHoming_2
(BOOL)
Execute
(BOOL)
Done
HomingDone_2
+2000.0
(REAL)
Position
(BOOL)
CommandAborted
HomingAbort...
#4031
(WORD)
RunAddress
(BOOL)
Busy
HomingBusy_2
&2
(INT)
AxisID
(BOOL)
Error
HomingError_1
(WORD)
ErrorID

3.6.12 MC_Stop

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Execute	BOOL	Send the stop command at rising edge
	Deceleration	REAL	Deceleration rate expressed in TML acceleration internal units.
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	State of function block execution
	Done	BOOL	Signal drive velocity zero.
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block stops the drive/motor motor with the deceleration rate set at **Deceleration** input. The drive/motor decelerates following a trapezoidal speed profile. The stop command is sent to the drive/motor with **AxisID**.

On detecting a rising edge at **Execute** input the function block sends the stop command and sets the **Busy** output. **Busy** remains set until the drive/motor signals “velocity zero”, moment when the function block sets the **Done** output and resets the **Busy**.

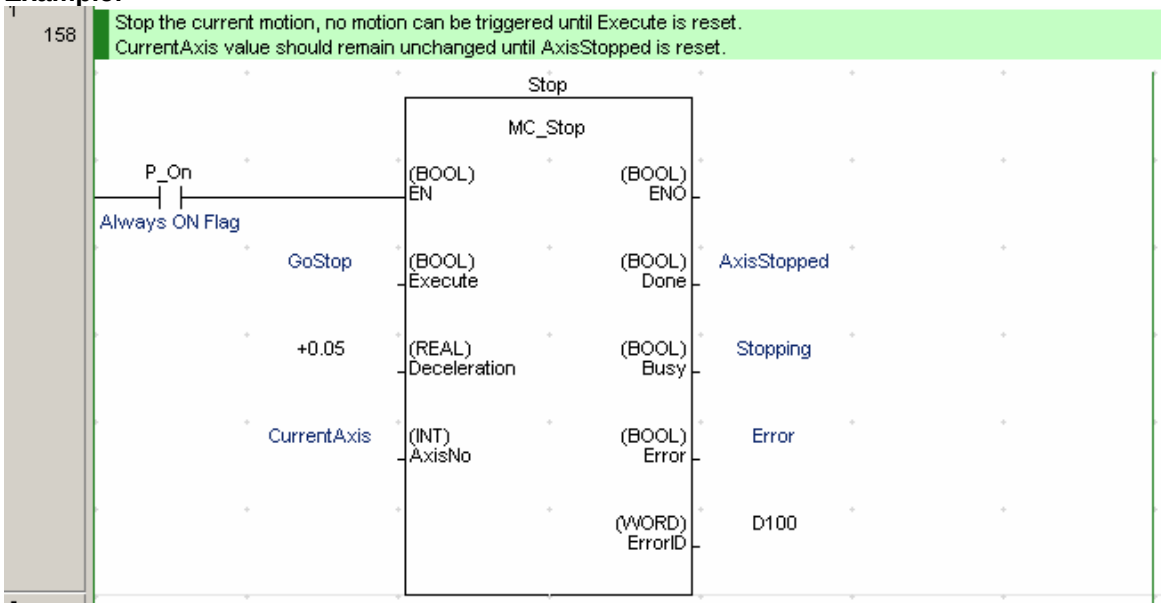
If the drive/motor error register is received, while the output **Done** is not set, the output Error is set and the its value is passed to **ErrorID.**

When the function block is called the axis is transferred to state **Stopping** and aborts the current motion function block. In the **Stopping** state no motion function block can be called.

The axis remains in state **Stopping** until the input **Execute** is reset when the axis is transferred to state **Standstill**.

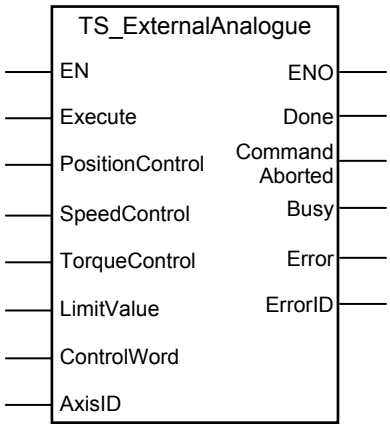
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:


158
Stop the current motion, no motion can be triggered until Execute is reset.
CurrentAxis value should remain unchanged until AxisStopped is reset.
Stop
MC_Stop
P_On
Always ON Flag
(BOOL)
EN
(BOOL)
ENO
GoStop
(BOOL)
Execute
(BOOL)
Done
AxisStopped
+0.05
(REAL)
Deceleration
(BOOL)
Busy
Stopping
CurrentAxis
(INT)
AxisNo
(BOOL)
Error
Error
(WORD)
ErrorID
D100

3.6.13 TS_ExternalAnalogue

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	At rising edge the commands are sent
	PositionControl	BOOL	The analogue reference is a position reference
	SpeedControl	BOOL	The analogue reference is a speed reference
	TorqueControl	BOOL	The analogue reference is a torque reference
	Limit value	REAL	Speed/acceleration limit value for position/speed control
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Motion commands sent successfully
	CommandAborted	BOOL	The function block is aborted by another motion function block
	Busy	BOOL	The function block is processing the command
	Error	BOOL	Signal if an error has occurred in the execution of the function block.
	ErrorID	WORD	Information about the error occurred

Description: The function block programs the drive/motor to work with an external analogue reference read via a dedicated analogue input (10-bit resolution). The analogue signal can be interpreted as a position, speed or torque analogue reference. Through inputs **PositionControl**, **SpeedControl** and **TorqueControl** you select the control type performed by the drive/motor. Table 3.1 shows the correspondence between the control type, inputs and the axis state.

Remark: *During the drive/motor setup, in the* ***Drive setup*** *dialogue, you have to:*

1. *Select the appropriate control type for your application at Control Mode.*
2. *Perform the tuning of controllers associated with the selected control mode.*

3. *Setup the analogue reference. You specify the reference values corresponding to the upper and lower limits of the analogue input. In addition, a dead-band symmetrical interval and its center point inside the analogue input range may be defined.*

Table 3.13 *Control type-Axis state*

Control type	Input	Axis state
Position	PositionControl	DiscreteMotion
Speed	SpeedControl	ContinuousMotion
Torque	TorqueControl	ContinuousMotion

On detecting a rising edge at the **Execute** input, the function block checks the inputs **PositionControl**, **SpeedControl**, **TorqueControl** and starts sending the motion commands according with the **first input found set**. The output **Busy** is set and remains set until the last motion command is sent, moment when the function block sets the **Done** output and resets **Busy**.

If none of the inputs **PositionControl**, **SpeedControl**, **TorqueControl** is set when the **Execute** rising edge is detected the function block sets the **Error** output and transfers the value 0x8000 at the output **ErrorID**.

The **CommandAborted** output is set if another motion function block sends motion commands to the same drive/motor.

In position control you can limit the maximum speed at sudden changes of the position reference and thus to reduce the mechanical shocks. In speed control you can limit the maximum acceleration at sudden changes of the speed reference and thus to get a smoother transition. These features are activated by setting ControlWord.0 = 1 and the maximum speed/acceleration value at **LimitValue** input.

In torque control you can choose how often to read the analogue input: at each slow loop sampling period or at each fast loop sampling period. The selection is made through ControlWord.1.

If ControlWord.15 is set then the axis ID read from **AxisID** input is interpreted as a group ID. The function block will send the motion commands to drives/motors members of the group. In this case the **Done** output is set after the motion commands are sent.

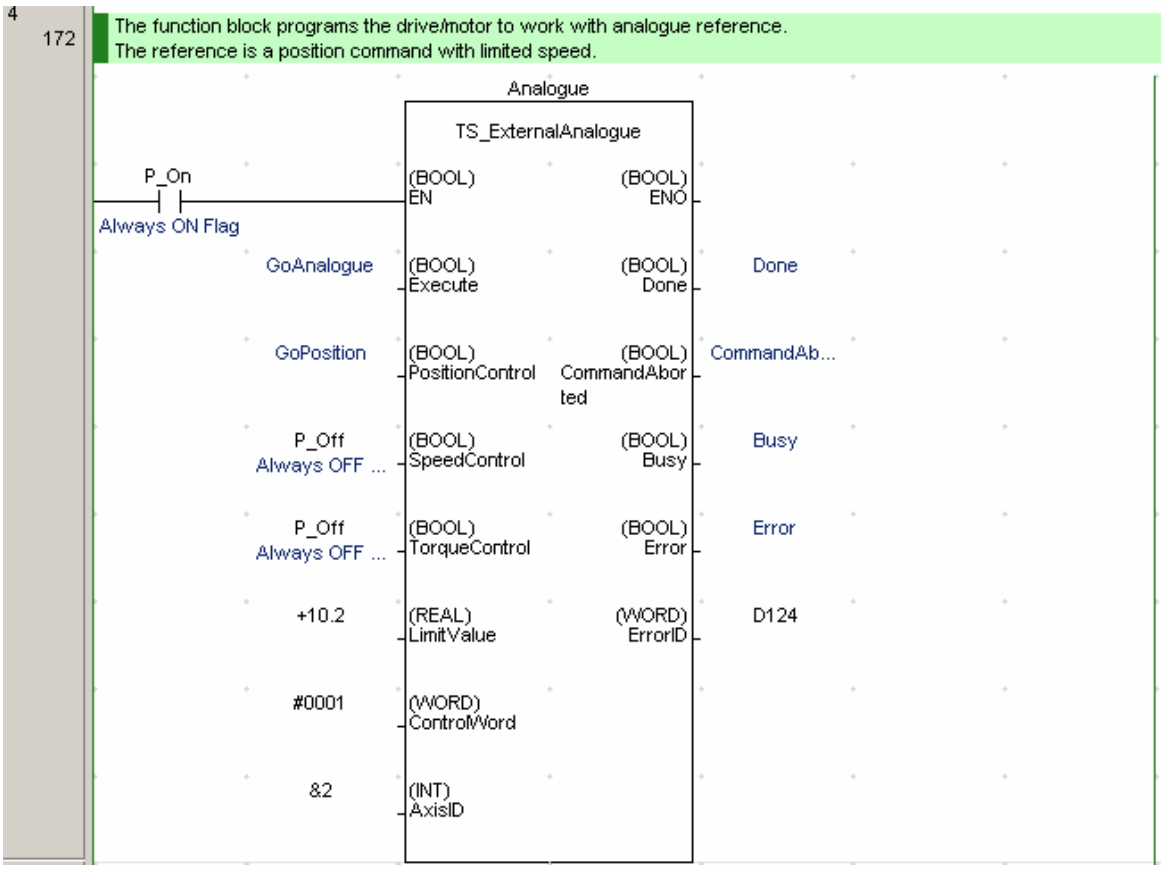
Table 3.14 *ControlWord bits description*

Bit	Value	Description
0	0	No speed/acceleration limit for position/speed external mode
	1	The speed/acceleration is limited for position/speed external mode
1	0	The analogue reference is read at each slow loop sampling
	1	The analogue reference is read at each fast loop sampling
2-14	0	Reserved for new features
15	0	The motion commands are sent to a single drive/motor
	1	The motion commands are sent to a group of drives/motors.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

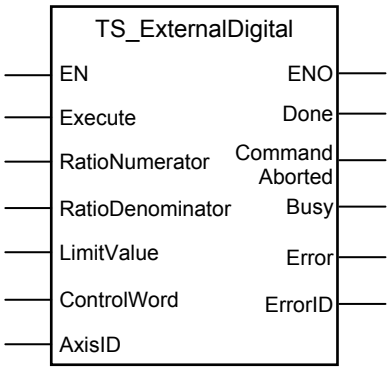
Example:

The example programs the drive/motor to work with analogue external reference. The analogue signal is interpreted as a position reference with limited speed.



3.6.14 TS_ExternalDigital

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	At rising edge the command is sent
	RatioNumerator	INT	Gear ratio numerator
	RatioDenominator	INT	Gear ratio denominator
	LimitValue	REAL	Acceleration limit value
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	The motion command send successfully
	CommandAborted	BOOL	The motion command is aborted by another motion command
	Busy	BOOL	The function block is processing the command
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block programs the drive/motor to work with an external digital reference provided as pulse & direction or quadrature encoder signals. In either case, the drive/motor performs a position control with the reference computed from the external signals.

Remarks:

1. *The function block requires the drive/motor position loop to be closed. During the drive/motor setup select **Position** at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between discrete motion (position control) and continuous motion (speed control) the speed loop must be closed, also. In the **Drive setup** dialogue select **Position** at **Control Mode**, then enter the **Advanced** dialogue and close the speed loop. After the selection perform the tuning of the position controller and speed controller.*

3. *The option for the input signals: pulse & direction or quadrature encoder is established during the drive/motor setup.*

On detecting a rising edge at the **Execute** input the function block starts sending the motion commands. The output **Busy** is set and remains set until the last motion command is sent, moment when the function block sets the **Done** output and resets **Busy**.

The **CommandAborted** output is set if another motion function block sends motion commands to the same drive/motor.

Set ControlWord.0 if you want to follow the external position reference with a different ratio than 1:1. The gear ratio is specified as a ratio of 2 integer values: **RatioNumerator** / **RatioDenominator**. The **RatioNumerator** value is signed, while the **RatioDenominator** is unsigned. The sign indicates the direction of movement: positive – same as the external reference, negative – reversed to the external reference.

You can limit the maximum acceleration at sudden changes of the external reference and thus to get a smoother transition. This feature is activated by setting ControlWord.0 = 1 and the maximum acceleration value at **LimitValue** input. The default value for acceleration is 0.5IU.

If ControlWord.15 is set then the axis ID read from **AxisID** input is interpreted as a group ID. The function block will send the motion commands to drives/motors members of the group. In this case the **Done** output is set after the motion commands are sent.

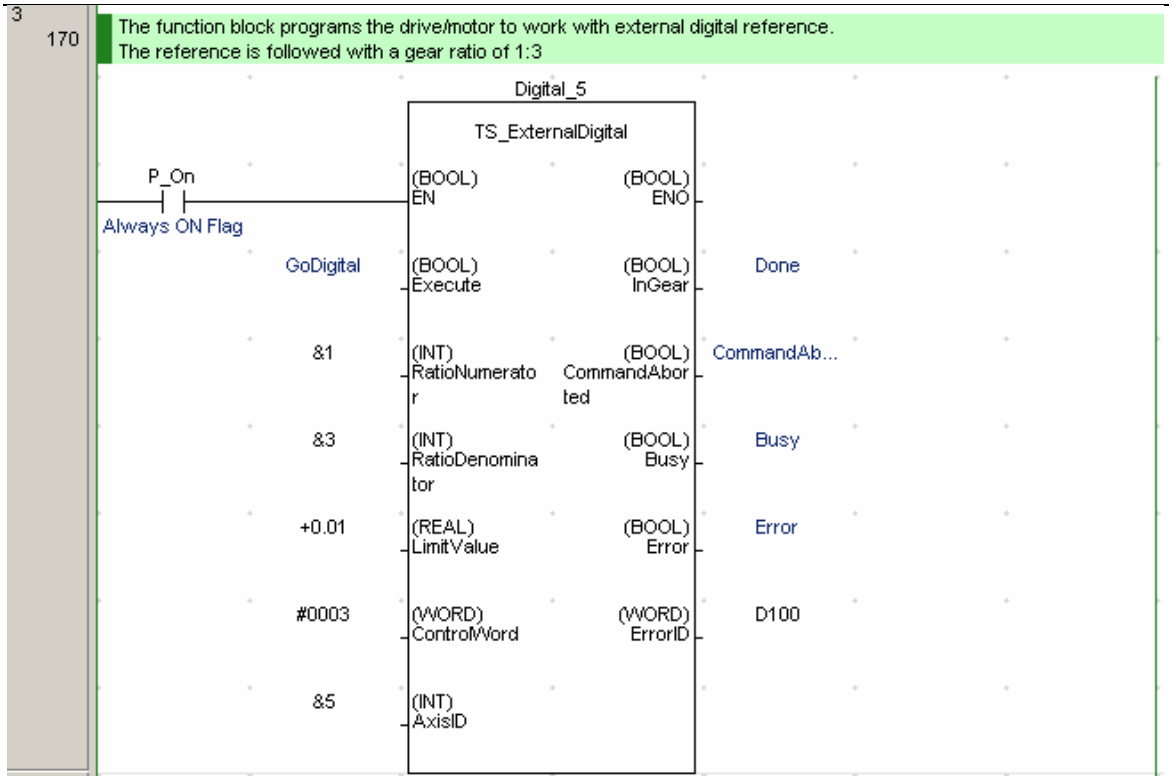
Table 3.15 *ControlWord bits description*

Bit	Value	Description
0	0	The acceleration is not limited
	1	The acceleration is limited at the value read from LimitValue input
1	0	The gear ratio is 1:1
	1	The gear ratio is **RatioNumerator**: **RatioDenominator**
2-14	0	Reserved for new features
15	0	The motion commands are sent to a single drive/motor
	1	The motion commands are sent to a group of drives/motors.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

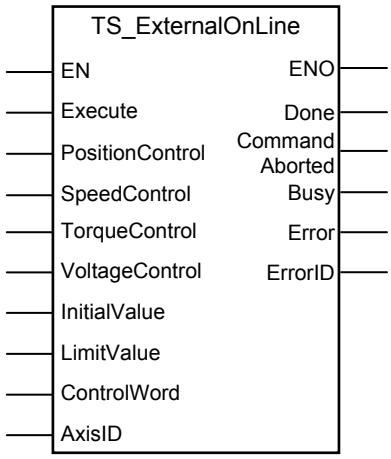
Example:

The example programs a drive/motor to use digital external reference. The position reference is followed with a ratio of 1:3 and the drive/motor acceleration is limited to 0.01IU.


3
170
The function block programs the drive/motor to work with external digital reference.
The reference is followed with a gear ratio of 1:3
Digital_5
TS_ExternalDigital
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
GoDigital
(BOOL) Execute
(BOOL) InGear
Done
&1
(INT) RatioNumerator
(BOOL) CommandAborted
CommandAb...
&3
(INT) RatioDenominator
(BOOL) Busy
Busy
+0.01
(REAL) LimitValue
(BOOL) Error
Error
#0003
(WORD) ControlWord
(WORD) ErrorID
D100
&5
(INT) AxisID

3.6.15 TS_ExternalOnLine

Symbol:



Parameter description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Send motion commands at rising edge
	PositionControl	BOOL	The external reference is a position reference
	SpeedControl	BOOL	The external reference is a speed reference
	TorqueControl	BOOL	The external reference is a torque reference
	VoltageControl	BOOL	The external reference is a voltage reference
	InitialValue	REAL	The initial value for reference received on-line.
	LimitValue	REAL	Speed/acceleration limit value for position/speed control
	ControlWord	WORD	Selects the block options
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	The motion commands sent successfully
	CommandAborted	BOOL	The motion command is aborted by another motion command
	Busy	BOOL	The function block is processing the command
	Error	BOOL	Signal if an error has occurred in the execution of the function block or an error message was received from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block programs the drive/motor to work with an external reference read via a communication channel. Depending on the control mode chosen, the external reference is saved in one of the TML variables:

- **EREFP**, which becomes the position reference if the input **PositionControl** is set
- **EREFS**, which becomes the speed reference if the input **SpeedControl** is set
- **EREFT**, which becomes the torque reference if the input **TorqueControl** is set
- **EREFV**, which becomes voltage reference if the input **VoltageControl** is set

Remark: *During the drive/motor setup, in the* ***Drive setup*** *dialogue, you have to:*

1. *Select the appropriate control type for your application at Control Mode.*
2. *Perform the tuning of controllers associated with the selected control mode.*

When the external reference is sent from the PLC use:

- Function block **TS_WriteLongParameter** to update the variable **EREFP** (long@0x02A8)
- Function block **TS_WriteFixedParameter** to update the variable **EREFS** (fixed@0x02A8)
- Function block **TS_WriteIntegerParameter** to update the variable **EREFT** or **EREFV** (int@0x02A9)

Table 3.16

Control type	Input	Axis state
Position	PositionControl	DiscreteMotion
Speed	SpeedControl	ContinuousMotion
Torque	TorqueControl	ContinuousMotion
Voltage	VoltageControl	ContinuousMotion

On detecting a rising edge at the **Execute** input, the function block checks the inputs **PositionControl**, **SpeedControl**, **TorqueControl, VoltageControl** and starts sending the motion commands according with the **first input found set**. The output **Busy** is set and remains set until the last motion command is sent, moment when the function block sets the **Done** output and resets **Busy**.

If none of the inputs **PositionControl**, **SpeedControl**, **TorqueControl or VoltageControl** is set when the **Execute** rising edge is detected the function block sets the **Error** output and transfers the value 0x8000 at the output **ErrorID**.

The **CommandAborted** output is set if another motion function block sends motion commands to the same drive/motor.

In position control you can limit the maximum speed at sudden changes of the position reference and thus to reduce the mechanical shocks. In speed control you can limit the maximum acceleration at sudden changes of the speed reference and thus to get a smoother transition. These features are activated by setting ControlWord.0 = 1 and the maximum speed/acceleration value at **LimitValue** input.

If the external device starts sending the reference AFTER the external online mode is activated, it may be necessary to initialize EREFP, EREFS, EREFT or EREFV. If ControlWord.1 is found set, when the **Execute** rising edge is detected, the function block initialize the TML variable (EREFP, EREFS, EREFT or EREFV) with the value read from **InitialValue** input.

If ControlWord.15 is set then the axis ID read from **AxisID** input is interpreted as a group ID. The function block will send the motion commands to drives/motors members of the group. In this case the **Done** output is set after the motion commands are sent.

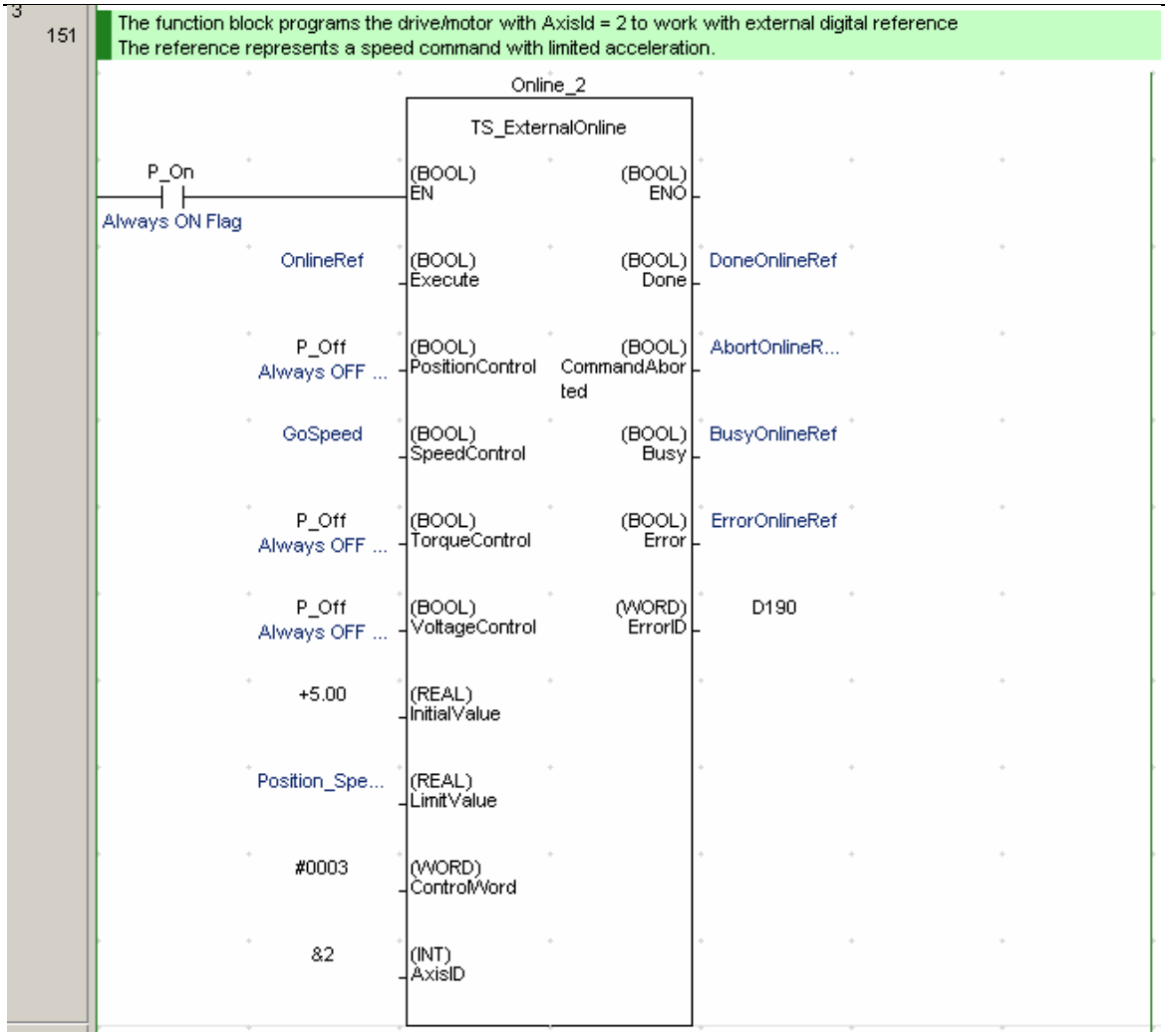
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Table 3.17 *ControlWord bits description*

Bit	Value	Description
0	0	No speed/acceleration limit for position/speed external mode
	1	The speed/acceleration is limited for position/speed external mode
1	0	Don't initialize EREFP, EREFS, EREFT or EREFV
	1	Initialize EREFP, EREFS, EREFT or EREFV
2-14	0	Reserved for new features
15	0	The motion commands are sent to one drive/motor
	1	The motion commands are sent to a group of drives/motors.

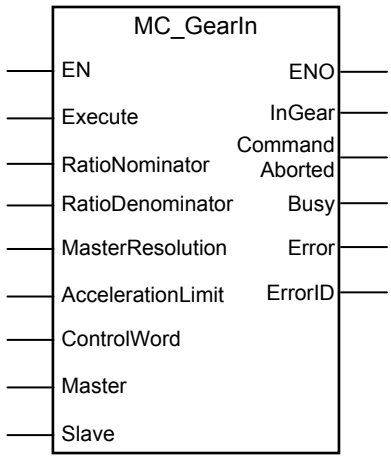
Example:

The example programs the drive/motor to use external online reference. The reference represents a speed command with limited acceleration. The TML variable EREFS is initialized with 5.00IU.


3
151
The function block programs the drive/motor with AxisId = 2 to work with external digital reference
The reference represents a speed command with limited acceleration.
Online_2
TS_ExternalOnline
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
OnlineRef
(BOOL) Execute
(BOOL) Done
DoneOnlineRef
P_Off
Always OFF ...
(BOOL) PositionControl
(BOOL) CommandAborted
AbortOnlineR...
GoSpeed
(BOOL) SpeedControl
(BOOL) Busy
BusyOnlineRef
P_Off
Always OFF ...
(BOOL) TorqueControl
(BOOL) Error
ErrorOnlineRef
P_Off
Always OFF ...
(BOOL) VoltageControl
(WORD) ErrorID
D190
+5.00
(REAL) InitialValue
Position_Spe...
(REAL) LimitValue
#0003
(WORD) ControlWord
&2
(INT) AxisID

3.6.16 MC_GearIn

Symbol:



Parameter description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Send motion commands at rising edge
	RatioNumerator	INT	Gear ratio numerator (negative or positive)
	RatioDenominator	INT	Gear ratio denominator (only positive)
	MasterResolution	DINT	Master's position sensor resolution expressed
	AccelerationLimit	REAL	Acceleration limit when the slave is coupling expressed in TML acceleration units
	ControlWord	WORD	Selects the block options
	Master	INT	Axis ID of the drive/motor set as master
	Slave	INT	Axis ID of the drive/motor set as slave
Output	ENO	BOOL	Status of function block execution
	InGear	BOOL	Is set when the gear ratio reached
	CommandAborted	BOOL	The function block is aborted by another function block
	Busy	BOOL	The function block is waiting the slave drive to reach the gear ratio
	Error	BOOL	Is set if drive's/motor's error register is received while **!InGear**
	ErrorID	WORD	Information about the error occurred

Description: The function block programs a drive/motor to operate as slave in electronic gearing. In electronic gearing slave mode the drive/motor performs a position control. At each slow loop sampling period, the slave computes the master's position increment and multiplies it with its programmed gear ratio. The result is the slave's position reference increment, which added to the previous slave position reference gives the new slave position reference.

The gear ratio is a fixed value containing the result of the division **RatioNumerator** / **RatioDenominator**. **RatioNumerator** is a signed integer, while the **RatioDenominator** is unsigned integer.

RatioNumerator sign indicates the direction of movement: positive – same as the master, negative – reversed to the master. **RatioNumerator** and **RatioDenominator** are used by an automatic compensation procedure that eliminates the round off errors, which occur when the gear ratio is an irrational number like: 1/3 (Slave = 1, Master = 3).

The slave can get the master position in two ways:

1. Via a communication channel (ControlWord.5 = 0), from a drive/motor set as master with function block TS_SetMaster
2. Via an external digital reference of type pulse & direction or quadrature encoder (ControlWord.5 = 1)

When master position is provided via the external digital interface, the slave computes the master position by counting the pulse & direction or quadrature encoder signals. The initial value of the master position is set by default to 0. Use function block TS_WriteLongParameter to change its value by writing the desired value in the TML variable APOS2 (long@0x091C).

Remarks:

1. *The function block requires drive/motor position loop to be closed. During the drive/motor setup select **Position** at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between synchronized motion (position control) and continuous motion (speed control), i.e. use MC_GearOut, you must also close the speed loop and perform the speed controller tuning*.
3. *Use function block **TS_SetMaster** to program a drive/motor as master in electronic gearing*
4. *When the reference is read from second encoder or pulse & direction inputs you don't need to program a drive/motor as master in electronic gearing*

You can smooth the slave coupling with the master, by limiting the maximum acceleration on the slave. This is particularly useful when the slave must couple with a master running at high speed. Setting ControlWord.7 and the maximum acceleration value **LimitValue** input activates the feature.

On detecting a rising edge at the **Execute** input the function block starts sending the motion commands. The output **Busy** is set and remains set until the drive/motor signals gear ratio reached, moment when the function block sets the **InGear** output and resets **Busy**.

If another motion function block starts sending motion commands while the output **InGear** is not set, the function block is aborted and the **CommandAborted** output is set. Also if the drive error register is received while the output **InGear** is not set, the output **Error** is set and the error register value is passed to **ErrorID**.

All outputs remain set until **Execute** input is reset, but at least for one block call.

During motion execution the axis is transferred into **SynchronousMotion** state and remains in this state until a new motion function block (except MC_GearIn and MC_CamIn) starts sending motion commands to the same drive/motor.

***Table 3.18** ControlWord bits description*

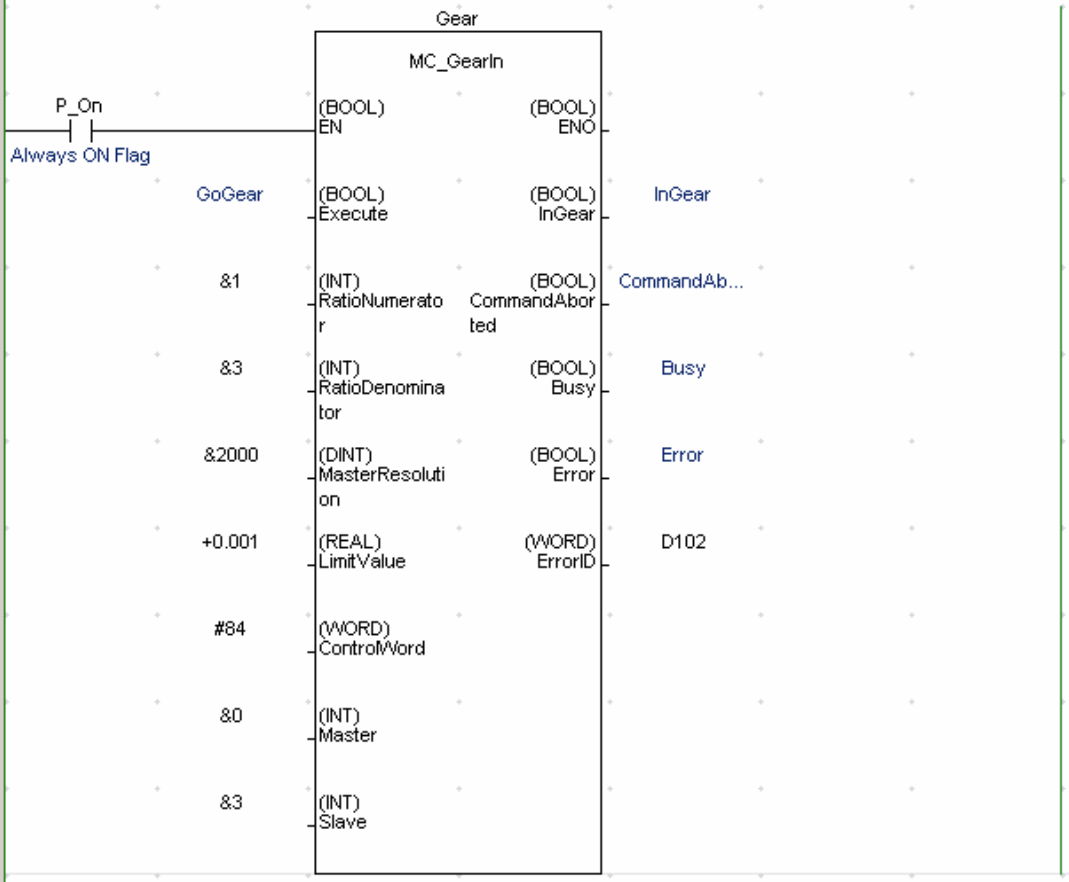
Bit	Value	Description
0	0	Reserved
1	0	Enable operation as slave in electronic gearing
	1	Don't enable operation as slave in electronic gearing
2	0	Send gear ratio parameters
	1	Don't send the gear ratio parameters
3	0	Send the master resolution
	1	Don't send the master resolution
4	0	Use master resolution read from input **MasterResolution**
	1	Master resolution is 0x80000001
5	0	Master position is received via communication channel
	1	Master position is read from 2nd encoder or pulse & direction inputs
6	0	Target update mode 1. Generates new trajectory starting from the actual values of position and speed reference (i.e. don't update the reference values with load/motor position and speed)
	1	Target update mode 0. Generates new trajectory starting from the actual values of load/motor position and speed (i.e. update the reference values with load/motor position and speed)
7	0	The slave acceleration is not limited
	1	Limit the acceleration of the slave drive/motor
8-14	0	Reserved
15	0	The motion commands are sent to one drive/motor
	1	The motion commands are sent to a group of drives/motors.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:

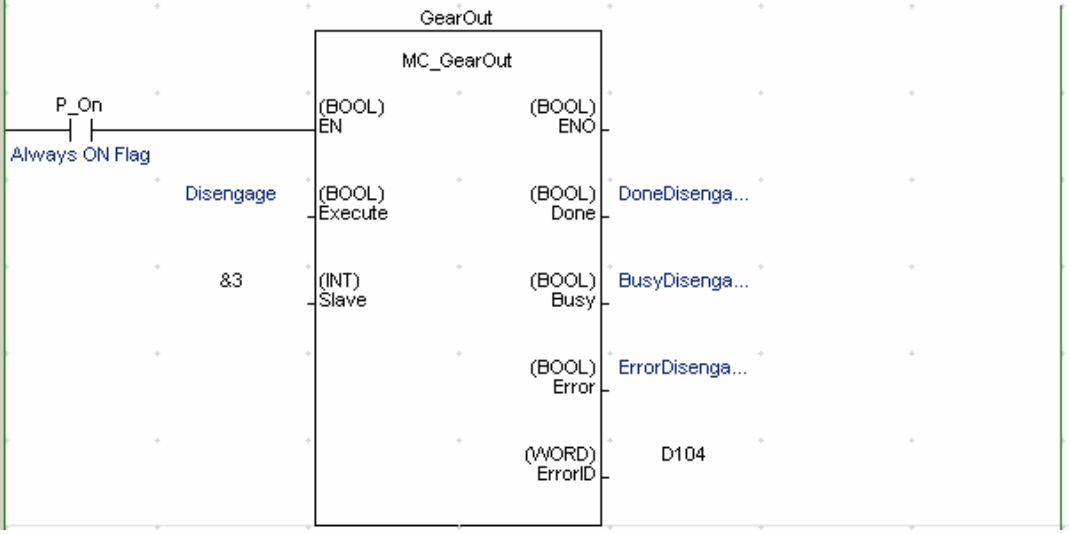
2

2 The function block enables the electronic gearing slave mode for drive motor with AxisID = 3.
The Master AxisID is not required and is set to 0.



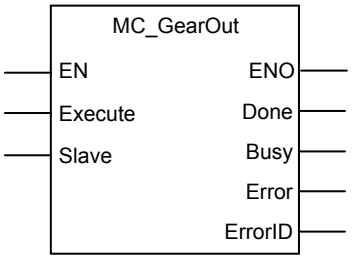
3

4 Disengage the slave
The function block disengages the slave from the master. The slave will keep its velocity.



3.6.17 MC_GearOut

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Start disengaging process at rising edge
	Slave	INT	AxisID of the slave drive/motor
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Disengaging completed.
	Busy	BOOL	The function block is waiting for drive/motor to signal disengage completed
	Error	BOOL	Is set if drive's/motor's error register is received while **!Done**
	ErrorID	WORD	Information about the error occurred

Description: The function block disengages a drive/motor, set in electronic gearing slave mode, from its master.

On detecting a rising edge at the input **Execute** the function block sends the commands for decoupling from the master and sets the Busy output. The **Done** output is set when the disengaging process was successfully executed.

The function block changes the slave state from **SynchronizedMotion** to **ContinuousMotion**, the slave keeping the velocity from the moment when the function block was called.

If the drive's/motor's error register is received, while the output **Done** is not set, the output **Error** is set and its value is passed to **ErrorID**.

All outputs remain set until **Execute** input is reset, but at least for one block call.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

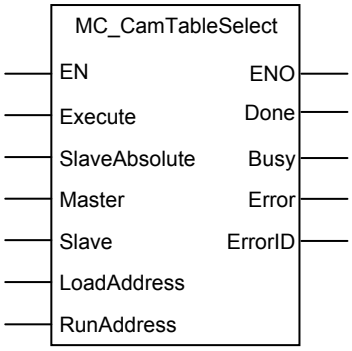
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Remark:

*The function block requires the drive/motor position and speed loops to be closed. During the drive/motor setup select **Position** at Control Mode and in **Advanced** dialogue choose option **Close Position, Speed and Current loop**. After the selection perform the controllers tuning.*

3.6.18 MC_CamTableSelect

Symbol:



Parameters description:

	Parameters	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Send motion commands at rising edge
	SlaveAbsolute	BOOL	Specifies the electronic camming type
	Master	INT	Axis ID of the master drive/motor
	Slave	INT	Axis ID of the slave drive/motor
	LoadAddress	WORD	EEPROM address where the cam table is downloaded
	RunAddress	WORD	RAM address from where the cam table is used
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Commanded position reached
	Busy	BOOL	Signal the function block is waiting for motion complete
	Error	BOOL	Signal if an error has occurred in the execution of the function block.
	ErrorID	WORD	Information about the error occurred

Description: The function block sets the cam table used by the slave drive/motor in electronic camming and the electronic camming type: absolute or relative. In the relative mode (SlaveAbsolute = 0), the output of the cam table is added to the slave actual position. In the absolute mode (SlaveAbsolute = 1), the output of the cam table is the target position to reach.

The cam tables are first downloaded into the EEPROM memory of the drive/motor, then, calling **MC_CamTableSelect**, the selected cam table is copied from the EEPROM memory into the drive/motor RAM memory.

You specify the EEPROM address where the cam table is downloaded through input **LoadAddress**. The RAM address where the cam table is copied is set through input **RunAddress**.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Remark: *The LoadAddress and RunAddress are drive/motor specific.*

If the cam tables are created and/or downloaded with EasyMotion Studio you can find the **first** cam table LoadAddress and RunAddress from **Memory Settings** dialogue. The **Memory Settings** dialogue is opened from **Application General Information** view.

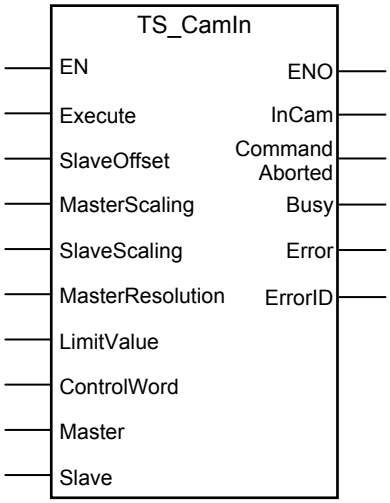
For applications that require several cam tables, their LoadAddress and RunAddress are obtained by adding the cam table length to the addresses of the first cam table.

Remarks:

- *The function must be called before enabling the electronic camming with function block MC_CamIn.*
- *During electronic camming slave mode, only one cam table can be active at time.*

3.6.19 TS_CamIn

Symbol:



Parameter description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	At rising edge the command is sent
	SlaveOffset	DINT	Cam table offset expressed in slave TML position units
	MasterScaling	REAL	CAM table input scaling factor
	SlaveScaling	REAL	CAM table output scaling factor
	MasterResolution	DINT	Master's position sensor resolution expressed in encoder counts
	LimitValue	REAL	Speed limit value expressed in TML speed units
	ControlWord	WORD	Selects the block options
	Master	INT	Axis ID of the master drive/motor
	Slave	INT	Axis ID of the slave drive/motor
Output	ENO	BOOL	Status of function block execution
	InCam	BOOL	Signal the cam is engaged for the first time
	CommandAborted	BOOL	The motion command is aborted by another motion command
	Busy	BOOL	The function block is waiting for the drive/motor to signal camming engaged
	Error	BOOL	Is set if drive's/motor's error register is received while **!InCam**
	ErrorID	WORD	Information about the error occurred

Description: The function block programs a drive/motor to operate as slave in electronic camming. The slave drive/motor executes a cam profile function of the master drive/motor position. The cam profile is defined by a cam table – a set of (X, Y) points, where X is cam table

input i.e. the master position and Y is the cam table output i.e. the corresponding slave position. Between the points the drive/motor performs a linear interpolation.

Remark: *The active cam table is selected with function block MC_CamTableSelect.*

The slave can get the master position in two ways:

1. Via a communication channel (ControlWord.5 = 0), from a drive/motor set as master with function block TS_SetMaster
2. Via an external digital reference of type pulse & direction or quadrature encoder (ControlWord.5 = 1)

When master position is provided via the external digital interface, the slave computes the master position by counting the pulse & direction or quadrature encoder signals. The initial value of the master position is set by default to 0. Use function block TS_WriteLongParameter to change its value by writing the desired value in the TML variable APOS2 (long@0x091C).

Remarks:

1. *The function block requires drive/motor position loop to be closed. During the drive/motor setup select* ***Position*** *at Control Mode and perform the position controller tuning.*
2. *If the application requires switching between synchronized motion (position control) and continuous motion (speed control), i.e. use MC_CamOut, you must also close the speed loop and perform the tuning of the speed controller*.
3. *When the reference is read from second encoder or pulse & direction inputs you don't need to program a drive/motor as master in electronic camming*

Through input **SlaveOffset** you can shift the cam profile versus the master position, by setting an offset for the slave. The cam table input is computed as the master position minus the cam offset. For example, if a cam table is defined between angles 100 to 250 degrees, a cam offset of 50 degrees will make the cam table to execute between master angles 150 and 300 degrees.

The electronic camming can be relative or absolute. In the relative mode, the output of the cam table is added to the slave actual position. In the absolute mode, the output of the cam table is the target position to reach, TML variable TPOS (long@0x02B2). The camming mode, absolute or relative, is selected with function block MC_CamTableSelect.

You can compress/extend the cam table input. Specify through input **MasterScaling** the correction factor by which the cam table input is multiplied. For example, an input correction factor of 2, combined with a cam offset of 180 degrees, will make possible to execute a cam table defined for 360 degrees of the master in the last 180 degrees.

You can also compress/extend the cam table output. Specify through input **SlaveScaling** the correction factor by which the cam table output is multiplied. This feature addresses the applications where the slaves must execute different position commands at each master cycle, all having the same profile defined through a cam table. In this case, the drive/motor is programmed with a unique normalized cam profile and the cam table output is multiplied with the relative position command updated at each master cycle.

The **Master Resolution** represents the number of encoder counts per one revolution of the master motor. The slaves need the master resolution to compute correctly the master position and speed (i.e. position increment). Set ControlWord.4 if master position is not cyclic (e.g. the resolution is equal with the whole 32-bit range of position). In this case the master resolution is set to value 0x80000001.

On detecting a rising edge at the **Execute** input the function block starts sending the motion commands and sets the **Busy** output. The function block sends the motion commands according with the selections made through **ControlWord** input. **Busy** remains set until the drive/motor signals coupling completed, moment when the function block sets the **InCam** output and resets **Busy**.

While the output **InCam** is not set the function checks:

- If other motion function blocks start sending commands to the same drive/motor. In this case the function is aborted and the output **CommandAborted** is set
- If the error register is received from the drive/motor. In this case the function block sets the output **Error**. The value of error register is transferred at the output **ErrorID**.

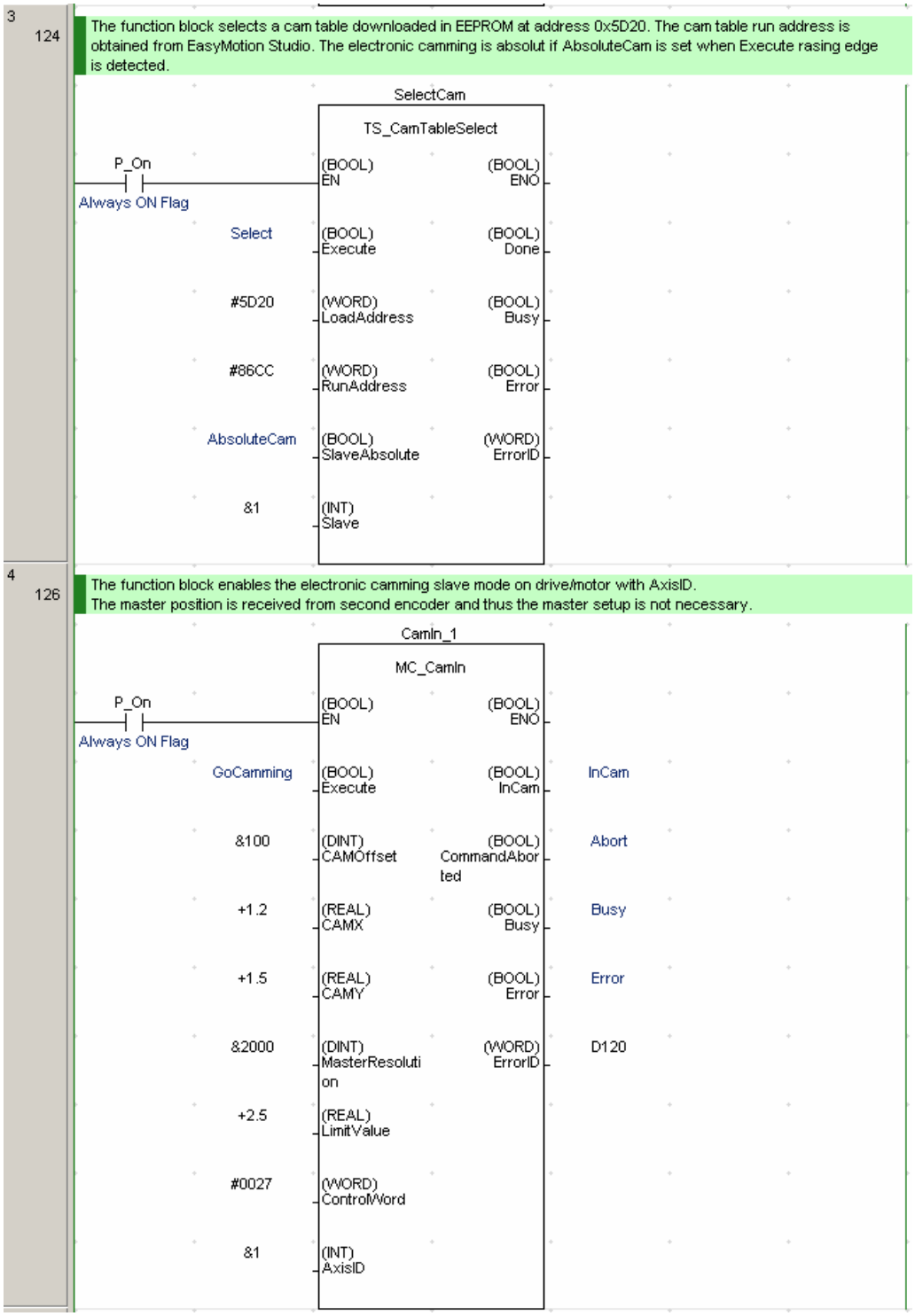
All outputs remain set until **Execute** input is reset, but at least for one block call.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Table 3.19. *ControlWord bits description*

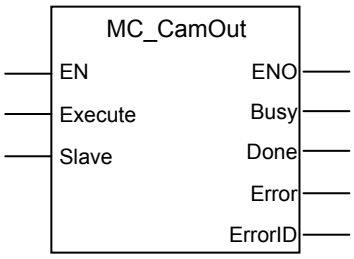
Bit	Value	Description
0	0	Send the cam table input scaling factor
	1	Don't send the cam table input scaling factor
1	0	Send the cam table output scaling factor
	1	Don't send the cam table input scaling factor
2	0	Send the cam table offset
	1	Don't send the cam table offset
3	0	Send the master resolution
	1	Don't send the master resolution
4	0	Use master resolution read from input MasterResolution
	1	Master resolution is 0x80000001
5	0	Master position is received via communication channel
	1	Master position is read from 2nd encoder or pulse & direction inputs
6	0	Target Update Mode 1 (TUM1). Generates new trajectory starting from the actual values of position and speed reference
	1	Target Update Mode 0 (TUM0). Generates new trajectory starting from the actual values of load/motor position and speed
7	0	The acceleration is not limited
	1	Limit the acceleration
8-14	0	Reserved
15	0	The motion commands are sent to one drive/motor
	1	The motion commands are sent to a group of drives/motors.

Example:


3
124
The function block selects a cam table downloaded in EEPROM at address 0x5D20. The cam table run address is obtained from EasyMotion Studio. The electronic camming is absolut if AbsoluteCam is set when Execute rasing edge is detected.
SelectCam
TS_CamTableSelect
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Select
(BOOL) Execute
(BOOL) Done
#5D20
(WORD) LoadAddress
(BOOL) Busy
#86CC
(WORD) RunAddress
(BOOL) Error
AbsoluteCam
(BOOL) SlaveAbsolute
(WORD) ErrorID
&1
(INT) Slave
4
126
The function block enables the electronic camming slave mode on drive/motor with AxisID.
The master position is received from second encoder and thus the master setup is not necessary.
CamIn_1
MC_CamIn
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
GoCamming
(BOOL) Execute
(BOOL) InCam
InCam
&100
(DINT) CAMOffset
(BOOL) CommandAborted
Abort
+1.2
(REAL) CAMX
(BOOL) Busy
Busy
+1.5
(REAL) CAMY
(BOOL) Error
Error
&2000
(DINT) MasterResolution
(WORD) ErrorID
D120
+2.5
(REAL) LimitValue
#0027
(WORD) ControlWord
&1
(INT) AxisID

3.6.20 MC_CamOut

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Start disengaging process at rising edge
	Slave	INT	AxisID of the slave drive/motor
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Disengaging completed.
	Busy	BOOL	The function block is waiting for drive/motor to signal disengage completed
	Error	BOOL	Is set if drive's/motor's error register is received while **!Done**
	ErrorID	WORD	Information about the error occurred

Description: The function block decouples a slave drive/motor set in electronic camming from its master.

On detecting a rising edge at the input **Execute** the function block sends the commands for decoupling from the master and sets the **Busy** output. The **Done** output is set when the disengaging process was successfully executed.

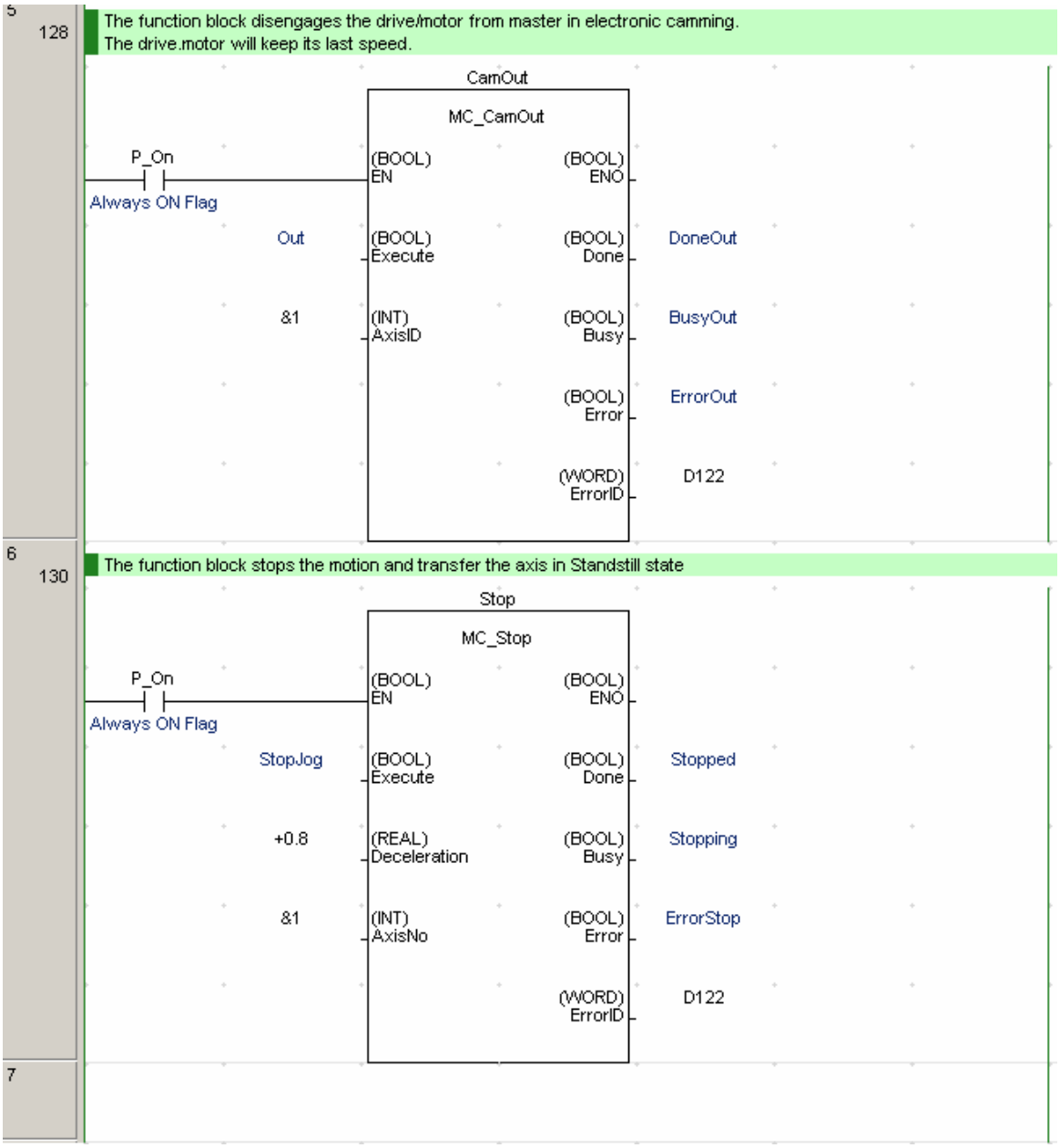
The function block changes the slave state from **SynchronizedMotion** to **ContinuousMotion**, the slave keeping the velocity from the moment when the function block was called.

If the drive's/motor's error register is received, while the output **Done** is not set, the output **Error** is set and its value is passed to **ErrorID**.

All outputs remain set until **Execute** input is reset, but at least for one block call.

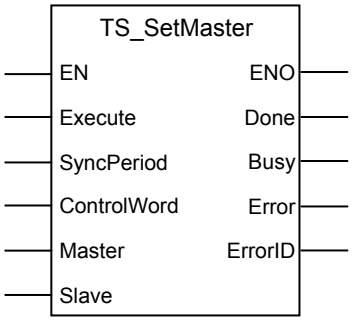
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:


5
128
The function block disengages the drive/motor from master in electronic camming.
The drive.motor will keep its last speed.
CamOut
MC_CamOut
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Out
(BOOL) Execute
(BOOL) Done
DoneOut
&1
(INT) AxisID
(BOOL) Busy
BusyOut
(BOOL) Error
ErrorOut
(WORD) ErrorID
D122
6
130
The function block stops the motion and transfer the axis in Standstill state
Stop
MC_Stop
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
StopJog
(BOOL) Execute
(BOOL) Done
Stopped
+0.8
(REAL) Deceleration
(BOOL) Busy
Stopping
&1
(INT) AxisNo
(BOOL) Error
ErrorStop
(WORD) ErrorID
D122
7

3.6.21 TS_SetMaster

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Execute	BOOL	Send the stop command at rising edge
	SyncPeriod	DINT	Time period for synchronization messages expressed in TML time internal units
	ControlWord	WORD	Selects the block options
	Master	INT	Master drive/axis AxisID
	Slave	INT	Slave drive/motor AxisID
Output	ENO	BOOL	State of function block execution
	Done	BOOL	Master configuration commands sent
	Busy	BOOL	The function block is processing the commands
	Error	BOOL	Not used
	ErrorID	WORD	Not used

Description: The function block programs a drive/motor as master in electronic gearing or camming. The master drive/motor AxisID is read from **Master** input, while the slave AxisID is read from the **Slave** input.

The master operation is enabled when the function block is called with ControlWord1 = 0 and disabled when the function block is called with ControlWord.0 = 1. In both cases, these operations have no effect on the motion executed by the master.

Once at each slow loop sampling time interval, the master sends either its load position (ControlWord.4 = 0) or its position reference (ControlWord.4=1) to the axis or the group of axes having **Slave** AxisID/GroupID.

When ControlWord.15=0 the **Slave** value is the AxisID of one slave. When ControlWord.15 =1 the value is interpreted as a group ID, i.e. the group of slaves to which the master should send its data.

Remark: *You need to specify the Axis ID or the Group ID where master sends its position only the first time (after power on) when a drive is set as master. If the master mode is later on disabled (function block called with ControlWord.0 = 1), then enabled (function block called with*

ControlWord.1 = 0) again, there is no need to set again the Axis ID or the Group ID, as long as they remain unchanged. In this case, set the ControlWord.2 =1.

When ControlWord.5 = 1 you can change the synchronization procedure between the master and the slave axes. The synchronization procedure is enabled when ControlWord.6 = 0, the time interval between synchronization messages is read from **SyncPeriod** input, or disabled when ControlWord.6 = 1. Recommended starting value is 20ms. When synchronization procedure is active, the execution of the control loops on the slaves is synchronized with those of the master within a 10µs time interval.

In electronic gearing if the master activation is done AFTER the slaves are set in electronic gearing mode and its position is different from zero, set ControlWord.3=1 before the function block is called. This determines the master to send its (actual or target/reference) position to the slaves, performing a necessary initialization. In electronic camming the ControlWord.3 must be set thus disabling the initialization of the slaves.

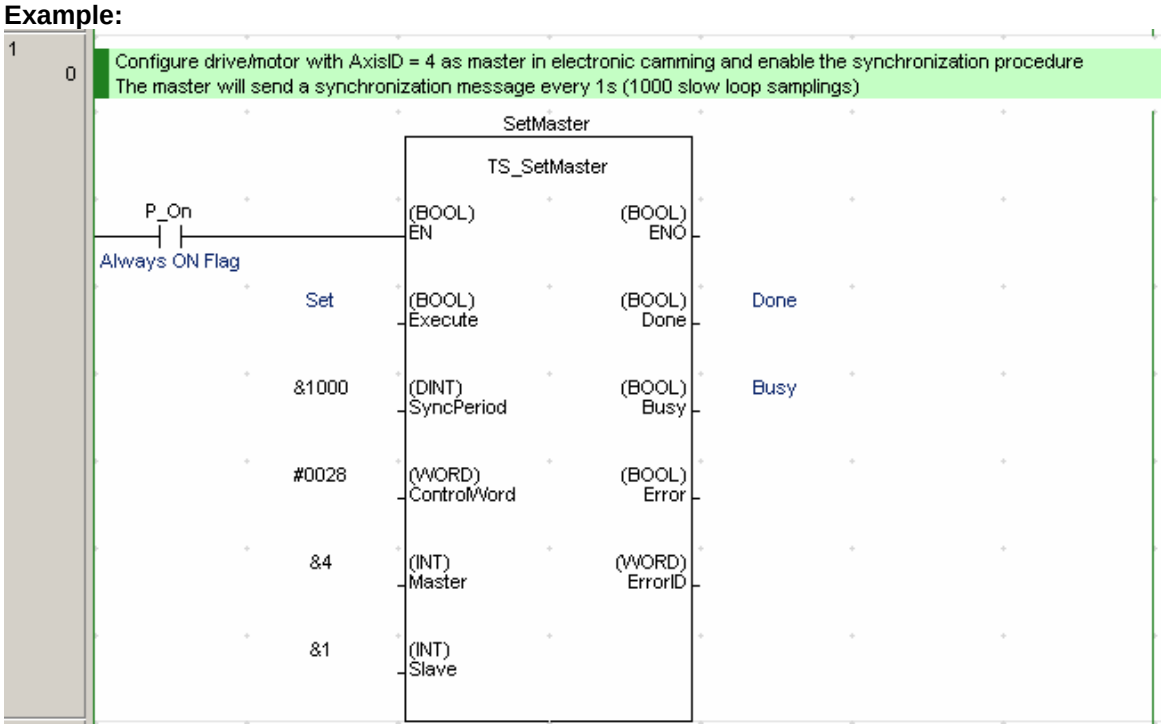
On detecting a rising edge at the input **Execute** the function block sends the configuration commands to the drive/motor with AxisID read from **Master** input. The function block sends the motion commands according with the selections made through **ControlWord** input. The **Done** output is set when the last message is sent. Outputs **Error** and **ErrorID** have no associated functionality.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

***Table 3.20** ControlWord bits description*

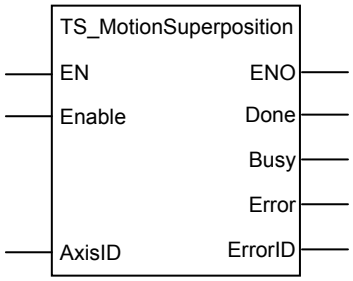
Bit	Value	Description
0	0	Nothing
	1	Disable operation as master in electronic gearing/camming
1	0	Enable the operation as master in electronic gearing/camming
	1	Don't enable the operation as master in electronic gearing/camming
2	0	Send the slave axis/group ID read from Slave input
	1	Don't send the slave axis/group ID
3	0	Initialize the slaves with actual/reference position in electronic gearing
	1	Don't initialize the slaves with actual/reference position in electronic gearing
4	0	The master sends its actual position to the slaves
	1	The master sends its reference position to the slaves
5	0	Skip the synchronization procedure setup
	1	Change the synchronization procedure according with ControlWord.6
6	0	Enable the synchronization procedure with time interval read from SyncPeriod
	1	Disable the synchronization procedure
7-14	0	Reserved
15	0	The master send its position to the drive/motor with axis ID read from Slave input
	1	The master send its position to drives/motors members of group ID read from Slave input

Example:


1
0
Configure drive/motor with AxisID = 4 as master in electronic camming and enable the synchronization procedure
The master will send a synchronization message every 1s (1000 slow loop samplings)
SetMaster
TS_SetMaster
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Set
(BOOL) Execute
(BOOL) Done
Done
&1000
(DINT) SyncPeriod
(BOOL) Busy
Busy
#0028
(WORD) ControlWord
(BOOL) Error
&4
(INT) Master
(WORD) ErrorID
&1
(INT) Slave

3.6.22 TS_MotionSuperposition

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	Enable	BOOL	Enable = 1 enables motion superposition, 0 disables motion superposition
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	Done	BOOL	Done = 1 motion superposition enabled Done = 0 motion superposition disabled
	Busy	BOOL	The function is processing the command
	Error	BOOL	Not used
	ErrorID	WORD	Not used
	ENO	BOOL	Status of function block execution

Description: The function block enables/disables the superposition of the electronic gearing slave mode with a second motion mode. When this superposed mode activated, the position reference is computed as the sum of the position references for each of the 2 superposed motions.

You may enable the superposed mode at any moment, independently of the activation/deactivation of the electronic gearing slave. If the superposed mode is activated during an electronic gearing motion, any subsequent motion mode change is treated as a second move to be superposed over the basic electronic gearing move, instead of replacing it. If the superposed mode is activated during another motion mode, a second electronic gearing mode will start using the motion parameters previously set. This move is superposed over the first one. After the first move ends, any other subsequent motion will be added to the electronic gearing.

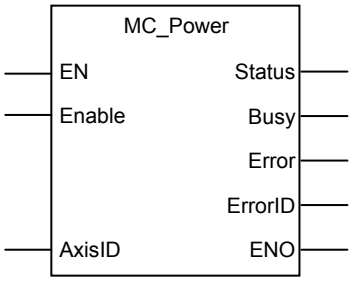
When you disable the superposed mode, the electronic gearing slave move is stopped and the drive/motor executes only the other motion. If you want to remain in the electronic gearing slave mode, set first the electronic gearing slave move and then disable the superposed mode.

On detecting a rising edge at **Enable** input the function block enables the motion superposition on the drive with **AxisID**. The motion superposition is disabled when the **Enable** input is reset.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

3.6.23 MC_Power

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	Enable	BOOL	Enable = 1 enables power stage, 0 disables power stage
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function block execution
	Status	BOOL	Effective power stage state
	Busy	BOOL	Signal the function is waiting for power stage status
	Error	BOOL	Is set when the power stage of the motor is disabled while Enable input is set

Description: The function block enables/disables (AxisOn/AxisOff) the power stage of the drive/motor with **AxisID**.

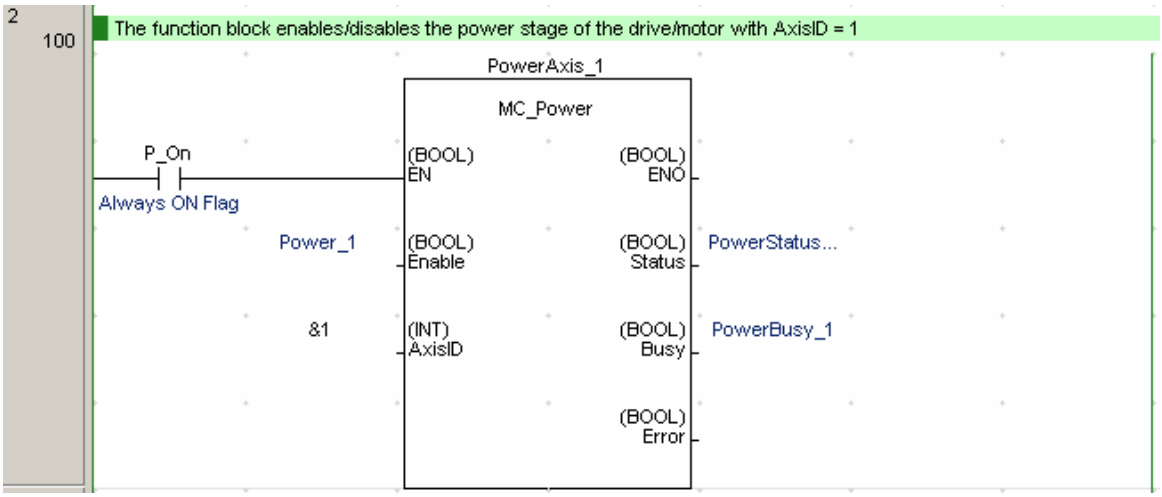
On detecting a rising edge at **Enable** input the function block enables the power stage. If no error occurs the axis is transferred from **Disable** state to **Standstill**. If the drive/motor responds with an error message the axis is transferred from **Disable** to **Errorstop**. The power stage of the drive/motor is disabled when the **Enable** input is reset and the axis is transferred to **Disable** state.

The output **Status** reflects the effective power stage state. If power fails, i.e. during operation the input **Enable** is set and the drive/motor power stage is disabled, the axis is transferred to **Errorstop** state and the **Error** output is set.

Remark: *The function TS_AxisState **must** be executed prior MC_Power call, in order to initialize the automatic messages feature of the drive/motor from the selected axis.*

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:



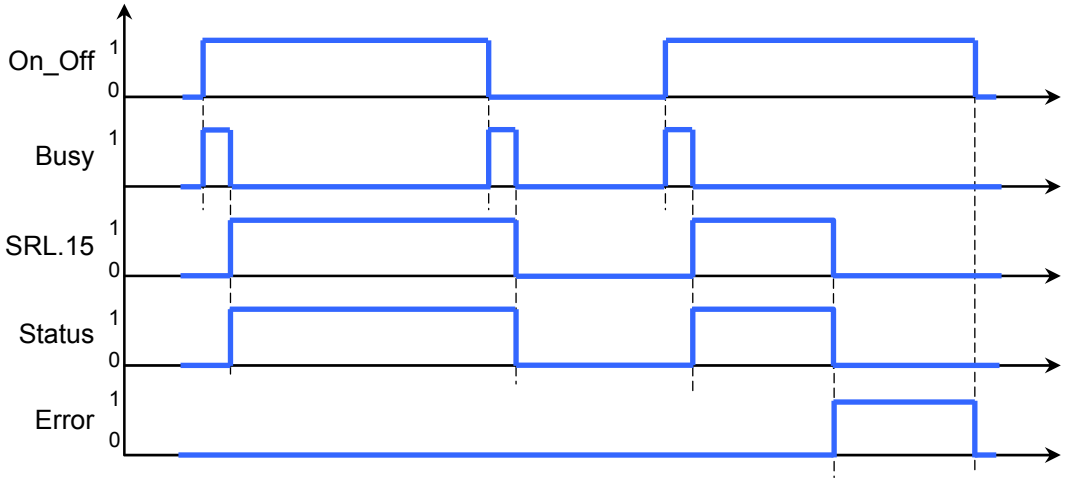
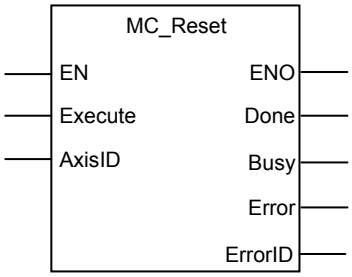


Figure 3.5 *Time diagram for MC_Power.*

3.6.24 MC_Reset

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Reset the axis at the rising edge
	AxisID	INT	Axis ID of the drive/motor where the commands are sent
Output	ENO	BOOL	Status of function execution
	Done	BOOL	The drive/motor reached StandStill state
	Busy	BOOL	The function block is waiting for drive/motor power stage to be re-enabled
	Error	BOOL	Signal if an error has occurred in the execution of the function block
	ErrorID	WORD	Information about the error occurred

Description: The function block makes the transition from **ErrorStop** to **Standstill** by resetting drive/motor FAULT status. The function can be called only from **ErrorStop** state.

A drive/motor enters in the FAULT status, when an error occurs. In the FAULT status:

- The drive/motor is in AXISOFF with the control loops and the power stage deactivated
- The TML program execution is stopped and all the TML functions called are cancelled
- The error register MER (uint@0x08FC) shows the type of errors detected and the status register SRH.15 signals the fault condition
- Ready and error outputs (if present) are set to the not ready level, respectively to the error active level. When available, ready greed led is turned off and error red led is turned on.

Remark: *The following conditions signaled in MER do not set the axis in* ***ErrorStop****:*

- *Command error*
- *Negative limit switch input on activate level*
- *Positive limit switch input on activate level*
- *Position wraparound*
- *Serial and CAN bus communication errors*

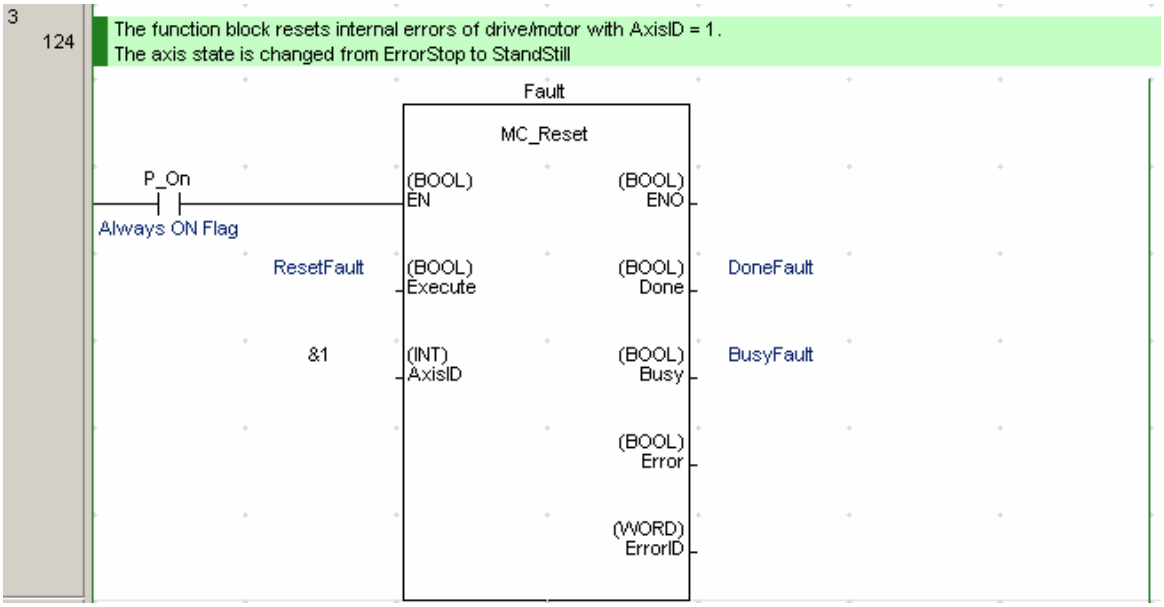
On detecting a rising edge at the **Execute** input the function block sends the reset commands and sets the **Busy** output. **Busy** remains set until the drive/motor power stage is re-enabled,

moment when the function block sets the **Done** output and resets the **Busy.** The axis is transferred to **Standstill** state.

If the power stage is not enabled the axis remains in **ErrorStop** and the **Error** output is set. The value 0x0001 is transferred to **ErrorID** output.

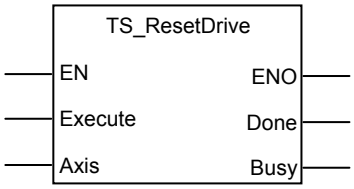
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:



3.6.25 TS_ResetDrive

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable input
	Execute	BOOL	Send stop command at rising edge
	AxisID	INT	AxisID of the drive/motor where the command is sent
Output	ENO	BOOL	Enable output
	Done	BOOL	Signal drive velocity zero.
	Busy	BOOL	The function process the commands

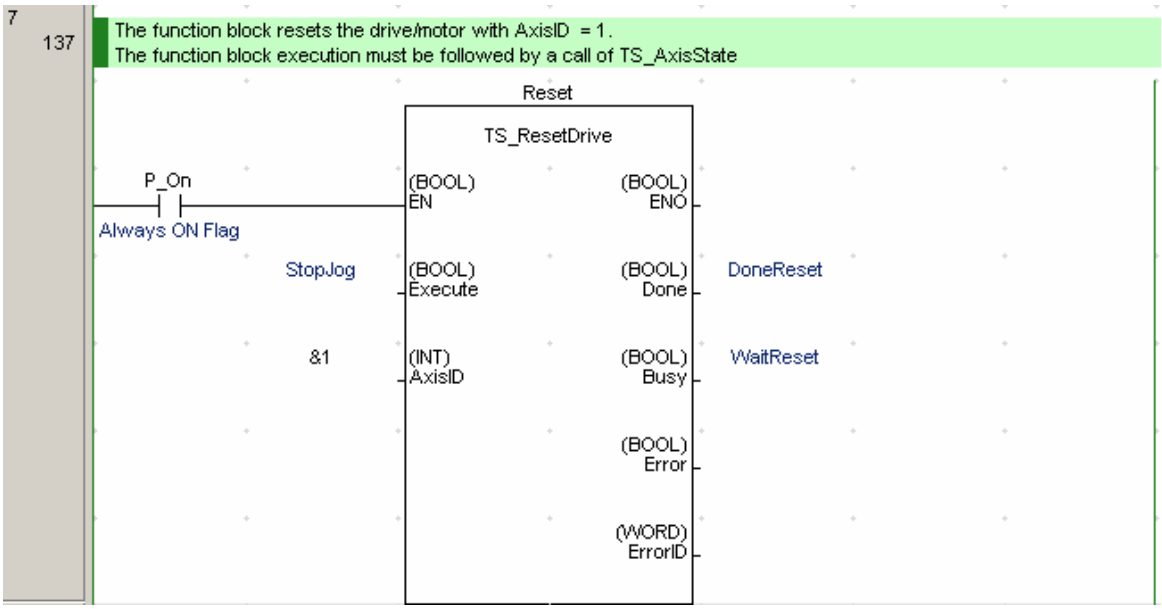
Description: Function block makes the transition from **ErrorStop** to **Disable** by resetting the drive/motor with **AxisID**. After a reset operation the drive/motor requires a time period (typically 30 ms) for initialization during which the communication is not available.

On detecting a rising edge at the **Execute** input the function block sends the reset commands and sets the **Busy** output. When,. The **Busy** remains set until the timer has elapsed, after 30ms, moment when the function block sets the **Done** output and resets the **Busy.** The axis is transferred to **Disable** state.

Remarks:

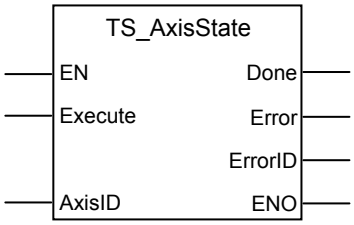
1. *The function block reinitializes the drive/motor with the setup data found in EEPROM. If you changed the values of drive/motor parameters during normal operation and want to use the same values after the reset call TS_SaveParameters prior function block TS_ResetDrive.*
2. *After the reset call function TS_AxisState to re-enable the drive/motor to send automatically messages with its status.*

Example:


7
137
The function block resets the drive/motor with AxisID = 1.
The function block execution must be followed by a call of TS_AxisState
Reset
TS_ResetDrive
P_On
Always ON Flag
(BOOL)
EN
(BOOL)
ENO
StopJog
(BOOL)
Execute
(BOOL)
Done
DoneReset
&1
(INT)
AxisID
(BOOL)
Busy
WaitReset
(BOOL)
Error
(WORD)
ErrorID

3.6.26 TS_AxisState

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable input
	Execute	BOOL	Send stop command at rising edge
	AxisID	STRUCT	Axis information
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Signal drive velocity zero.
	Error	BOOL	Is set when the drive motor from the drive
	ErrorID	WORD	Information about the error occurred

Description: The function block enables a drive/motor to send automatically messages with its status. The configuration messages are sent to the drive/motor having the **AxisID**. Also the function block monitors all messages sent by the drive/motor towards the PLC and stores the information received in the memory reserved.

Remark: *Due to its functionality the function block must be called individually for each axis defined in the application.*

If the drive/motor is powered when the PLC program starts the function block will initialize it automatically. In case the drive/motor is powered after the PLC program starts or it was reset during the operation it is required to call the function block.

After the last initialization command is sent to the drive/motor the **Done** output is set and the function starts to monitor the messages received from the drive.

If the error register is received the **Error** output is set and its value it's value is passed to the **ErrorID** output. The **Error** output is automatically reset when the PLC receives a message from the drive/motor that signals no fault state.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

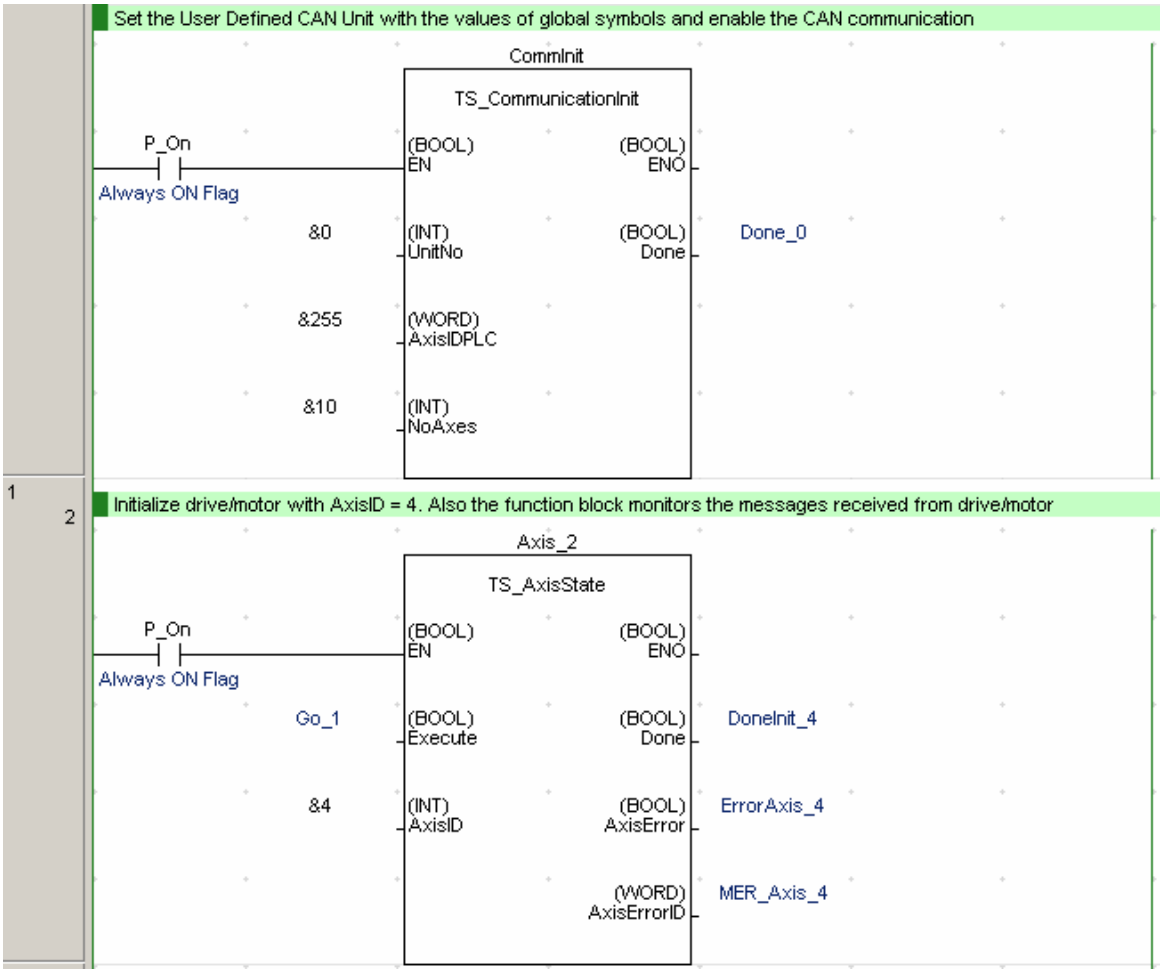
The function block will change the axis state to **ErrorStop** if the drive/motor signals any of the following errors:

- undervoltage
- overvoltage
- drive overtemperature
- motor overtemperature

- I2T
- Overcurrent
- Control error
- Short-circuit

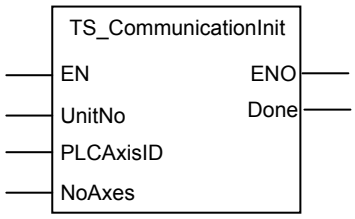
Remark: *Use function MC_Reset to restore the state* ***StandStill****.*

Example:



3.6.27 TS_CommunicationInit

Symbol:



Parameter description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	UnitNo	INT	The unit number identical with the one set through rotative switch
	PLCAxisID	INT	The axis ID assigned to the PLC
	NoAxes	INT	The number of axes used in the application
Output	ENO	BOOL	Status of the function execution
	Done	BOOL	Signals the initialization ended

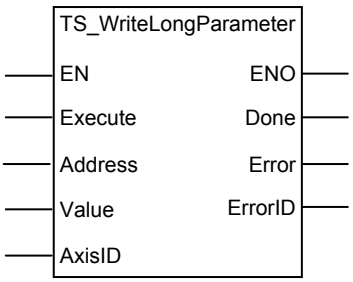
Description: When the PLC program starts the function configures automatically the User Defined CAN Unit and enables the CAN communication. At the end of the initialization the output **Done** is set.

The function must be called before any function block call that uses the User Defined CAN Unit.

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

3.6.28 TS_WriteLongParameter

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Execute	BOOL	Write long parameter at the rising edge
	Address	WORD	Data memory address where the value will be written
	Value	DINT	Value to be written
	AxisID	INT	Axis ID of the drive where the write will be performed
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Is set after the write data message is sent
	Error	BOOL	Is set when the parameter address is out of range
	ErrorID	WORD	Information about the error occurred

Description: The function block sends a "write long/ulong data" message to the drive/motor with **AxisID**. When the drive/motor receives the message it will write the **Value** in the data memory location **Address**.

The TML uses the following data types:

- int – 16-bit signed integer
- uint – 16-bit unsigned integer
- fixed – 32-bit fixed-point data with the 16MSB for the integer part and the 16LSB for the factionary part
- **long** – 32-bit signed integer
- **ulong** –32-bit unsigned integer

The data type uint or ulong are reserved for the TML predefined data. The user-defined variables are always signed. Hence you may declare them of type: int, fixed or long.

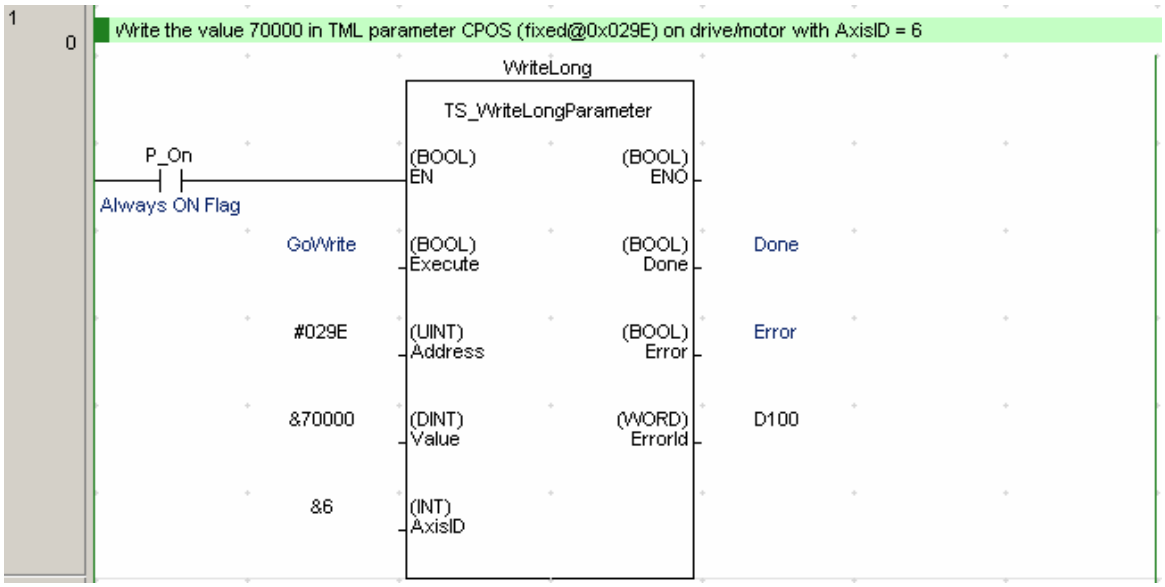
Each TML data (parameters, variables or registers) has an associated address. This represents the address of the data memory location where the TML data exists. Address ranges for TML data are from 0x0200 to 0x03AF and from 0x0800 to 0x09FF. For user-defined variables the address range is between 0x03B0 and 0x03FF.

On detecting a rising edge at the **Execute** input, the function block sends the "write data" message. The **Done** output is set when the function block enables the user defined CAN unit to send the message. The output **Done** remains set until **Execute** input is reset.

If the value supplied at the **Address** input is not in the above-specified ranges then the **Error** output is set and the **ErrorID** output will have the value 0x0001.

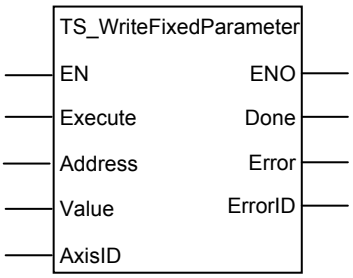
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:



3.6.29 TS_WriteFixedParameter

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Write fixed parameter with the rising edge
	Address	WORD	Data memory address where the value will be written
	Value	REAL	Value to be written
	AxisID	INT	Axis ID of the drive where the write will be performed
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Is set when the write data message is sent
	Error	BOOL	Is set when the parameter address is out of range
	ErrorID	WORD	Information about the error occurred

Description: The function block sends a "write fixed data" message to the drive/motor with **AxisID**. When the drive/motor receives the message it will write the **Value** (converted to 32-bit fixed-point) in the data memory location **Address**.

The TML uses the following data types:

- int – 16-bit signed integer
- uint – 16-bit unsigned integer
- **fixed** – 32-bit fixed-point data with the 16MSB for the integer part and the 16LSB for the factionary part
- long – 32-bit signed integer
- ulong –32-bit unsigned integer

The data type uint or ulong are reserved for the TML predefined data. The user-defined variables are always signed. Hence you may declare them of type: int, fixed or long.

Each TML data (parameters, variables or registers) has an associated address. This represents the address of the data memory location where the TML data exists. Address ranges for TML data are from 0x0200 to 0x03AF and from 0x0800 to 0x09FF. For user-defined variables the address range is between 0x03B0 and 0x03FF.

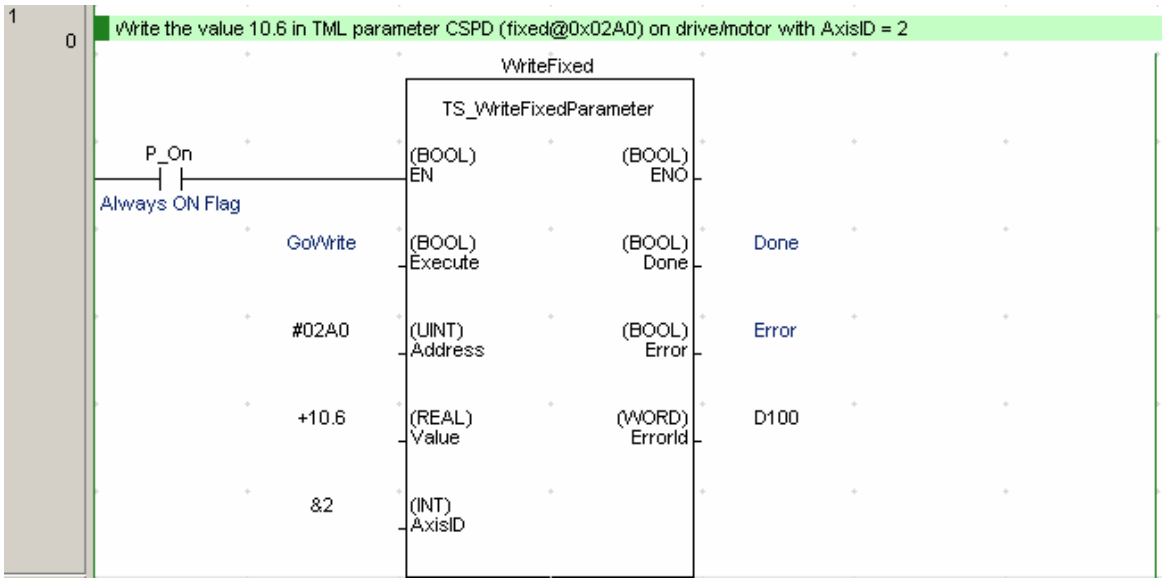
On detecting a rising edge at the **Execute** input, the function block converts the real value read from **Value** input to 32-bit fixed-point data with the 16MSB for the integer part and the 16LSB for

the factionary part. The **Done** output is set when the function block enables the user defined CAN unit to send the message. The output **Done** remains set until **Execute** input is reset.

If the value supplied at the **Address** input is not in the above-specified ranges then the **Error** output is set and the **ErrorID** output will have the value 0x0001.

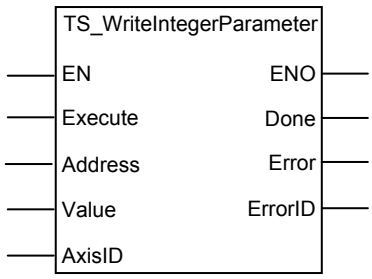
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:



3.6.30 TS_WriteIntegerParameter

Symbol:



Parameters description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Execute	BOOL	Write the parameter with the rising edge
	Address	WORD	Data memory address where the value will be written
	Value	INT	Value to be written at the specified address
	AxisID	INT	Axis ID of the drive where the write will be performed
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Is set when the write data message is sent
	Error	BOOL	Is set when the parameter address is out of range
	ErrorID	WORD	Information about the error occurred

Description: The function block sends a “write integer data” message to the drive/motor with **AxisID**. When the drive/motor receives the message it will write the **Value** in the data memory location **Address**.

The TML uses the following data types:

- **int** – 16-bit signed integer
- uint – 16-bit unsigned integer
- fixed – 32-bit fixed-point data with the 16MSB for the integer part and the 16LSB for the factionary part
- long – 32-bit signed integer
- ulong –32-bit unsigned integer

The data type uint or ulong are reserved for the TML predefined data. The user-defined variables are always signed. Hence you may declare them of type: int, fixed or long.

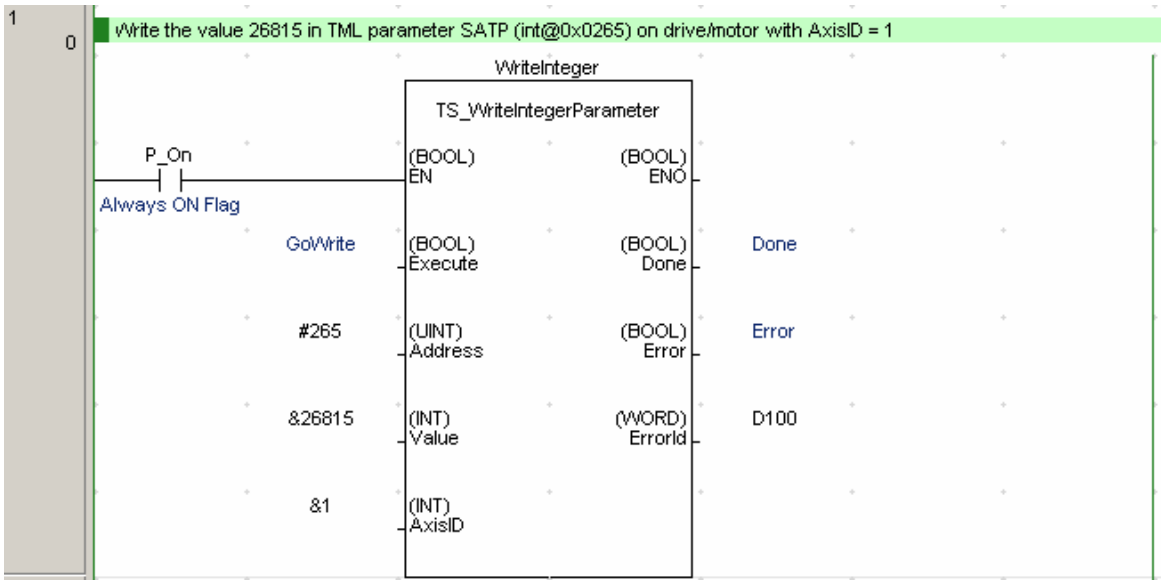
Each TML data (parameters, variables or registers) has an associated address. This represents the address of the data memory location where the TML data exists. Address ranges for TML data are from 0x0200 to 0x03AF and from 0x0800 to 0x09FF. For user-defined variables the address range is between 0x03B0 and 0x03FF.

On detecting a rising edge at the **Execute** input, the function block sends the “write integer data” The **Done** output is set when the function block enables the user defined CAN unit to send the message. The output **Done** remains set until **Execute** input is reset.

If the value read from the **Address** input is not in the above-specified ranges then the **Error** output is set and the **ErrorID** output will have the value 0x0001.

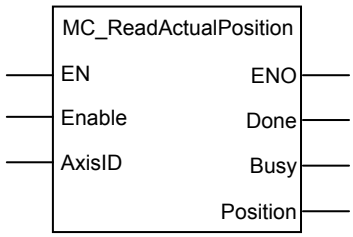
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:



3.6.31 MC_ReadActualPosition

Symbol:



Parameter description:

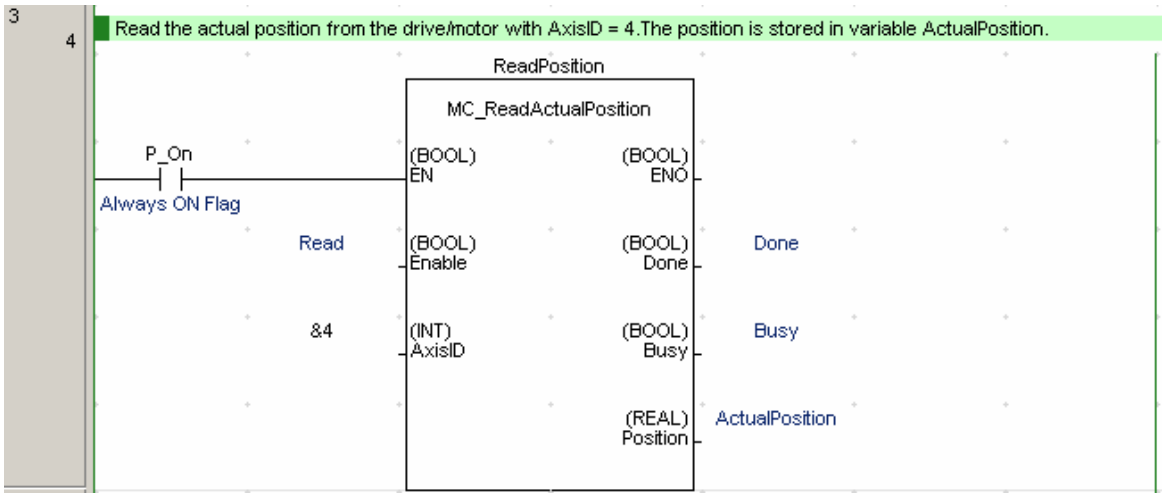
	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Enable	BOOL	Request the value of actual position at the rising edge
	AxisID	INT	AxisID of the drive/motor where the request is sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	The actual position read successfully
	Busy	BOOL	The function block is waiting to receive the value of actual position
	Position	REAL	Actual position expressed in TML position units

Description: The function block requests the value of the actual position from the drive/motor with **AxisID**.

On detecting a rising edge at the **Enable** input, the function block sends the request message and sets the **Busy** output. **Busy** remains set until the requested value is available, moment when, the function block sets the **Done** output and resets **Busy**. The value received is converted to real and transferred to output **Position**. The output **Done** remains set until **Enable** input is reset.

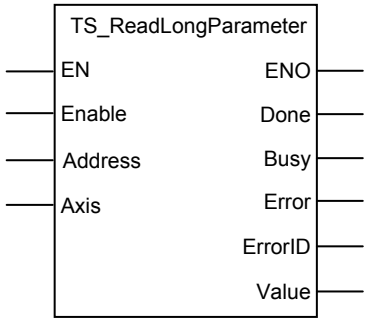
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:


3
4
Read the actual position from the drive/motor with AxisID = 4.The position is stored in variable ActualPosition.
ReadPosition
MC_ReadActualPosition
P_On
Always ON Flag
(BOOL)
EN
(BOOL)
ENO
Read
(BOOL)
Enable
(BOOL)
Done
Done
&4
(INT)
AxisID
(BOOL)
Busy
Busy
(REAL)
Position
ActualPosition

3.6.32 TS_ReadLongParameter

Symbol:



Parameter descriptions:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Enable	BOOL	Request the parameter at the rising edge
	Address	WORD	Data memory address from where the value will be read
	AxisID	INT	AxisID of the drive/motor where the request is sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Is set when the requested value is available
	Busy	BOOL	The function block is waiting to receive the requested value
	Error	BOOL	Is set when the parameter address is out of range
	ErrorID	WORD	Information about the error occurred
	Value	DINT	The value returned by the drive/motor

Description: The function block requests the value of the TML variable/parameter with data memory **Address**. The TML data is of type long/ulong – 32-bit signed/unsigned integer. The request is sent to the drive/motor with **AxisID**.

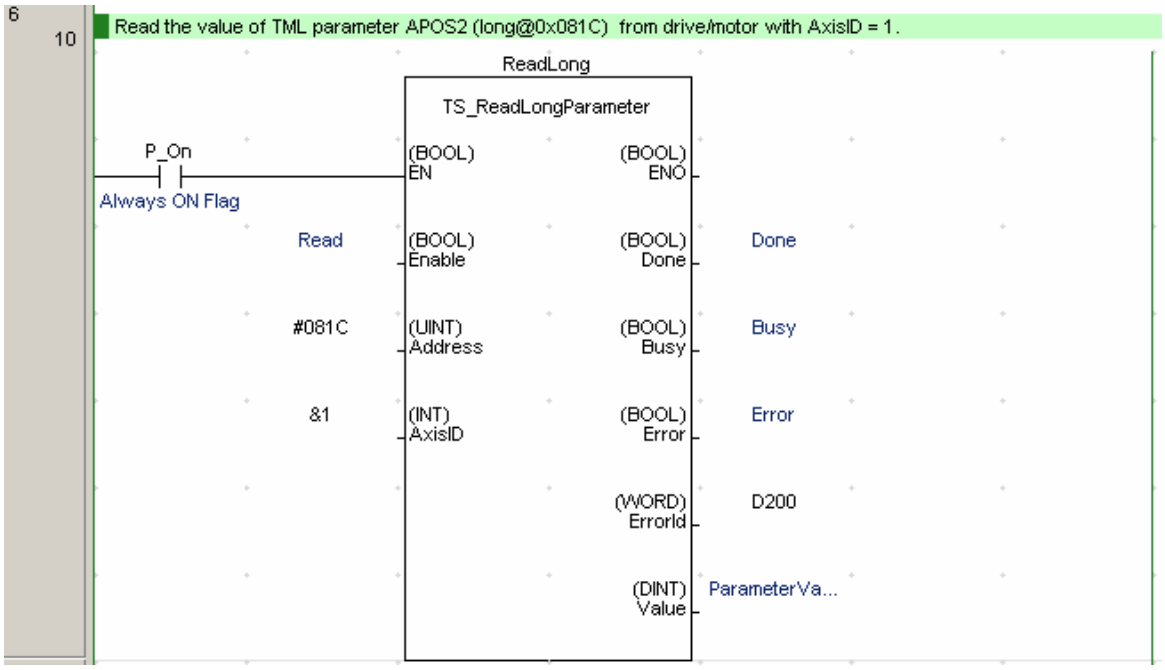
Each TML data (parameters, variables or registers) has an associated address. This represents the address of the data memory location where the TML data exists. Address ranges for TML data are from 0x0200 to 0x03AF and from 0x0800 to 0x09FF. For user-defined variables the address range is between 0x03B0 and 0x03FF.

On detecting a rising edge at the **Enable** input, the function block sends the request message and sets the **Busy** output. **Busy** remains set until the requested value is available, moment when, the function block sets the **Done** output and resets **Busy**. The value received is transferred to output **Value**. The output **Done** remains set until **Enable** input is reset.

If the value read from the **Address** input is not in the above-specified ranges then the **Error** output is set and the **ErrorID** output will have the value 0x0001.

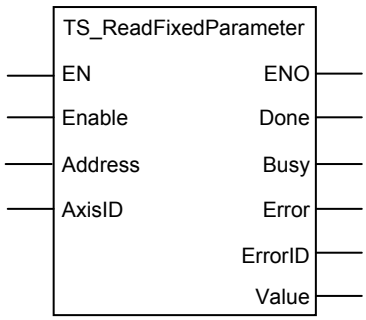
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:


6
10
Read the value of TML parameter APOS2 (long@0x081C) from drive/motor with AxisID = 1.
ReadLong
TS_ReadLongParameter
P_On
Always ON Flag
(BOOL)
EN
(BOOL)
ENO
Read
(BOOL)
Enable
(BOOL)
Done
Done
#081C
(UINT)
Address
(BOOL)
Busy
Busy
&1
(INT)
AxisID
(BOOL)
Error
Error
(WORD)
ErrorId
D200
(DINT)
Value
ParameterVa...

3.6.33 TS_ReadFixedParameter

Symbol:



Parameter descriptions:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Enable	BOOL	Request parameter continuously at the rising edge
	Address	WORD	Data memory address from where the value will be read
	AxisID	INT	AxisID of the drive/motor where the request is sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Is set when the requested value is available
	Busy	BOOL	The function block is waiting to receive the requested value
	Error	BOOL	Is set when the parameter address is out of range
	ErrorID	WORD	Information about the error occurred
	Value	REAL	The value returned by the drive/motor

Description: The function block requests the value of the TML variable/parameter with data memory **Address**. The TML data is of type fixed – 32-bit fixed-point data with the 16MSB for the integer part and the 16LSB for the factionary part. The request is sent to the drive/motor with **AxisID**.

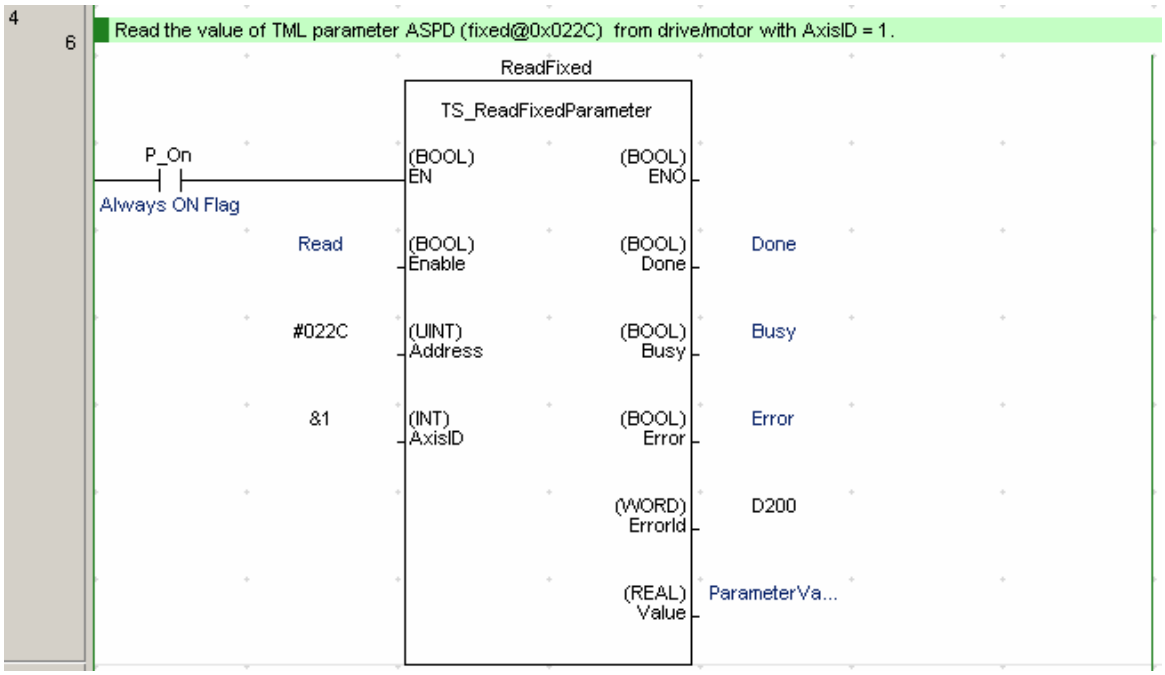
Each TML data (parameters, variables or registers) has an associated address. This represents the address of the data memory location where the TML data exists. Address ranges for TML data are from 0x0200 to 0x03AF and from 0x0800 to 0x09FF. For user-defined variables the address range is between 0x03B0 and 0x03FF.

On detecting a rising edge at the **Enable** input, the function block sends the request message and sets the **Busy** output. **Busy** remains set until the requested value is available, moment when, the function block sets the **Done** output and resets **Busy**. The value received is converted to real and transferred to output **Value**. The output **Done** remains set until **Enable** input is reset.

If the value read from the **Address** input is not in the above-specified ranges then the **Error** output is set and the **ErrorID** output will have the value 0x0001.

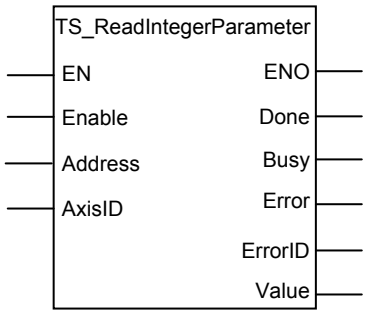
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:


4
6
Read the value of TML parameter ASPD (fixed@0x022C) from drive/motor with AxisID = 1.
ReadFixed
TS_ReadFixedParameter
P_On
Always ON Flag
(BOOL)
EN
(BOOL)
ENO
Read
(BOOL)
Enable
(BOOL)
Done
Done
#022C
(UINT)
Address
(BOOL)
Busy
Busy
&1
(INT)
AxisID
(BOOL)
Error
Error
(WORD)
ErrorId
D200
(REAL)
Value
ParameterVa...

3.6.34 FB TS_ReadIntegerParameter

Symbol:



Parameters descriptions:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Enable	BOOL	Request parameter continuously at the rising edge
	Address	WORD	Data memory address from where the value will be read
	AxisID	INT	AxisID of the drive/motor where the request is sent
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	Is set when the requested value is available
	Busy	BOOL	The function block is waiting to receive the requested value
	Error	BOOL	Is set when the parameter address is out of range
	ErrorID	WORD	Information about the error occurred
	Value	INT	The value returned by the drive/motor

Description: The function block requests the value of the TML variable/parameter with data memory **Address**. The TML data is of type int/uint – 16-bit signed/unsigned integer. The request is sent to the drive/motor with **AxisID**.

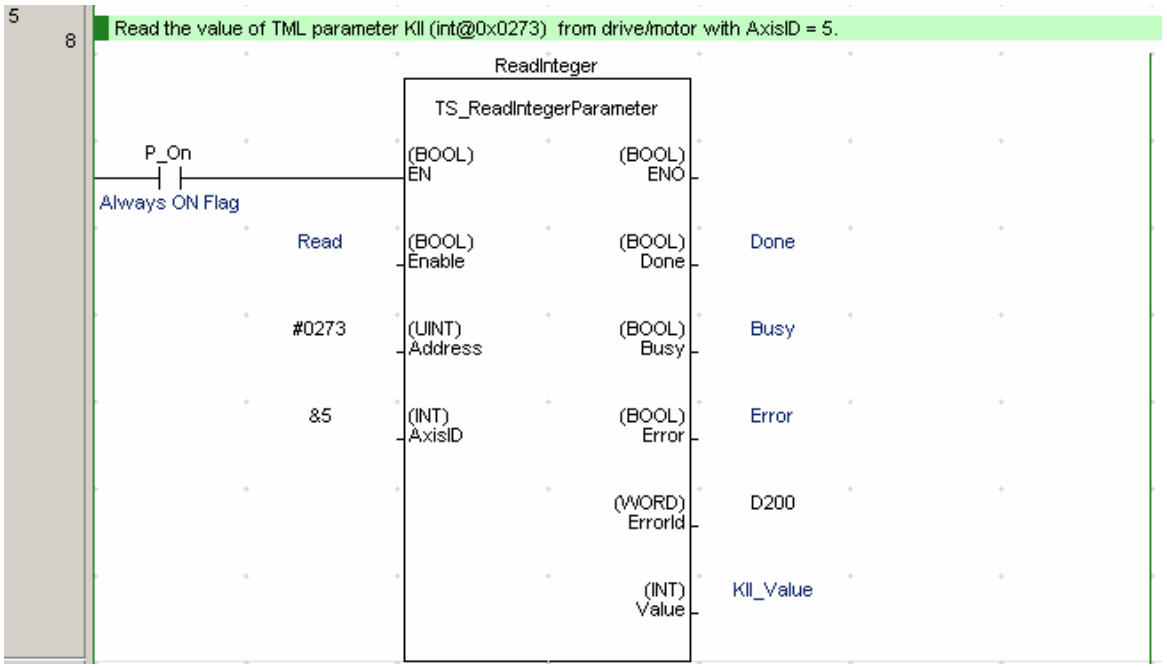
Each TML data (parameters, variables or registers) has an associated address. This represents the address of the data memory location where the TML data exists. Address ranges for TML data are from 0x0200 to 0x03AF and from 0x0800 to 0x09FF. For user-defined variables the address range is between 0x03B0 and 0x03FF.

On detecting a rising edge at the **Enable** input, the function block sends the request message and sets the **Busy** output. **Busy** remains set until the requested value is available, moment when, the function block sets the **Done** output and resets **Busy**. The value received is transferred to output **Value**. The output **Done** remain set until **Enable** input is reset.

If the value read from the **Address** input is not in the above-specified ranges then the **Error** output is set and the **ErrorID** output will have the value 0x0001.

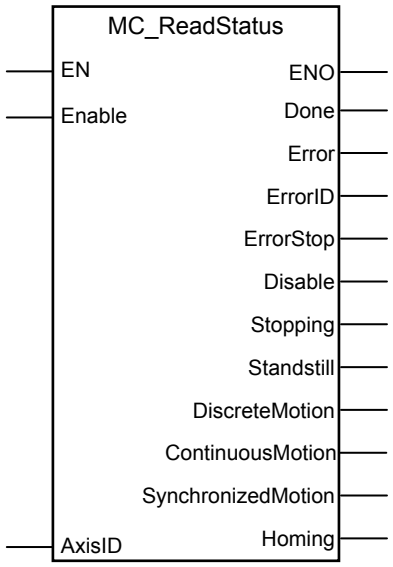
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:


5
8
Read the value of TML parameter KII (int@0x0273) from drive/motor with AxisID = 5.
ReadInteger
TS_ReadIntegerParameter
P_On
Always ON Flag
(BOOL) EN
(BOOL) ENO
Read
(BOOL) Enable
(BOOL) Done
Done
#0273
(UINT) Address
(BOOL) Busy
Busy
&5
(INT) AxisID
(BOOL) Error
Error
(WORD) ErrorId
D200
(INT) Value
KII_Value

3.6.35 FB MC_ReadStatus

Symbol:



Parameter description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Enable	BOOL	Return the axis state while enabled
	AxisID	INT	AxisID of the drive/motor which status is monitored
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	The status read successfully
	Error	BOOL	Not used
	ErrorID	WORD	Not used
	Errorstop	BOOL	Set when the axis is in Errorstop state
	Disable	BOOL	Set when the axis is in Disable state
	Stopping	BOOL	Set when the axis is in Stopping state
	Standstill	BOOL	Set when the axis is in Standstill state
	DiscreteMotion	BOOL	Set when the axis is in DiscreteMotion state
	ContinuousMotion	BOOL	Set when the axis is in ContinuousMotion state
	SynchronizedMotion	BOOL	Set when the axis is in SynchronizedMotion state
	Homing	BOOL	Set when the axis is in Homing state

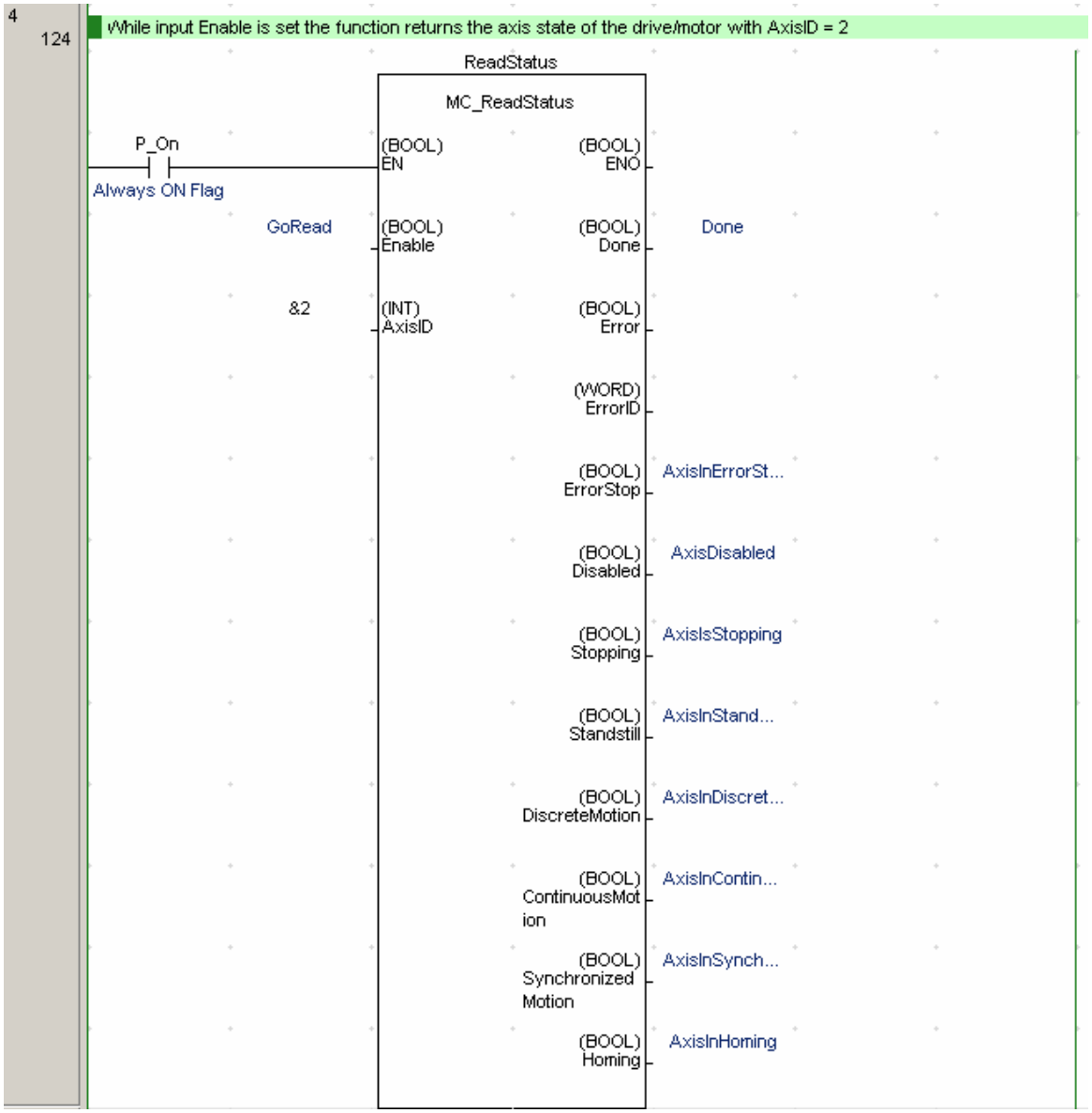
Description: As long as the **Enable** input is true, the function returns the axis state, defined at PLC level, through the corresponding output. **Done** output is set if valid outputs are available.

All outputs remain set until **Enable** input is reset, but at least for one block call.

Error and **ErrorID** outputs are reserved for future development and have no associated functionality.

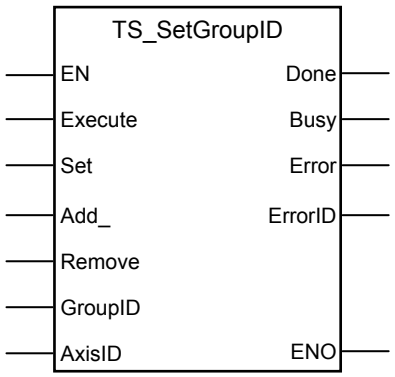
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:



3.6.36 TS_SetGroupID

Symbol:



Parameter description:

	Parameter	Data type	Description
Input	EN	BOOL	Enable function block execution
	Execute	BOOL	On rising edge the configuration command is sent
	Set	BOOL	Set the GroupID of the axis
	Add_	BOOL	Add the GroupID to the axis
	Remove	BOOL	Remove the GroupID from the selected axis
	GroupID	INT	The value of GroupID used by the selected operation
	AxisID	INT	Axis information
Output	ENO	BOOL	Status of function block execution
	Done	BOOL	The position read successfully
	Busy	BOOL	
	Error	BOOL	Is set if an error has occurred in the execution of the function block
	ErrorID	WORD	Information about the error occurred

Description: The function block changes the group ID of the drive/motor with **AxisID**. Each drive/motor can be programmed to be member of one or several of the 8 possible groups.

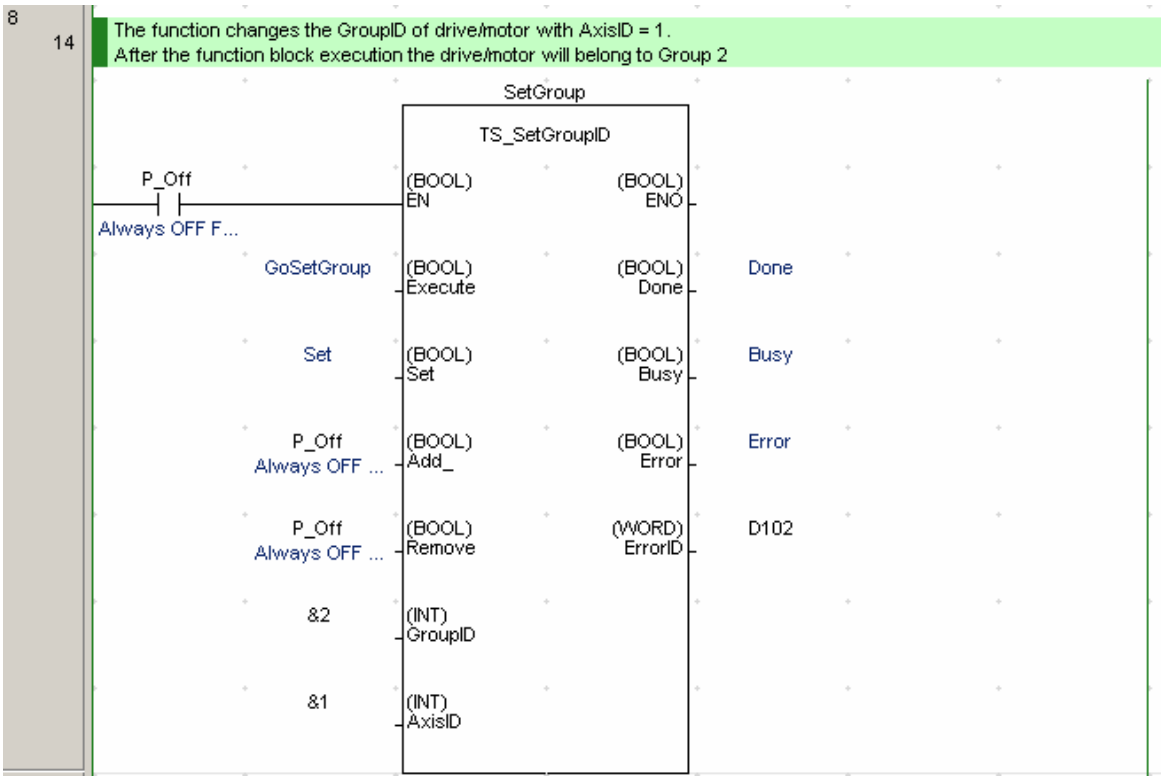
The group ID of an axis can have any value between 1 and 255. For example if the group ID is 11 (1011b) this means that the axis will receive all messages sent to groups 1, 2 and 4.

All changes (set/add/remove) of group ID are executed with the rising edge of **Execute** input using the value read from **GroupID**. The function block tests the inputs **Set**, **Add, Remove** and executes the operation associated to the **last input found set**. **Busy** output will remain set for one function block call, in the next call, the function block sets the **Done** output and resets **Busy**. The output **Done** remain set until **Enable** input is reset.

The inputs **Set**/**Add**/**Remove** must be set, according with the requested operation, before the transition of **Execute** input occurs. If none of the inputs **Set**, **Add** or **Remove** is set when **Execute** transition low to high occurs the output **Error** is set and the **ErrorID** becomes equal with 0x8000.

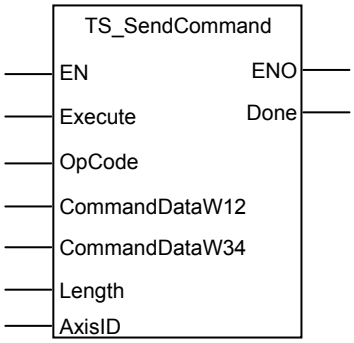
The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:


8
14
The function changes the GroupID of drive/motor with AxisID = 1.
After the function block execution the drive/motor will belong to Group 2
SetGroup
TS_SetGroupID
P_Off
Always OFF F...
(BOOL) EN
(BOOL) ENO
GoSetGroup
(BOOL) Execute
(BOOL) Done
Done
Set
(BOOL) Set
(BOOL) Busy
Busy
P_Off
Always OFF ...
(BOOL) Add_
(BOOL) Error
Error
P_Off
Always OFF ...
(BOOL) Remove
(WORD) ErrorID
D102
&2
(INT) GroupID
&1
(INT) AxisID

3.6.37 TS_SendCommand

Symbol:



Parameters description:

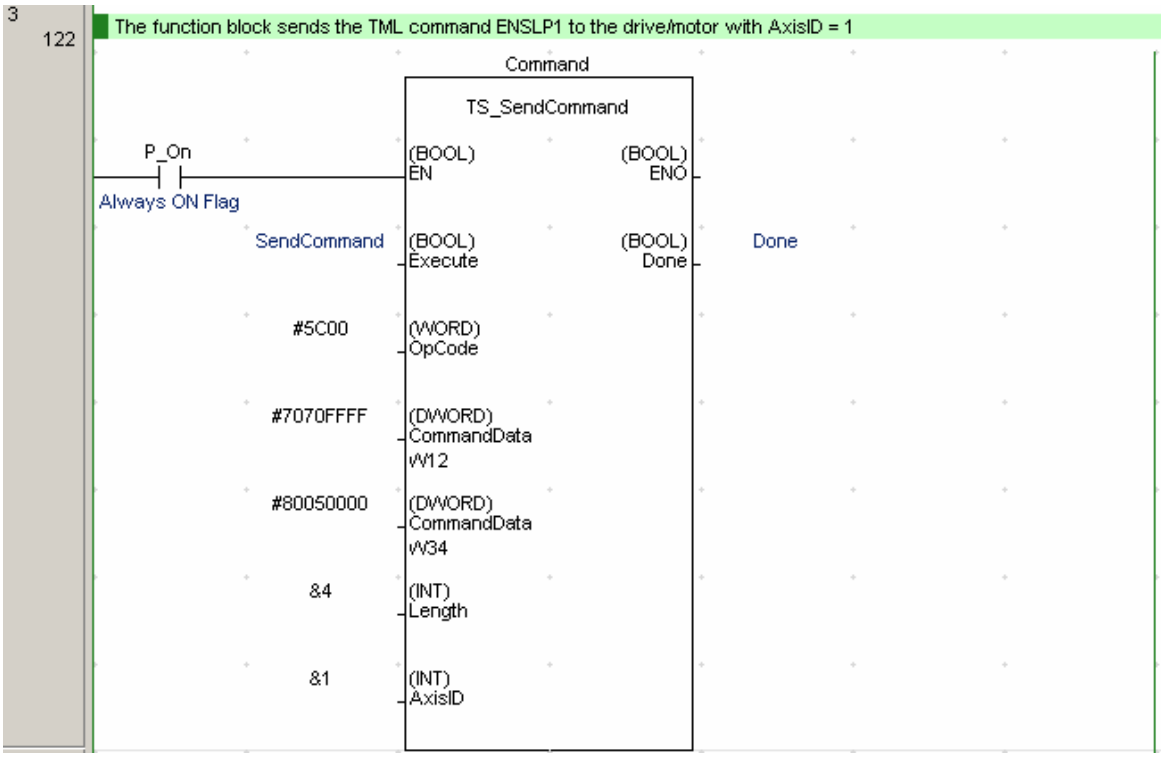
	Parameter	Data type	Description
Input	EN	BOOL	Enable function execution
	Execute	BOOL	Send stop command at rising edge
	OpCode	WORD	TML command operation code
	CommandDataW12	DWORD	Data words 1 and 2 of TML command
	CommandDataW34	DWORD	Data words 3 and 4 of TML command
	Length	INT	Number of data bytes
	AxisID	INT	Axis information
Output	ENO	BOOL	Status of function execution
	Done	BOOL	The TML command sent

Description: The function sends a TML command to the drive/motor with **AxisID**.

With the rising edge of **Execute** the function block builds and sends a TML command from its binary code. The binary code for a TML command can be found with **Binary code viewer** a tool integrated in **EasyMotion Studio**. The **Done** output is when the command is sent successfully

The input **EN** must be connected to the left bus through **P_On** flag for proper execution of the function

Example:


3
122
The function block sends the TML command ENSLP1 to the drive/motor with AxisID = 1
Command
TS_SendCommand
P_On
Always ON Flag
(BOOL)
EN
(BOOL)
ENO
SendCommand
(BOOL)
Execute
(BOOL)
Done
Done
#5C00
(WORD)
OpCode
#7070FFFF
(DWORD)
CommandData
W12
#80050000
(DWORD)
CommandData
W34
&4
(INT)
Length
&1
(INT)
AxisID

TECHNOSOFT