Modicon M340

BMX MSP 0200 (PTO) module
Unity Pro

04/2009

EIO0000000058.01

www.schneider-electric.com

Schneider Electric

Schneider Electric assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

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

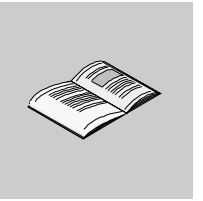
All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

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

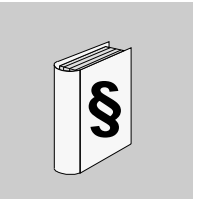
Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

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

Table of Contents



Safety Information



Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

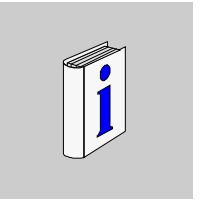
⚠ CAUTION
CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

CAUTION
CAUTION, used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, **can result in** equipment damage.

PLEASE NOTE

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

About the Book



At a Glance

Document Scope

This documentation describes the hardware and software implementation of the PTO module for Modicon M340 PLCs.

Validity Note

This documentation is valid for Unity Pro v4.1.

Product Related Information

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

Follow all local and national safety codes and standards.

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

User Comments

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

BMX MSP 0200 Product Overview

I

Overview

This part gives an overview of the BMX MSP 0200 PTO module and its technical specifications.

What's in this Part?

This part contains the following chapters:

Module Introduction

1

Overview

This chapter gives a quick description of the PTO module.

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.1
- Each implementation of the BMX MSP Pulse Train Offer must be individually and thoroughly tested for proper operation before being placed into service.

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

What's in this Chapter?

This chapter contains the following topics:

General Information on PTO Function

At a Glance

The main purpose of the MSP 0200 PTO module is to control third party drives with open collector input and integrated position loop.

Description

In order to do this, the PTO module provides a square wave output for a specified number of pulses and a specified cycle time. It can be programmed to produce either one train of pulses or a pulse profile consisting of multiple trains of pulses.

For example, a pulse profile can be used to control a stepper motor or servo-motor through a simple ramp up, run, and ramp down sequence or more complicated sequences.

The control positioning is achieved according to an open loop mode meaning without the need of feedback information on the real position of the mobile.

General Information about the BMX MSP 0200 PTO Module

Introduction

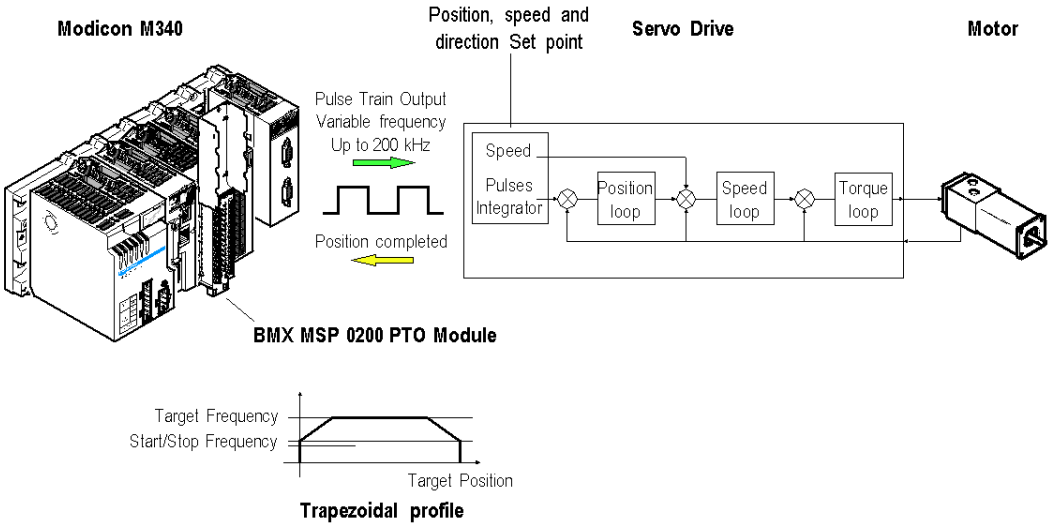
The BMX MSP 0200 PTO module is a standard format module that enables to control of a third party drives with an open collector compatible input and integrated position loop.

The module has 2 PTO channels.

This module may be installed in any available slot in a Modicon M340 PLC station rack.

Illustration

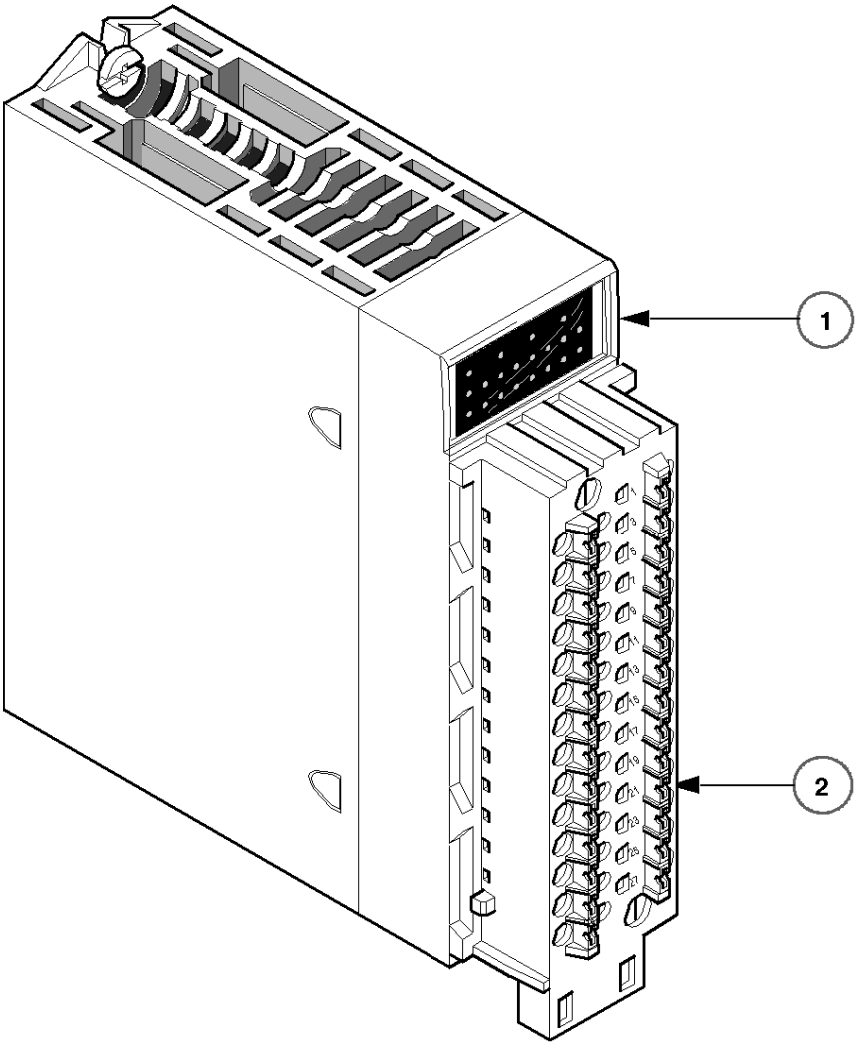
The following illustration shows the command diagram to a third party drive.



Physical Description of the BMX MSP 0200 PTO module

Illustration

The figures below present the BMX MSP 0200 PTO module :



Physical Elements of the Modules

This table presents the elements of the MSP 0200 PTO module :

Number	Description
1	Module state LEDs: • State LEDs at module level • State LEDs at channel level
2	28-pin connector

Accessories

The BMX MSP 0200 PTO module requires the use of a BMX FTB 2820 Spring type, 28-pin terminal block.

Board unit characteristics

Overview

This is the technical description of the board unit characteristics

Caracteristics table

Board unit characteristics

Consumption 3.3 V	Typical	< 150 mA
	Maximum	200 mA
Consumption 24 V pre-actuator	Without load	Maximum: 35 mA
Dissipated power		AT 24V, 0 active input: 1.4 W AT 24V, 8 active inputs: 2.8 W
Dielectric strength (internal logic)	Primary / Secondaries	1500 Vrms
	Between channel groups	Not Isolated
Insulation resistance		>10 MΩ
Temperature derating		No derating -25 to 70° C

⚠ DANGER

Hazardous Performance

Respect the working temperature range.

Failure to follow these instructions will result in death or serious injury.

PTO module installation

2

Overview

This chapter provides information to install the module.

What's in this Chapter?

This chapter contains the following topics:

Mounting the BMX MSP 0200 PTO Module

At a Glance

The BMX MSP 0200 PTO module is powered by the rack bus. The module itself may be installed or removed without turning off the power supply to the rack.

Mounting operations (installation, assembly and disassembly) are described below.

Installation Precautions

The PTO modules may be installed in any of the positions in the rack except for the first two (marked PS and 00) which are reserved for the rack's power supply module and the processor respectively. Power is supplied by the bus at the bottom of the rack (3.3 V and 24 V).

Before installing a module, you must take off the protective cap from the module connector located on the rack.

⚠ DANGER

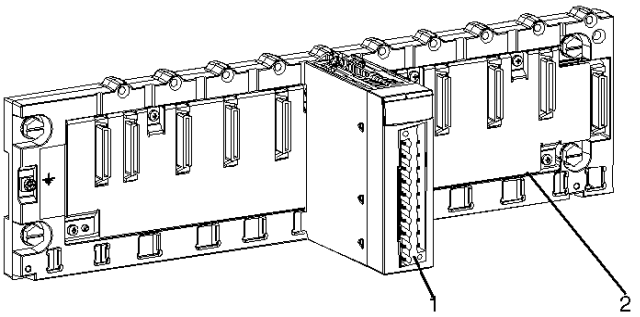
HAZARD OF ELECTRIC SHOCK

- disconnect voltage supplying sensors and pre-actuators before plugging / unplugging the terminal block on the module.
- remove the terminal block before plugging / unplugging the module on the rack.

Failure to follow these instructions will result in death or serious injury.

Installation

The following diagram below shows a PTO module mounted in the rack:

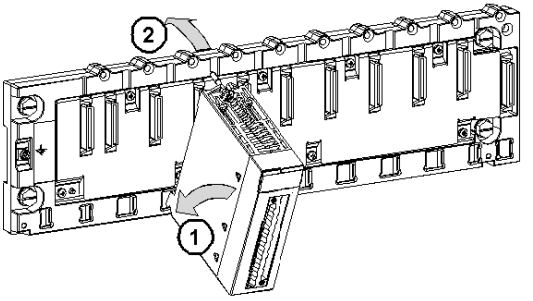
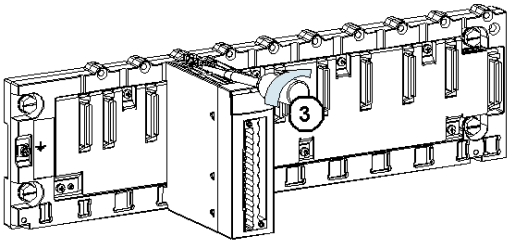


The following table describes the different elements which make up the assembly below:

Number	Description
1	BMX MSP 0200 PTO module
2	Standard rack

Installing the Module in the Rack

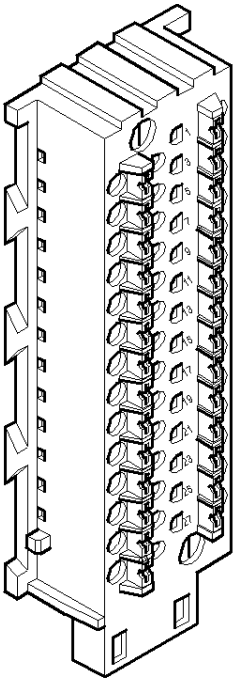
The following table shows the procedure for mounting the BMX MSP 0200 PTO modules in the rack:

Step	Action	Illustration
1	Position the locating pins situated at the rear of the module (on the bottom part) in the corresponding slots in the rack. Note: Before positioning the pins, make sure you have removed the protective cover *(see Modicon M340 Using Unity Pro, Processors, Racks, and Power Supply Modules, Setup Manual)*.	Steps 1 and 2 
2	Swivel the module towards the top of the rack so that the module sits flush with the back of the rack. It is now set in position.	
3	Tighten the mounting screw to ensure that the module is held in place on the rack. Tightening torque: Max. 1.5 N.m	Step 3 

Mounting the BMX FTB 2820 Terminal Block

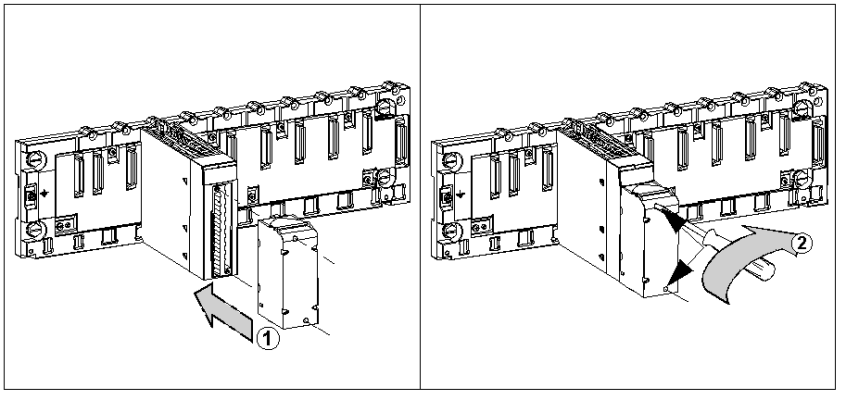
At a Glance

BMX MSP 0200 PTO modules requires the BMX FTB 2820 Spring type, 28-pin terminal block to be inserted into the front of the module. These fitting operations (assembly and disassembly) are described below.



Installing the 28-Pin Terminal Block

The following table shows the procedure for assembling the 28-pin terminal block onto a BMX MSP 0200 PTO module:



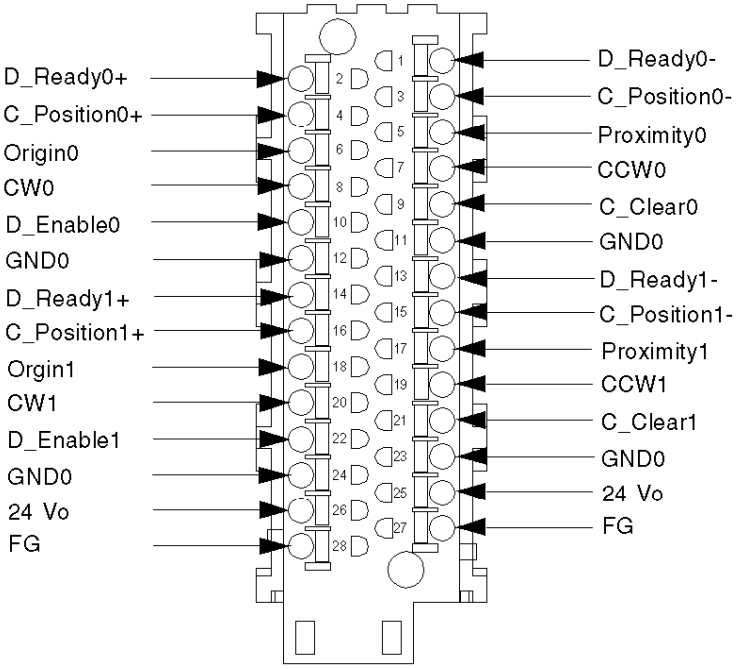
Assembly procedure:

Step	Action
1	Once the module is in place on the rack, install the terminal block by inserting the terminal block encoder (the rear lower part of the terminal) into the module's encoder (the front lower part of the module), as shown above.
2	Fix the terminal block to the module by tightening the 2 mounting screws located on the lower and upper parts of the terminal block. Tightening torque: 0.4 N.m.

NOTE: If the screws are not tightened, there is a risk that the terminal block will not be properly fixed to the module.

28 Pin Terminal Block Arrangements

The terminal block is arranged as followed:



⚠ CAUTION

Reverse connection of external supply

Follow the wiring *(see page 31)*, mounting and installation *(see page 19)* instructions.

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

How to Avoid Electromagnetic Interference

Overview

⚠ WARNING

UNEXPECTED EQUIPMENT OPERATION

Follow those instructions to reduce electromagnetic perturbations:

- adapt the programmable filtering to the frequency applied at the inputs, or
- use a shielded cable and connect the shield to pins 27 and 28 (functional ground) of the module.

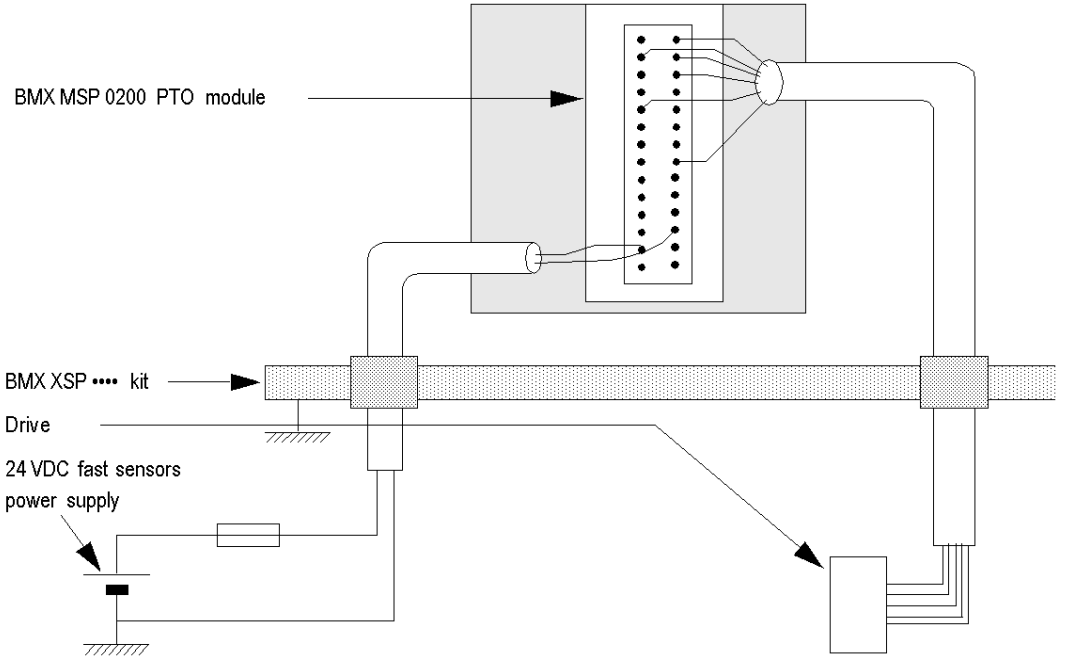
In a highly disturbed environment,

- use the BMX XSP 0400/0600/0800/1200 electromagnetic protection kit *(see Modicon M340 Using Unity Pro, Processors, Racks, and Power Supply Modules, Setup Manual)* (See Modicon M340 using Unity Pro, Processors, Racks and Power Supply Modules, BMX XSP xxx Protection Bar) to connect the shielding without programmable filtering and
- use a stabilised 24 VDC supply for inputs and a shielded cable for connecting the supply to the module.
- use a shielded cable for each PTO channel respectively and note that 24VDC and GND must be included in the shielded cable. (Each shielded cable includes 4 inputs, 4 outputs, 24 VDC and GND.)

Electromagnetic perturbations may cause the application to operate in an unexpected manner.

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

The figure below shows the recommended circuit for high-noise environment using the BMX XSP 0400/0600/0800/1200 electromagnetic protection kit:



⚠ CAUTION

IMPROPER FUSE SELECTION

Use fast acting fuses to protect the electronic components of the module from overcurrent and reverse polarity of the input/output supplies. Improper fuse selection could result to damage to the module.

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

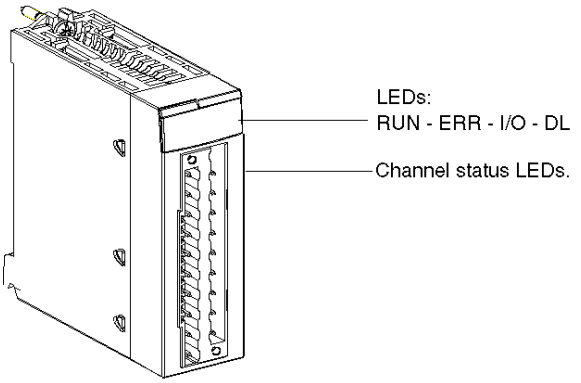
LED indicator

At a Glance

The BMX MSP 0200 PTO module is equipped with LEDs that display the module's channels status and detected errors.

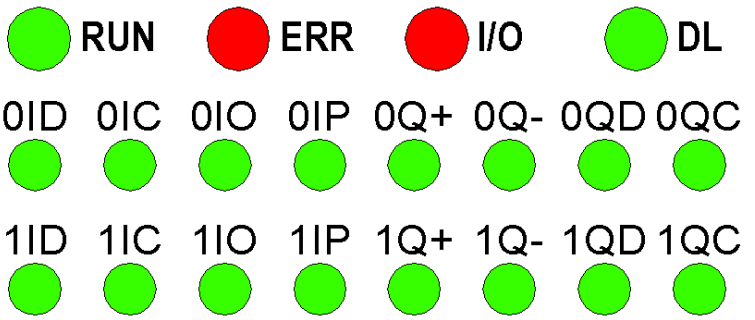
Illustration

The figure below shows the position of the channel status display LEDs on the front panel of the PTO module.



Display Panels

LED display



The top row of LEDs indicates module information.

The middle row 0xx corresponds to PTO channel 0

The bottom row 1xx corresponds to PTO channel 1

The inputs for both rows of LEDs are represented in the following way: (y = 0 or 1 depending on the PTO channel)

- LED yID: Drive_Ready&Emergency Input for channel y
- LED yIC: Counter_in_Position Input for channel y
- LED yIO: Origin Input for channel y
- LED yIP: Proximity&LimitSwitch Input for channel y

The outputs for both rows of LEDs are represented the following way: (y = 0 or 1 depending on the PTO channel)

- LED yQ+: PTO CW Output for channel y
- LED yQ-: PTO CCW Output for channel y
- LED yQD: Drive_Enable Output for channel y
- LED yQC: Counter_Clear Output for channel y

When a voltage is present on an input or output, the corresponding LED is lit.

Description

The following table allows you to perform diagnostics of the module status according to the LEDs: RUN, ERR, I/O and channels (LEDs 0ID to 1QC):

Module status	**Status LEDs**			
	RUN	**ERR**	**I/O**	**LEDs 0ID to 1QC**
The unit is not receiving power or LEDs are out of order.	○	○	○	x
The unit is configuring its channels	⊗	○	○	x
Internal error detected in module	○	●	○	x
Unit has lost communication with CPU	●	⊗	○	x
Channels are operational.	●	○	○	LEDs 0ID to 1QC are representative of the state of the corresponding input/output. ● if Channel state active ○ if Channel state inactive
I/O Error detected	●	○	●	⊗ Power lost ⊗ Short-circuit / Overload (only for Output LEDs)
Legend:				
○ LED off				
⊗ LED Blinking (slow)				
⊗ LED flashing rapidly				
● LED on				

Module status	Status LEDs			
	RUN	ERR	I/O	LEDs 0ID to 1QC
No PTO Channel configured	○	⊗	○	x
Unit in self-tests	⊗	⊗	⊗	x
Unit has lost communication with CPU	●	⊗	○	x
Channels are operational.	●	○	○	LEDs 0ID to 1QC are representative of the state of the corresponding input/output. ● if Channel state active ○ if Channel state inactive
I/O Error detected	●	○	●	⊗ Power lost ⊗ Short-circuit / Overload (only for Output LEDs)

Legend:
○ LED off
⊗ LED Blinking (slow)
⊗ LED flashing rapidly
● LED on

The 4th standard LED in the first line – "DL" – is used during firmware download:

RUN	ERR	IO	DL	Status
⊗	○	○	●	Start of download
⊗	○	○	⊗	Download in progress
○	●	○	⊗	Download error
●	○	○	●	End of download
⊗	⊗	⊗	⊗	Upgrade done. Module to be restarted
○	⊗	○	○	Upgrade done with identical version. Module to be restarted

I/O Specification

3

Overview

This chapter contains information about the inputs / outputs of the PTO module.

NOTE: The PTO performances described in this chapter are only valid with correct wiring as indicated in this documentation.

What's in this Chapter?

This chapter contains the following topics:

Inputs for PTO

Overview

There are 4 auxiliary inputs for every PTO channel:
- Auxiliary Input 0: Drive_Ready&Emergency
- Auxiliary Input 1: Counter_in_Position
- Auxiliary Input 2: Origin (Signal used only for homing mode)
- Auxiliary Input 3: Proximity&LimitSwitch

⚠ DANGER

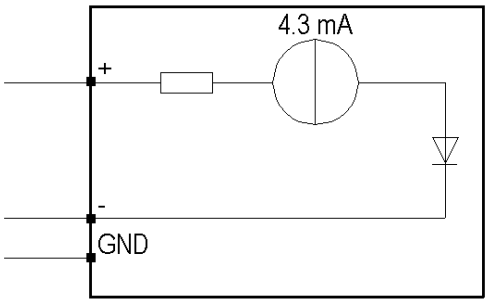
HAZARD OF ELECTRIC SHOCK

- disconnect voltage supplying sensors and pre-actuators before plugging / unplugging the terminal block on the module.
- remove the terminal block before plugging / unplugging the module on the rack.

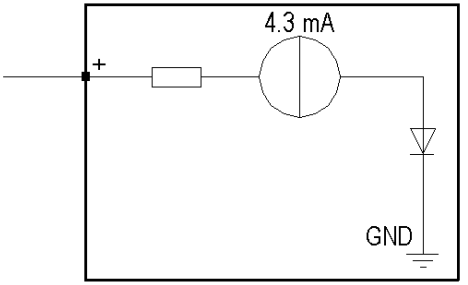
Failure to follow these instructions will result in death or serious injury.

Diagram

Drive_Ready&Emergency inputs or Counter_in_Position (SINK/SOURCE input type):

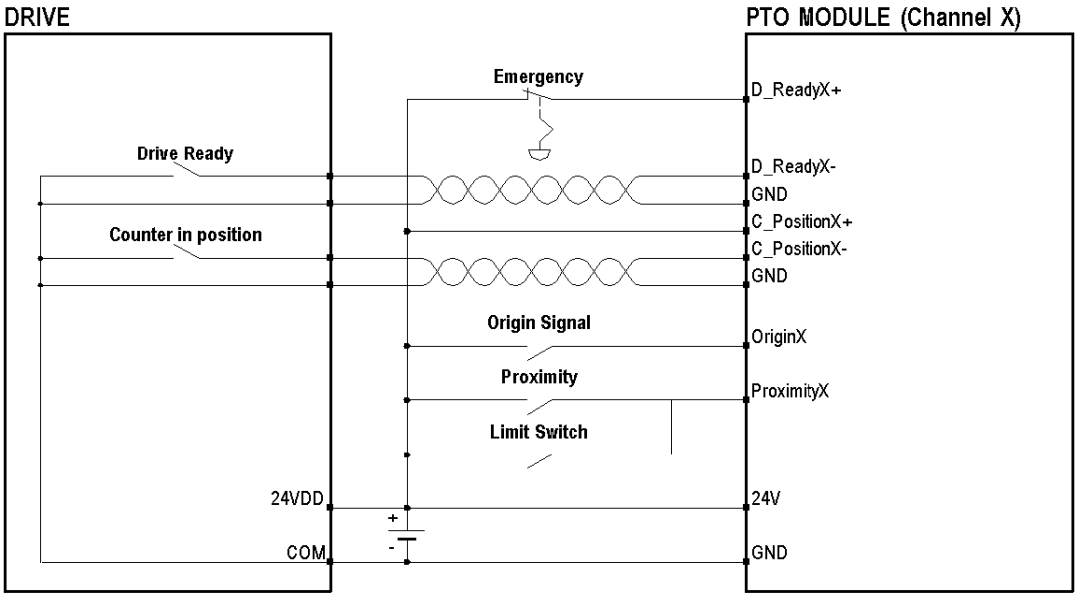


Origin or Proximity&LimitSwitch inputs (SINK input type):



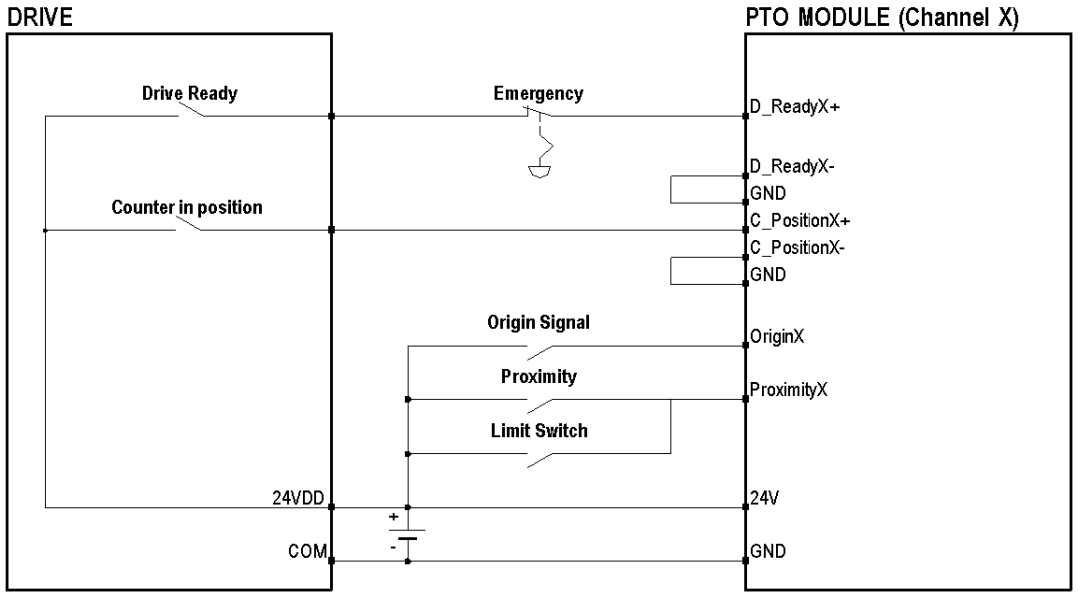
Wiring the inputs

If the Drive_Ready&Emergency and Counter_in_Position outputs from the drive are of SINK type:



A twisted pair cable is necessary to connect the module to the drive.

If the Drive_Ready&Emergency and Counter_in_Position outputs from the drive are of SOURCE type:



NOTE: In order to stop the PTO module when the PLC is set to STOP, connect the D_ReadyX+ input to the PTO module via a BMX DRA (0805 or 1605). This will make all outputs stop when the D_Ready&Emergency input is set to 0.

⚠ CAUTION

Insignificant input, short-Circuit or overload

Respect mounting and installation procedure and use the given wiring cable diagrams when using the PTO module.

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

Input Characteristics

Input Characteristics Table

The table below describes the BMX MSP 0200 input characteristics

Characteristics		**Input**
Nominal input values	Voltage	24 VDC
	Current	4.3 mA
Input limit values	Voltage at state 1	≥11 V
	Voltage at state 0	5V
	Current at state 1	> 2 mA for U ≥ 11 V
	Current at state 0	< 1.5 mA
	Sensor supply (Ripple included)	From 19 to 30 V
Input Impedance	At U_{nom}	Current limited to 4.3 mA
Response time	Origin Input & Proximity Input	<60 µs without bounce filter
	Position Completed Input & Drive Ready Input	<200 µs without bounce filter
Reverse polarity		Protected
IEC61131-2- Edition 2 (2003)		Type 3
Compatibility	(2 wires, 3 wires prox. Sensors)	IEC 947-5-2
Dielectric strength	Primaries / secondary	1500 VRMS
Insulation resistance		> 10 MΩ
Input type	Origin Input & Proximity&LimitSwitch input	Input Current sink
	Counter_in_Position input& Drive_Ready&Emergency input	Current sink or source
Input paralleling		Yes
Sensor voltage Monitoring threshold	Normal condition	> 12 VDC
	Low-voltage condition	< 8 VDC

Pulse Train Characteristics

Overview

The PTO function provides a square wave output for a specified number of pulses and a specified cycle time.

The PTO function can be programmed to produce either one train of pulses or a pulse profile consisting of multiple trains of pulses. For example, a pulse profile can be used to control a stepper motor through a simple ramp up, run, and ramp down sequence or more complicated sequences. The control positioning is achieved according to an open loop mode, meaning without the need for feedback information on the real position. The position loop is integrated in the servo-drive.

Characteristics

Number of pulses is from -2,147,483,648 to 2,147,483,647 (32 bits depth)

Maximum frequency:
- For CW / CCW and pulse/direction modes with a cable length up to 10 m (32.81ft), the maximum frequency is 200 kHz.
- For A/B phases control mode the maximum frequency is 100 kHz.

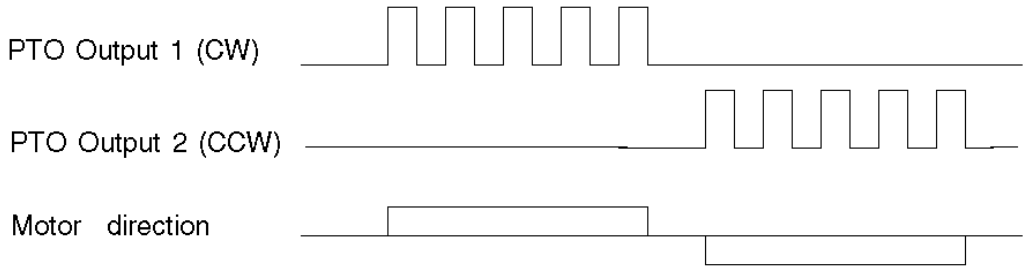
Average frequency accuracy:
- 0.2% up to 50 kHz
- Increasing up to 0.5% around 200 kHz

NOTE: There are some limitations in case of usage of USIC + Lexium 05 and a 24 V power supply

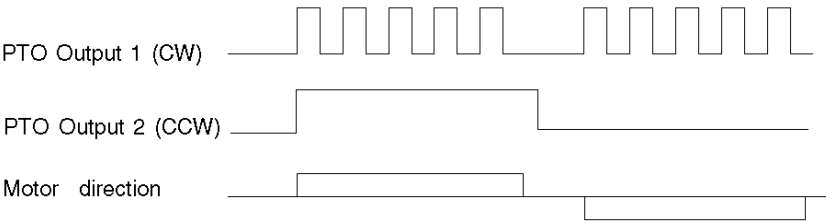
Pulse train output modes

There are 3 types of pulse train output mode that can be configured.

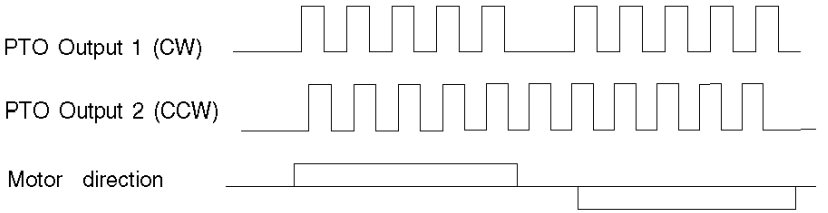
Pulse+ /Pulse- (CW/CCW):



Pulse + direction:



A/B phases (Quadrature):



In order to select the axis movement direction in accordance with the motion command direction on the PTO module, the PTO Unity Software has 3 pulse-train output configuration modes, each allowing reverse direction.

⚠ WARNING

AXIS DIRECTION REVERSED

The following axis adjustment parameter must be taken into account:

- The PTO module output characteristics: positive direction is defined by the logical state 1 corresponding to the state of the "sink" type active physical output (low state).
- The type of wiring circuit between PTO module and drive: compatible RS422 input with 5 V polarization, compatible RS422 input with 24 V polarization, 24 V source inputs, drive through USIC accessory.
- The active input level of drive.
- The kinematic system (direction depending on the type of axis, gear box used or not...).

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

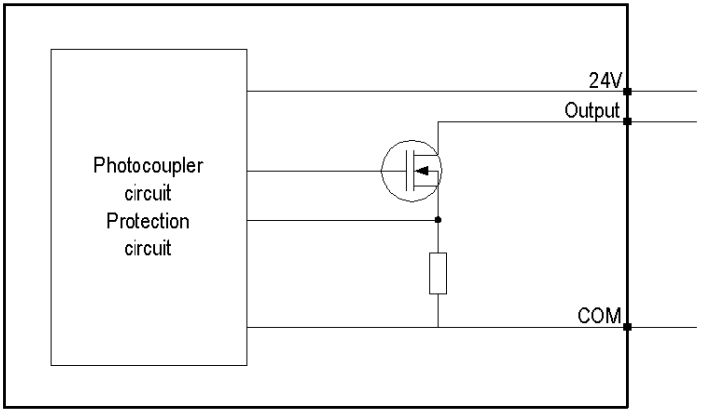
Output Command Drive

Overview

The following output interface wiring is necessary regarding the drive's available input. There are four points for each PTO output

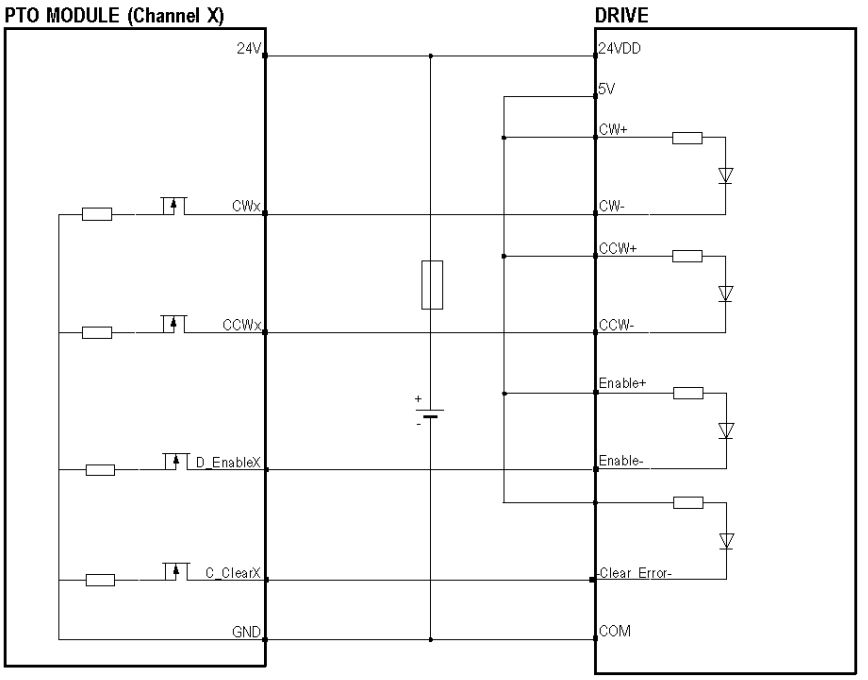
Output Type

Internal output circuit:



RS422 Compatible Inputs and 5 V Polarisation

Drive with RS422 compatible inputs and 5 V polarisation



⚠ CAUTION

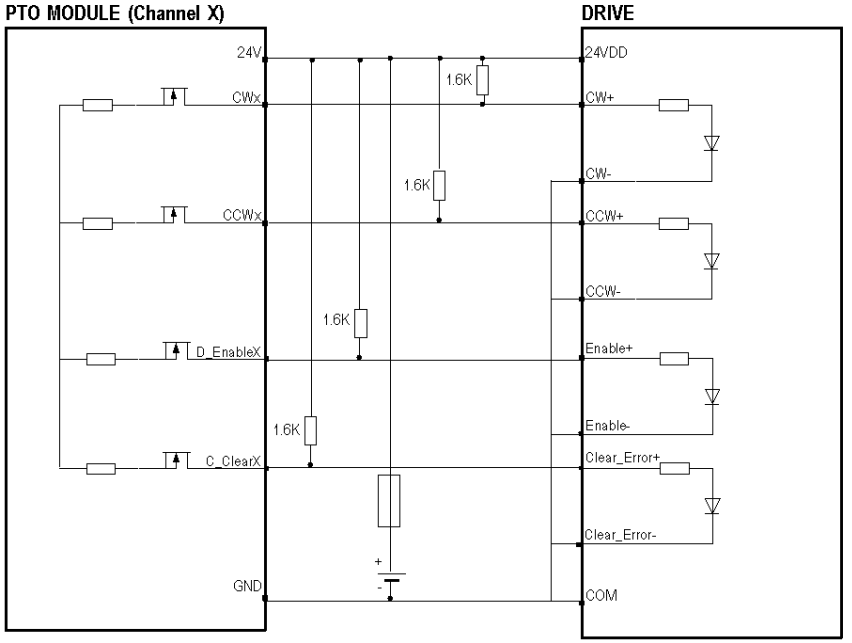
IMPROPER FUSE SELECTION

Use fast acting fuses to protect the electronic components of the module from overcurrent and reverse polarity of the input/output supplies. Improper fuse selection could result to damage to the module.

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

RS422 Compatible Inputs and 24 V Polarisation

Drive with RS422 compatible inputs and 24 V supply



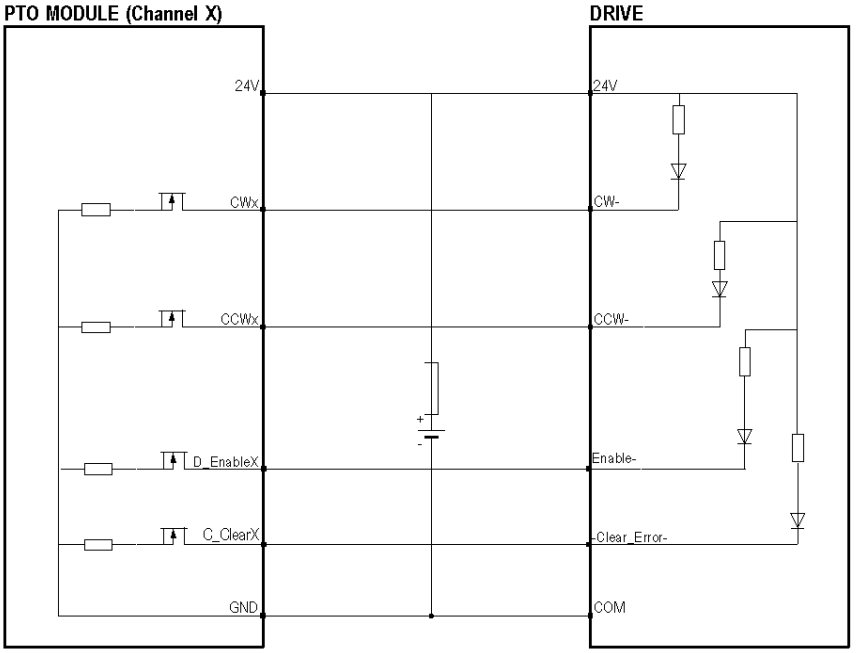
⚠ CAUTION

IMPROPER FUSE SELECTION

Use fast acting fuses to protect the electronic components of the module from overcurrent and reverse polarity of the input/output supplies. Improper fuse selection could result to damage to the module.

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

24 VDC Source Input



Only SOURCE inputs (100 mA maximum) are compatible with Drive_Enable and Counter_Clear

NOTE: The pre-actuator power supply and the output external power supply should be from the same source.

⚠ CAUTION

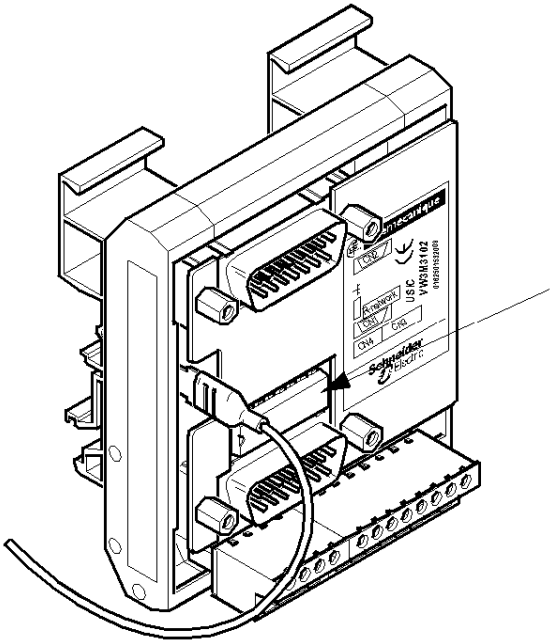
IMPROPER FUSE SELECTION

Use fast acting fuses to protect the electronic components of the module from overcurrent and reverse polarity of the input/output supplies. Improper fuse selection could result to damage to the module.

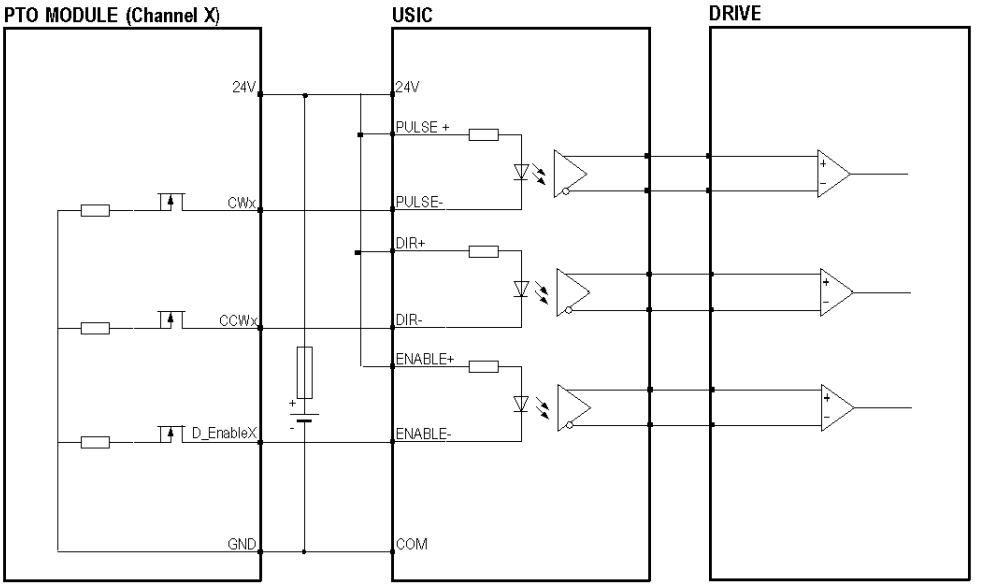
Failure to follow these instructions can result in injury or equipment damage.

USIC: Accessory for RS422 interface

The Lexium drives or drives with RS422 line-receiver cannot be connected directly to the PTO channel. It is necessary to use a Universal Signal Interface Converter (ref: VW3M3102), an external RS422 accessory to connect the drive to the PTO channel.



Wiring the PTO module to a drive via the USIC:



For connection from PTO channel to USIC use the prefabricated cable (ref: VW3M8210R05) available in Schneider catalogue.

To connect the USIC to the drive, a prefabricated cable (ref: VW3M8201R50) can be used with a SUB-D15 connector wired as shown in the example *(see page 59)*.

⚠ WARNING

Material Destruction

Remove the network resistance from the USIC.

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

⚠ CAUTION

IMPROPER FUSE SELECTION

Use fast acting fuses to protect the electronic components of the module from overcurrent and reverse polarity of the input/output supplies. Improper fuse selection could result to damage to the module.

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

⚠ WARNING

RANDOM COMMAND AND PERFORMANCE REDUCTION

Do not use a cable with a length above 0.5 m (1.64 ft).

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

Protection of Outputs

Each output is protected against short-circuit and overload.

The overload detection starts at 0.13 A as load current.

In case of detected error:

- The peak current will be limited to 1 A for 50 µs,
- The outputs will be automatically switched off.
- A fast auto-recovery will be attempted four times before a short-circuit condition is registered.
- This condition is reported in the channel status information (EXT_FLT_OUTPUTS: %MWr.m.c.2.1), and after waiting a second, a recovery is reattemped.

NOTE: Error detection upon one output sets all outputs of the connector in off state. This condition is then reported to the status word of all channels on the connector.

⚠ CAUTION

Output Short-Circuit or Overload

Respect mounting procedure and use the given wiring cable

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

Output Characteristics

Output Characteristics Table

The table below describes the output characteristics of the BMX MSP 0200 in the documented wiring configuration.

Characteristics		**PTO output**	**Auxiliary output**
Nominal values	Voltage	24 VDC	
	Current	0.05 A	
Limit values	Voltage	19---30V	
	Current/Point	0.1 A (Disjunction at 0.13 A)	
	Current/PTO Channel	0.4 A	
Leakage current	At state 0	< 50 µA	
Residual voltage	At state 1	< 150 mV (with drive interface)	
Minimum load impedance		15 kΩ	
Maximum capacity		100 nF	
Output frequency		• 200 kHz with cable length < 10 m (32.8 ft) with the RS422 compatible circuits. • 100 kHz with cable length < 5 m (16.4 ft) with the normal source input circuit in 24V. • 200 kHz with USIC and VW3M8210R05 (0.5 m (1.64 ft)) connected to PTO side.	< 150 µs
Max overload time		50µs	
Switching frequency on inductive load		Not applicable (only resistive load allowed)	
Output paralleling		Not applicable (dedicated function by output)	
Compatibility with DC inputs		With RS422: 7 mA inputs With SOURCE inputs: 5 V to 24 V With signal converter (USIC)	
Built-in protection	Against overvoltage	No	No
	Against reverse polarity	Yes, by reverse-mounted diode.	Yes, by reverse-mounted diode.
	Against short circuits and overloads	Yes, by current limiter and electronic circuit-breaker for one PTO channel (4 outputs) 0.13 A < Id (by output) < 1 A	
Preactuator voltage Monitoring threshold	OK	> 14 V	> 14 V
	On low-voltage condition	< 8 V	< 8 V
Monitoring response time	On disappearance	1.2 ms < T < 1.5 ms	
	On appearance	1.2 ms < T < 1.5 ms	

Set up sequence

4

Set up Sequence

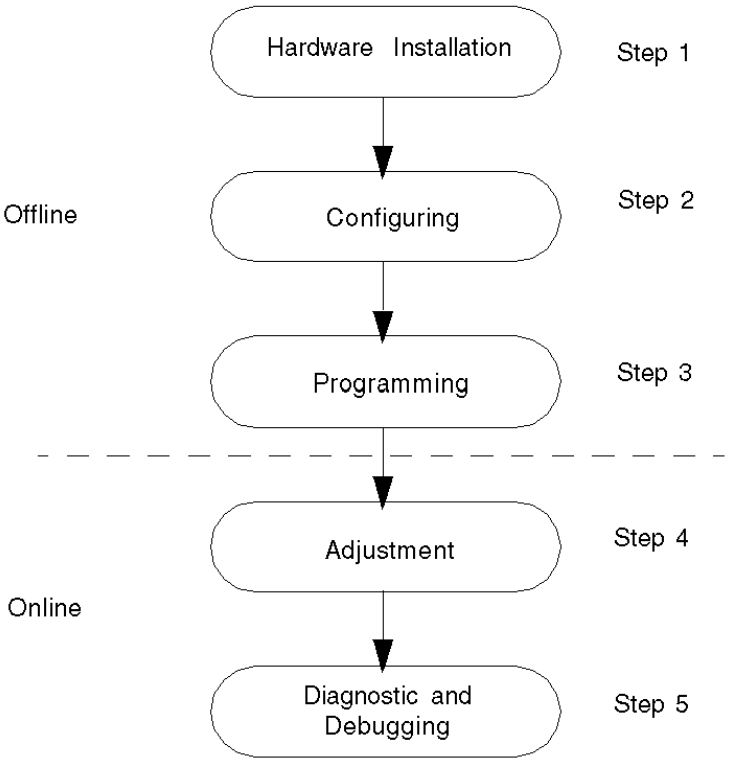
Overview

The software installation of the application-specific modules is carried out from the various Unity Pro editors in offline and online mode.

IWhen a processor is not available, Unity Pro allows to carry out an initial test using the simulator.

Sequence

This is a 5-step sequence:



Step 1: PTO module installation *(see page 19)* and I/O Specification *(see page 31)*

Step 2: Configuration parameters *(see page 105)*

Step 3: Programming features *(see page 115)*

Step 4: Adjustment *(see page 207)*

Step 5: Diagnostic and debugging the MSP 0200 PTO module *(see page 213)*

PTO Module Start Up Example for a Single Axis Configuration

II

Overview

This part provides an example of using the BMX MSP 0200 PTO module.

What's in this Part?

This part contains the following chapters:

Example Overview

5

At a glance

This chapter describes the overview structure of the start up example for using the PTO module.

What's in this Chapter?

This chapter contains the following topics:

Example Introduction

At a Glance

This example describes the steps in the installation of a drive using a BMX MSP 0200 PTO module. These steps are:

- Hardware installation
- Software configuration
- Programming a movement
- Diagnosis and debugging

Objective

The example's objective is to give a full review of the BMX MSP 0200 PTO module's implementation by creating a fully operational program.

Requirements

The hardware needed to do this example is:

- A Modicon M340 platform (Rack, CPU and Power Supply)
- A BMX MSP 0200 PTO module
- A Lexium 05
- USIC module

The software needed to do this example is:

- Unity Pro 4.0
- Power Suite 2.5

NOTE: In this example, a Lexium 05 with a USIC is used but any other drive with an open collector compatible input and integrated position loop would be convenient for the example.

NOTE: Basic knowledge of Unity Pro programming is required for this example.

Application Background

At a Glance

The application described is a packet conveyor manager: a machine that contains a product transport conveyor and a digital jack system which will place each product in a free cell. Once a product is detected to sort in a cell, the application starts.

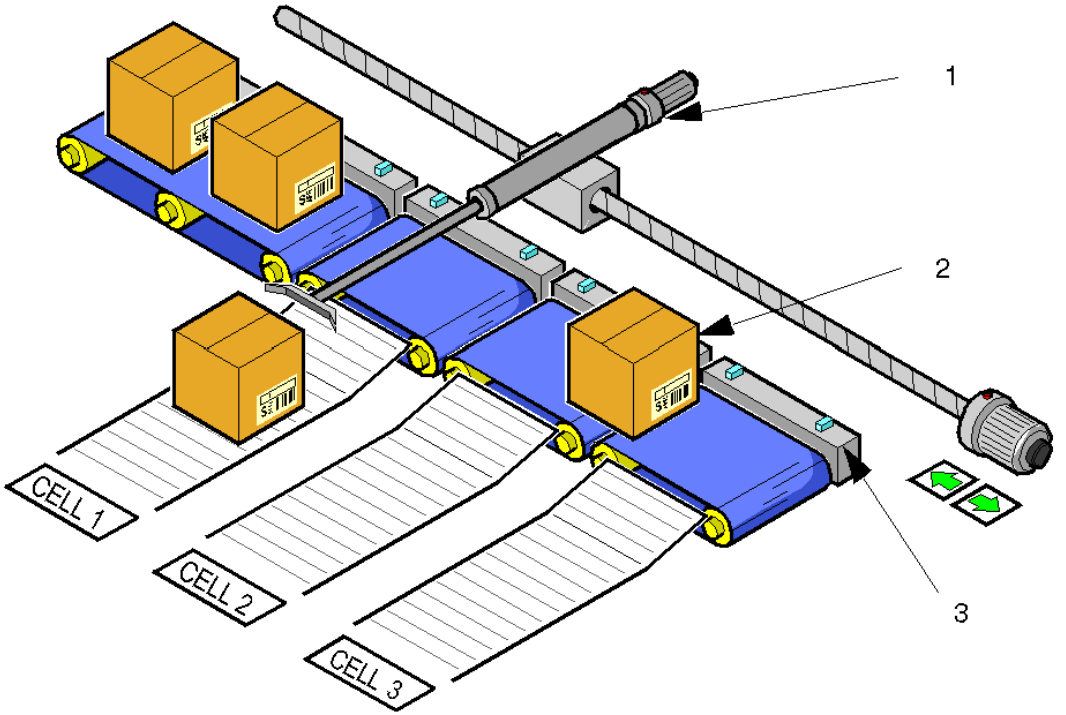
This system has 2 orthogonal linear axes equipped with drives:

- Drive 1 for the Jack that pushes the product into the cell
- Drive 2 for the transverse axis

The application example deals with the Jack's movement once a product is detected.

Illustration

Packet conveyor managemer



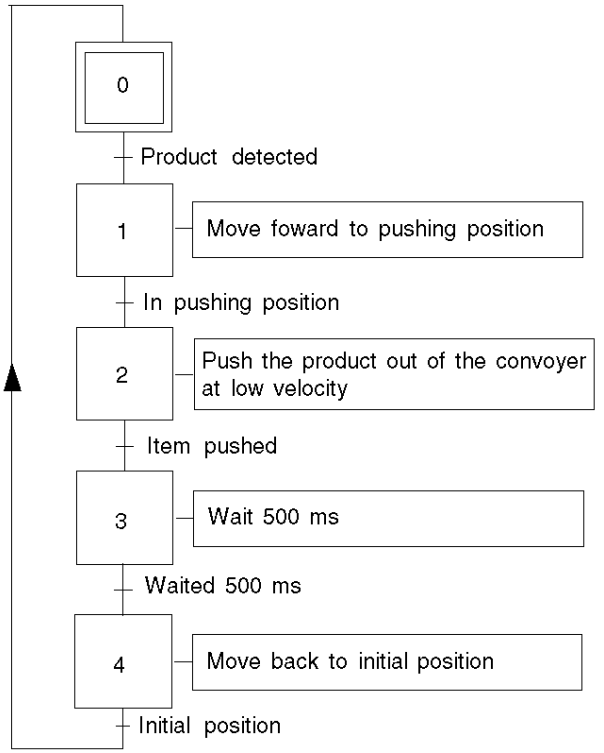
1 Digital Jack
2 Conveyor with products transported
3 Presence Sensor

When the product is detected, there is a 4-step sequence which starts:

- The jack moves forward to pushing position, this is a high-speed aproach phase.
- The product is pushed out of the belt at lower velocity.
- After pushing the item, there is a 500 ms break before moving the jack again.
- After waiting, the jack goes back into its original position.

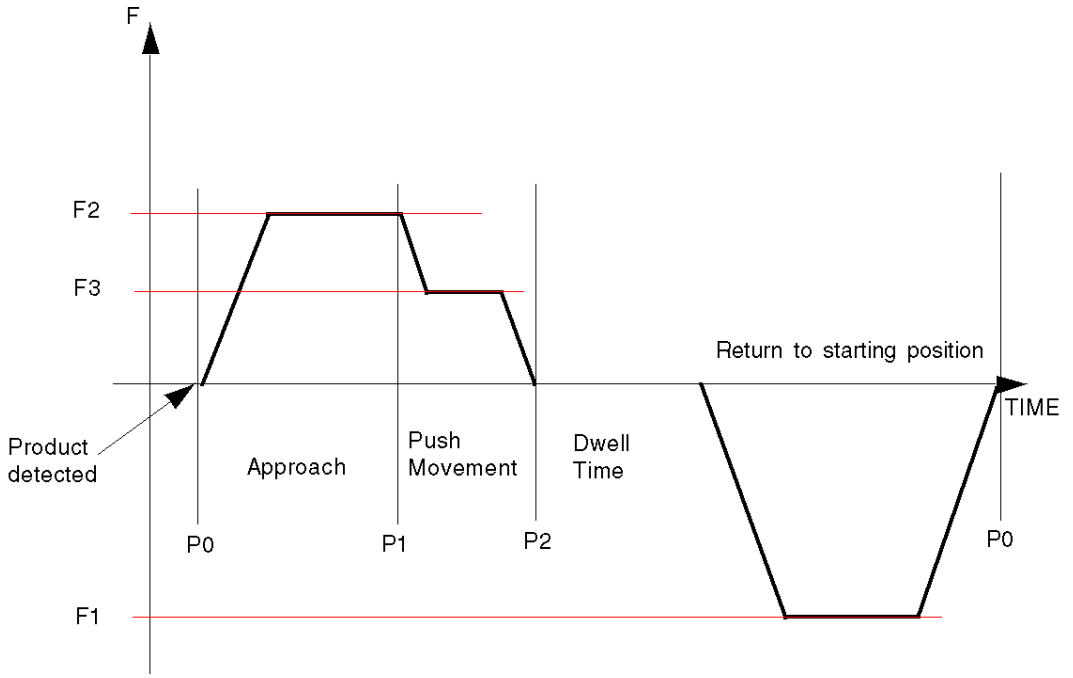
Sequence Diagram

The sequence can be represented by the following diagram.



Velocity Diagram

The jack's speed will be like the following diagram:



F Frequency
P Position

Hardware installation

6

Overview

This chapter concerns the hardware installation, mounting, wiring and configuration of the Lexium 05.

What's in this Chapter?

This chapter contains the following topics:

Mounting the module and the terminal

At a Glance

This part is fully described in the module installation. *(see page 19)*

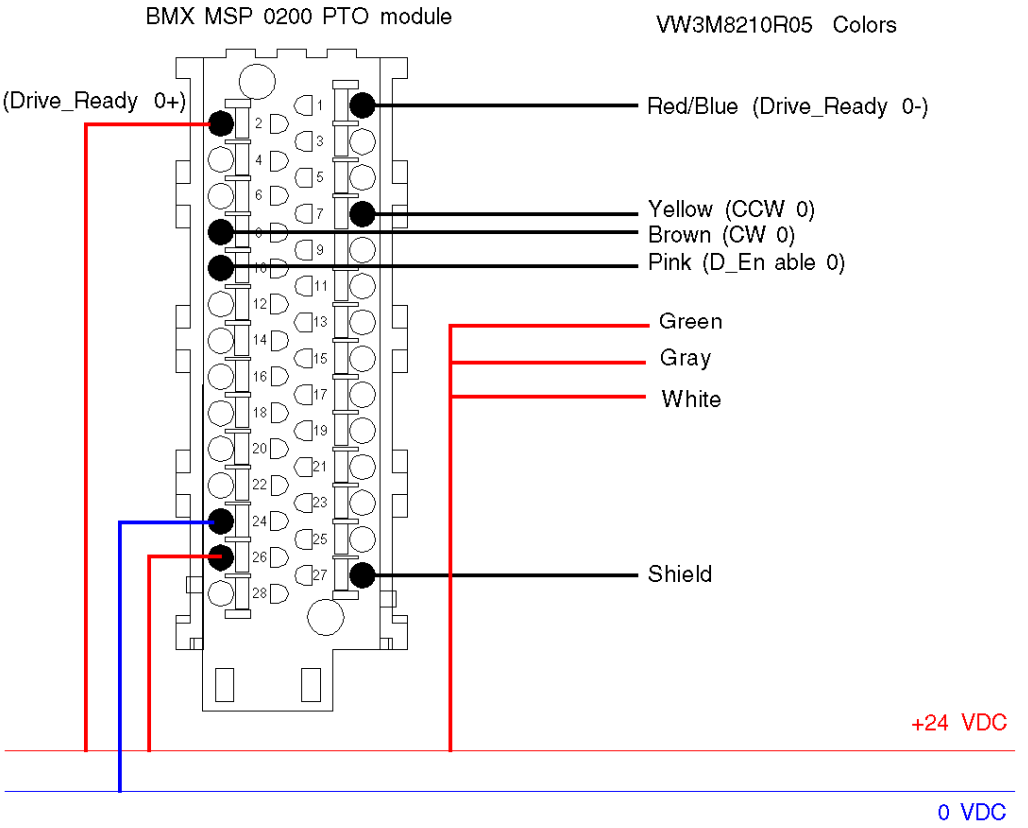
Wiring the PTO module to the LEXIUM 05 via the USIC

At a Glance

It is necessary to use a USIC, an external RS422 accessory to connect the Lexium 05 drive to the PTO channel as the drive cannot be connected directly.

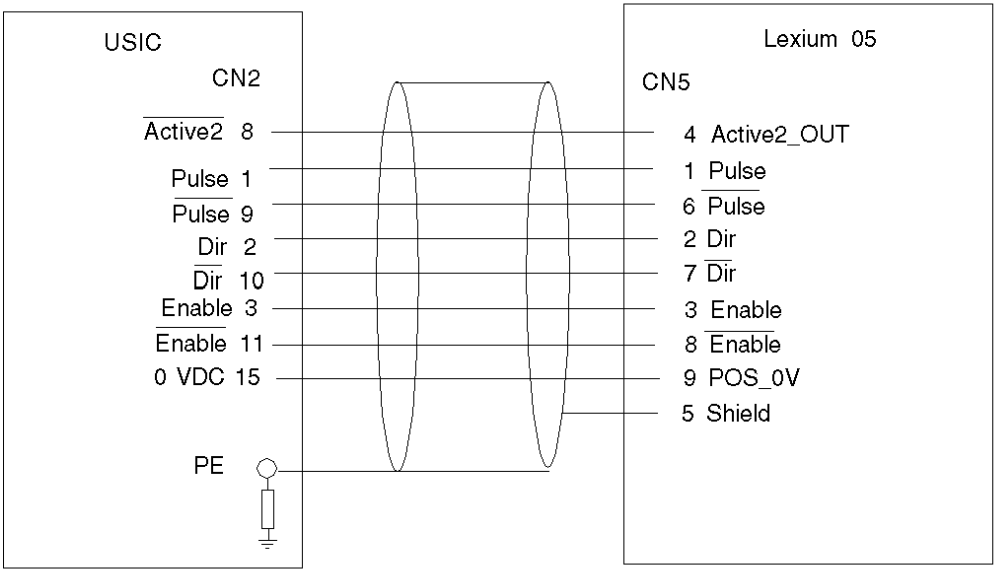
Wiring PTO Module to USIC

For this diagram, it is considered the PTO channel 0 is configured. A reference: VW3M8210R05 cable is required for this wiring.



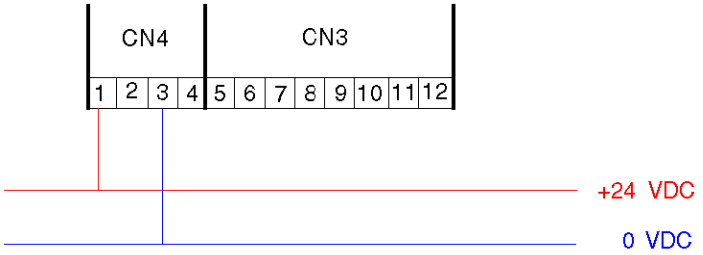
Wiring USIC to Lexium 05

This wiring can be done by using the prefabricated cable reference: VW3M8209R30 (or 05, 15, 50)



Wiring Usic

The CN4 and CN3 USIC pins need to be wired as shown:



Configuring the Lexium 05 in PowerSuite

Overview

PowerSuite allows to configure a drive.

PowerSuite gives access to all the configurable elements of the Lexium 05 as well as a monitoring and simulation element. Once configured, the software creates a configuration file which can be saved on the Lexium 05.

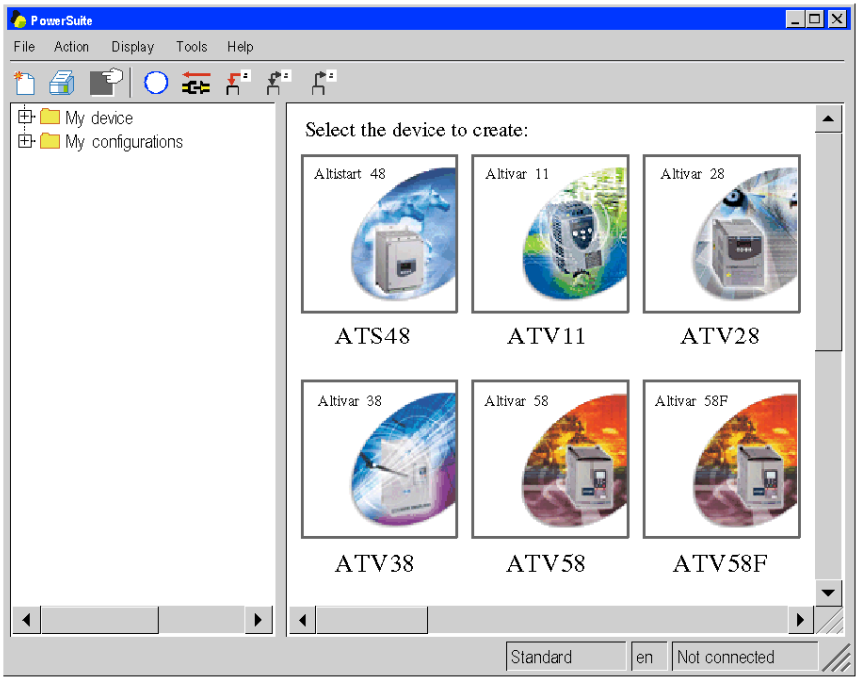
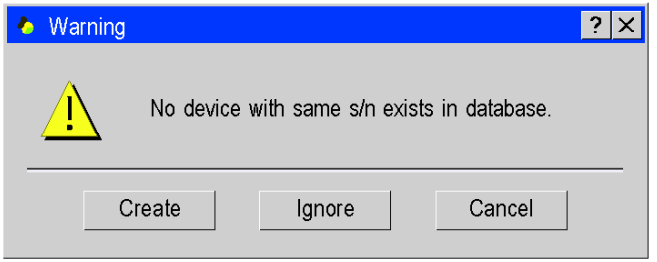
In this part, the following elements are needed:

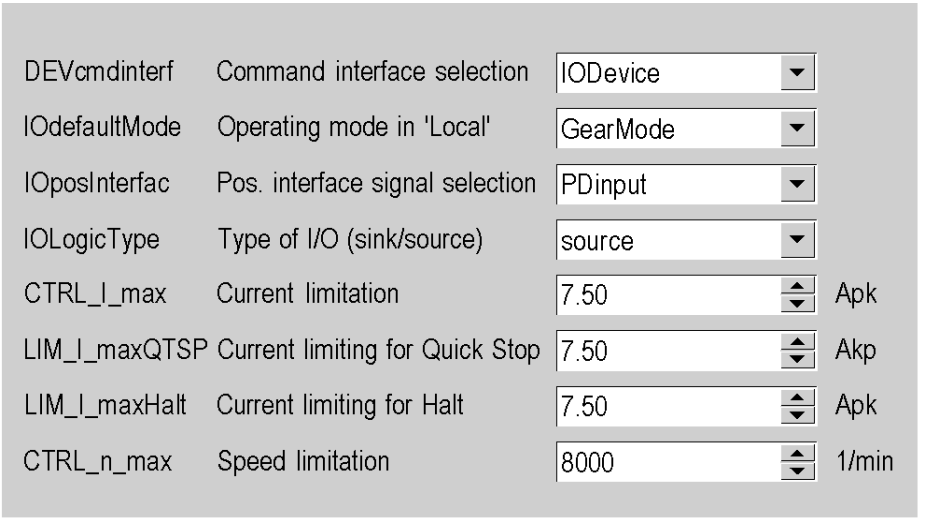
- PowerSuite 2.5
- Network cable (RJ45)
- A RS232/RS485 accessory (ref: W814944430221)

NOTE: Required signals LIMN, LIMP and REF must be wired or deactivated by the tuning software.

Connecting and Configuring the Lexium 05

This table describes the procedure for connecting to the **Lexium 05**:

Step	Action
1	Connect the PC with PowerSuite to the **Lexium 05** with the **RJ45** and the **RS232/RS485 accessory** to the servodrive.
2	Start PowerSuite 2.5, **Result:** the following start-up screen is displayed: 
3	Right click on `My Devices` and then `Connect`. **Result:** a text box is displayed  Press `Create`.

Step	Action
4	Type a project name (Lexium05_PTO) and then click on **OK**. **Result:** a transfer confirmation window is displayed.
5	The Lexium 05 configuration is transferred from the servodrive to the connected work station.
6	PowerSuite displays a configuration screen in a new window that gives access to device control, tuning and monitoring functions. Select `Basic Configuration` in the `Simply Start` section. **Result:** a window with factory settings will be displayed. Set these settings as followed: 
7	Click on the `Configuration` menu, then `Save to EEPROM` and validate by clicking on `OK` to save the configuration to the Lexium 05
8	Turn power off and back on to reboot the Lexium 05. If the Lexium 05 is configured properly, it will display **rdy**

Configuring the Lexium 05 with the User Interface

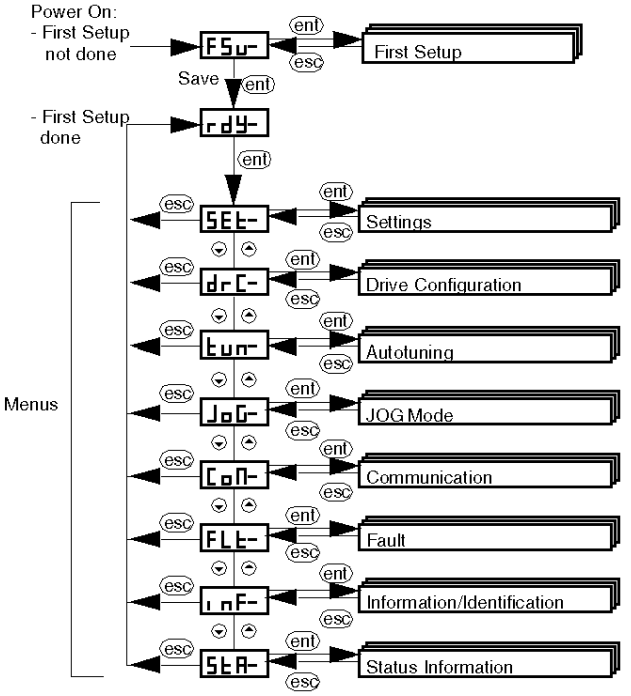
Overview

A user interface is integrated in the **Lexium 05**. With this interface, you can:

- put the device online
- configure the device
- carry out a diagnostic

Interface Menu Structure

The following graphic presents an overview of access to the interface's main menus:



Basic Settings

The following table describes the procedure for entering the settings for our application.

Step	Action
1	If the HMI has **FSu-** displayed, then the first setup needs to be done, refer to the Lexium 05 Simplified manual (id: 1760970) in order to do this.
2	The HMI displays **rdy** Press the **ENT** button on the interface. **Result:** the **SET** (Setting) menu is displayed on the interface's status indicator.
3	Press **ENT** Press  or  and select **iMAH**, validate with **ENT**. Set the value to 7.50 with the  or  Press **ENT** Press **ESC**
4	Press  or  and select **Li95**, validate with **ENT**. Set the value to 7.50 with the  or  Press **ENT** Press **ESC**
5	Press  or  and select **LihA**, validate with **ENT**. Set the value to 7.50 with the  or  Press **ENT** Press **ESC** twice
6	Press the  button several times to access the **drC-** menu and press Press **ENT**. **Result:** the **A2Mo** menu is displayed on the interface's status indicator.
7	Press the  button several times to access the **io-M** menu and press Press **ENT**.
8	Press  or  and select **GEAr**, validate with **ENT**. (If the previous configuration wasn't gear, then it will blink once to validate the change). Press **ESC**
9	Press  select **ioPi**, validate with **ENT**.

Step	Action
10	Press  or  and select **Pd**, validate with **ENT**. (If the previous configuration wasn't Pd, then it will blink once to validate the change). Press **ESC** twice to return to the **drC-** menu
11	Press **ESC** to return to the main display (**RDY** by default).

Configuring the BMX MSP 0200 on Unity Pro

7

Overview

This chapter describes the different steps to configure the module on Unity Pro.

What's in this Chapter?

This chapter contains the following topics:

Creating the Project

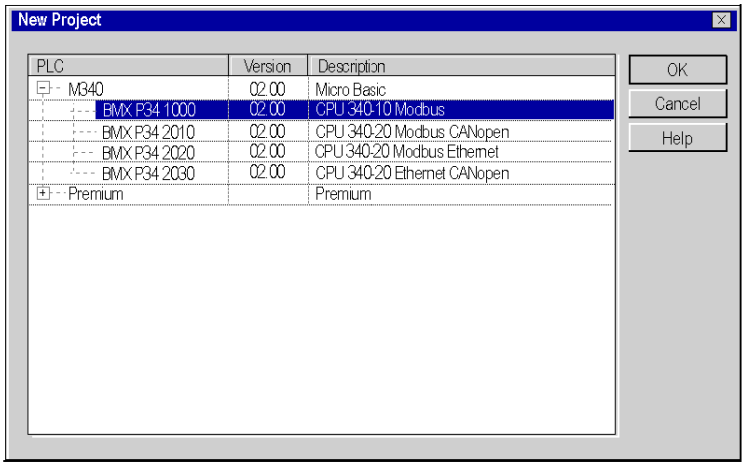
At a Glance

Developing an application using Unity Pro involves creating a project associated with a PLC.

NOTE: For more information, see Unity Pro online help (click on `?`, select `Content` tab; click on `Unity`, then `Unity Pro Software`, then `Operating modes`, and `Project configuration`).

Procedure for Creating a Project

The table below shows the procedure for creating the project using Unity Pro.

Step	Action
1	Launch the Unity Pro software.
2	Click on File then New, the new project window will appear.
3	Select a M340 PLC. 
4	Confirm with OK.

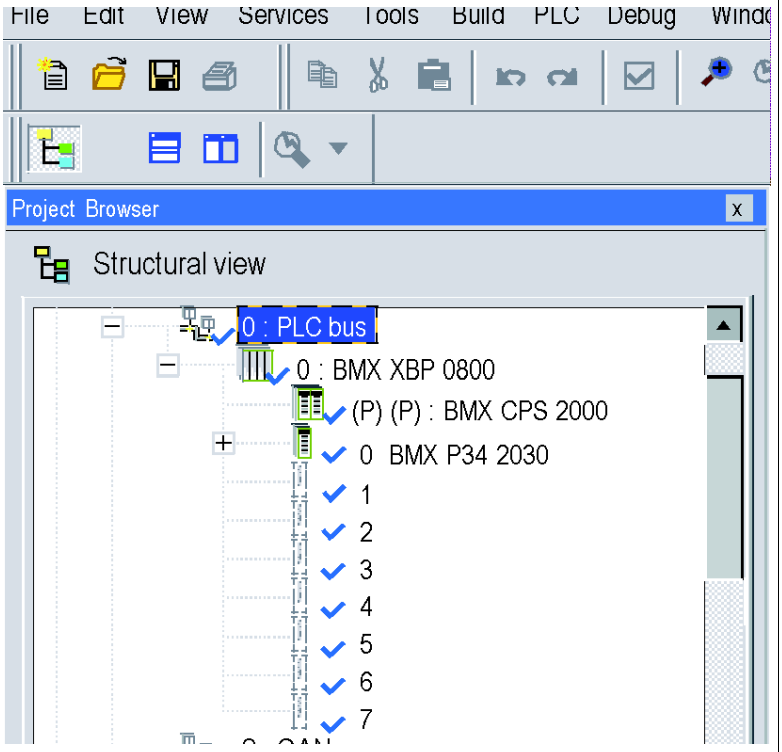
Configuring the BMX MSP 0200 PTO Module

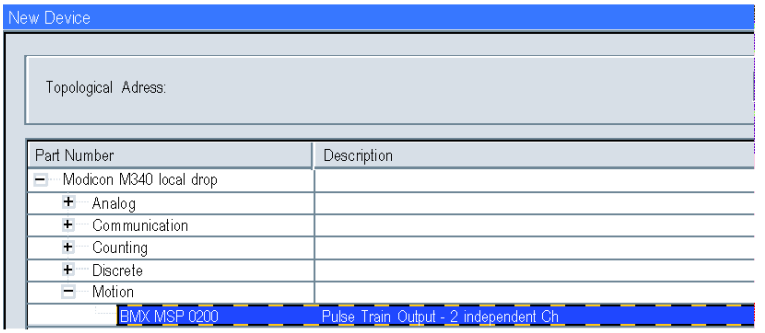
At a Glance

Developing an application with a PTO module involves choosing the right module and appropriate configuration.

Module Selection

The table below shows the procedure for selecting the pulse train output module.

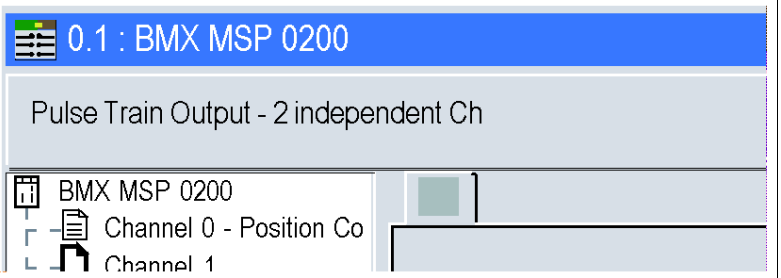
Step	Action
1	In the `Project browser` double-click on `Configuration` then on `0:PLC Bus` and on `0:BMX XBP •••` (Where 0 is the rack number) 
2	In the `PLC Bus` window, select slot 1 and double-click

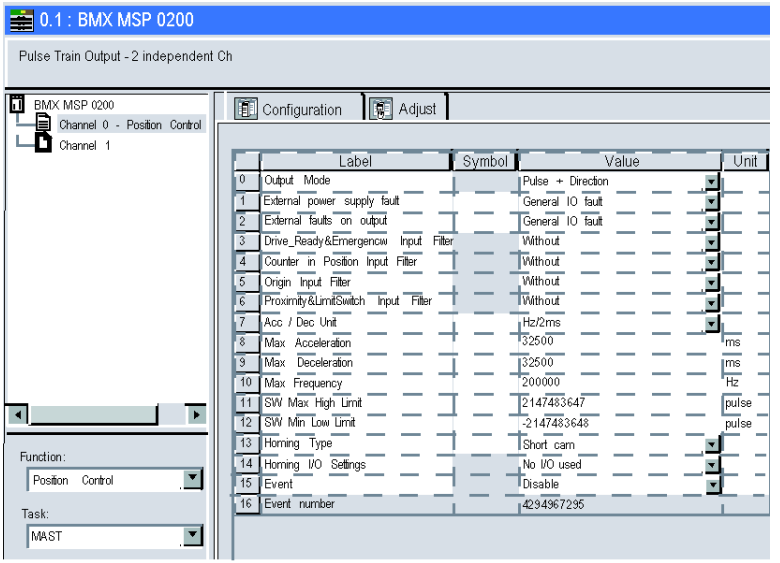
Step	Action
3	Choose the `BMX MSP 0200` Pulse Train Output module 
4	Confirm with OK.

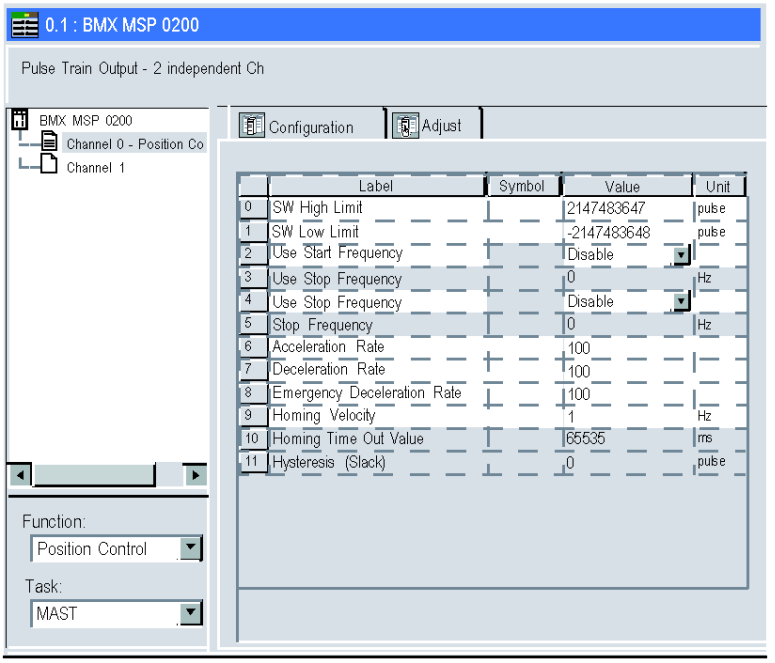
PTO Module Configuration

The table below shows the procedure for selecting the pulse train output module and configuring the module reflex outputs.

Step	Action
1	In the `PLC Bus` window, double-click on the `BMX MSP 0200` Pulse Train Output module
2	Select channel 0

Step	Action
3	Select the module function `Position Control` 

Step	Action
4	In the configuration screen set Acc/Dec Unit to Hz/2ms. 

Step	Action
5	At this stage the adjustment parameter remain unchanged. 

Programming a Movement

8

Overview

This chapter describes how to create a movement profile on Unity Pro.

What's in this Chapter?

This chapter contains the following topics:

Declaration of Variables

At a Glance

All of the variables used in the different sections of the program must be declared. Undeclared variables cannot be used in the program.

The following table shows the details of the variables used in the application.

Variable	Type	Definition
Elementary Variables		
Abort	BYTE	BufferMode parameter (value = 0)
ApproachInProgress	BOOL	Approach in progress
BlendingPrevious	BYTE	BufferMode parameter (value = 2)
Buffered	BYTE	BufferMode parameter (value = 1)
BufferFree	BOOL	
Cmd0Nb	BYTE	1st command output number
Cmd1Nb	BYTE	2nd command output number
Cmd2Nb	BYTE	3rd command output number
Cmd3Nb	BYTE	4th command output number
InitProcess	BOOL	Process initialisation
ItemToSort	BOOL	Item to sort detection
Derived Variables		
Approach_Result	Result	Array with approach status
Pushing_Result	Result	Array with pushing status
SortingOperation_Result	Result	Array with sorting operation status
IO Derived Variables		
R1CH0	IODDT	IODDT of type T_PTO_BMX for the %CH0.1.0 address.

Declaring Elementary Variables

Overview

The first variables to declare are the elementary variables.

Procedure for Declaring Variables

The table below shows the procedure for declaring application variables *(see Unity Pro, Operating Modes,)*.

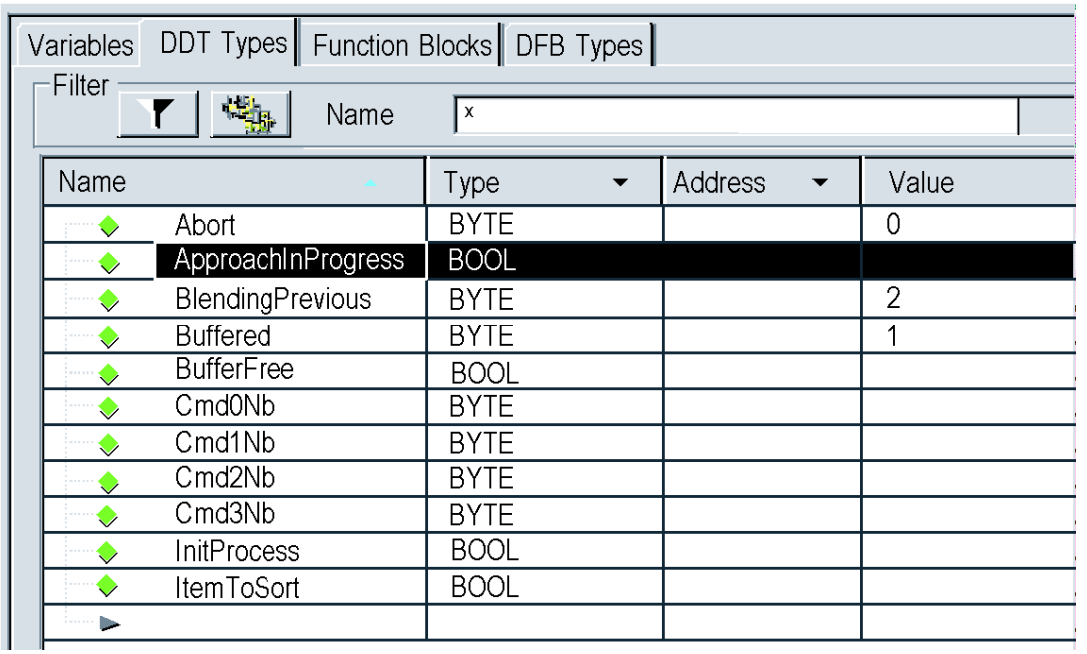
Step	Action
1	In `Project browser / Variables & FB instances`, double-click on `Elementary variables`
2	In the `Data editor` window, select the box in the Name column and enter a name for your first variable.
3	Now select a Type for this variable.
4	Declare all the variables as said then close the window.

Elementary Variables Used for the Application

The following table shows the details of the elementary variables used in the application.

Variable	Type	Definition
Abort	BYTE	BufferMode parameter (value = 0)
ApproachInProgress	BOOL	Approach in progress
BlendingPrevious	BYTE	BufferMode parameter (value = 2)
Buffered	BYTE	BufferMode parameter (value = 1)
BufferFree	BOOL	
Cmd0Nb	BYTE	1st command output number
Cmd1Nb	BYTE	2nd command output number
Cmd2Nb	BYTE	3rd command output number
Cmd3Nb	BYTE	4th command output number
InitProcess	BOOL	Process initialisation
ItemToSort	BOOL	Item to sort detection

The following screen shows the application variables created using the data editor:



Declaring Derived Variables

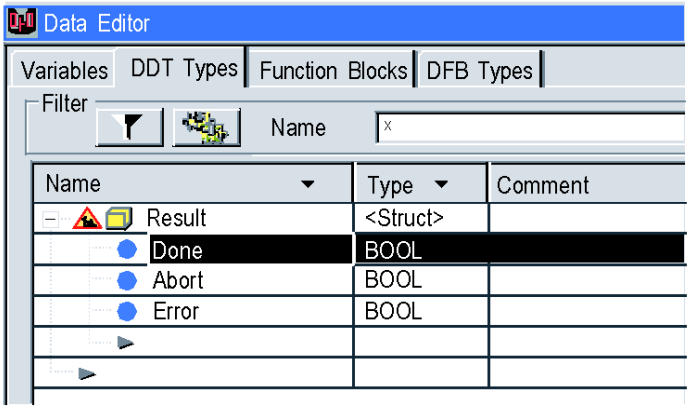
Overview

This is a 2-step procedure

1. Create the derived data type
2. Create the derived variables

Creating the Result Type

In order to create the derived variables, the Result type needs to be created. Follow these steps to do so:

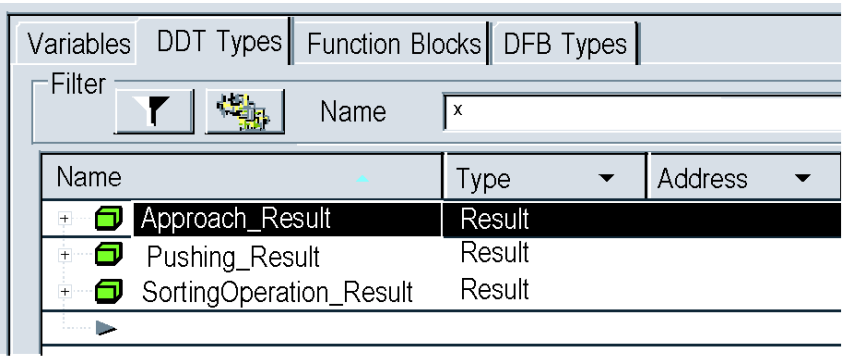
Step	Action
1	In `Project browser / Derived Data Types`, double-click on the folder to open the window.
2	Type "Result" in the name, and keep `Struct` type. A new Result data type will be in a creation (illustrated by the worker icon)
3	Expand the structure and add the elements (Done, Abort, Error). 
4	The worker icon will disappear if the analyze type command is used or next time the application is built.

Create the Derived Variables Used for the Application

The table shows the details of the Derived variables used in the application.

Variable	Type	Definition
Approach_Result	Result	Array with approach status
Pushing_Result	Result	Array with pushing status
SortIngOperation_Result	Result	Array with sorting operation status

The screen shows the application variables created using the data editor :



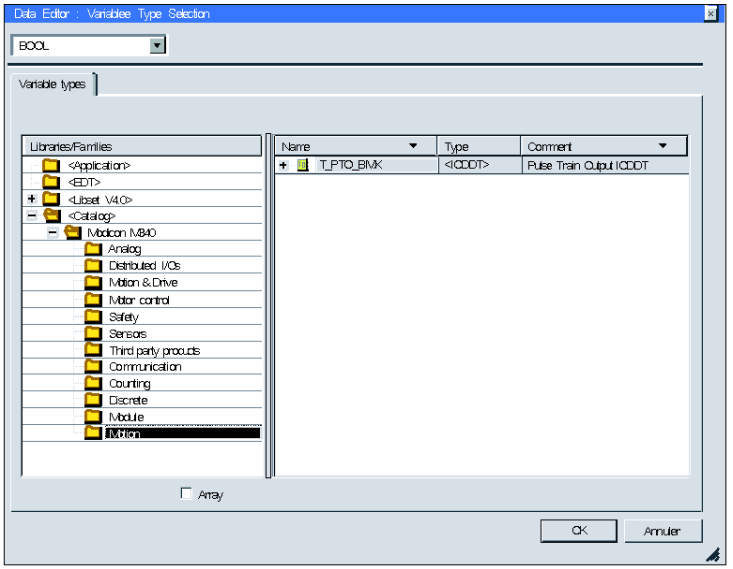
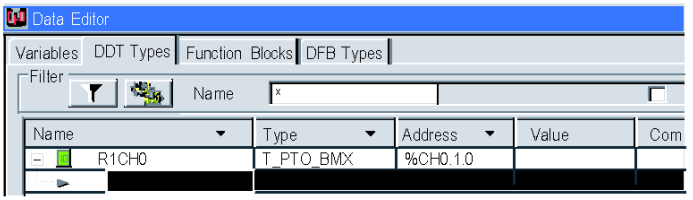
NOTE: Click on ⊞ in front of the derived variable **Approach_Result** to expand the I/O objects list.

Declaring IODDT Variables

Overview

The final step is to declare the IODDT type variable.

IODDT Used for the Application

Step	Action
1	In `Project browser / IO Derived Variable.`
2	In the `Data editor` window, select the box in the Name column and enter the R1CH0.
3	Select Type = T_PTO_BMX for this variable. You can find the type here:  Select it and click Ok
4	Specify the IODDT's address: %CH0.1.0 (Rack 1, PTO channel 0) 

Programming the Example

At a Glance

Just after declaration and parameter setting of the hardware, motion programming is the second development phase of the tutorial example.

Axis programming is divided in 4 steps according to the speed diagram:

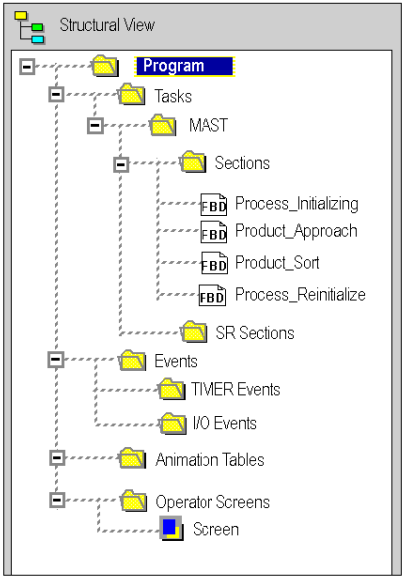
- Process initializing
- Approaching at high speed
- Sorting at low speed
- Waiting 500 ms and moving back to initial position

Declaring the Sections

The table below presents a summary of the program sections to create

Section name	Language	Description
Process_initializing *(see page 84)*	FBD	This section initializes the motion by referencing the axis.
Product_Approach *(see page 87)*	FBD	This section generates a movement at a high speed to a certain position close to the product.
Product_Sort *(see page 90)*	FBD	This section generates a low speed movement of the jack to sort the product.
Process_Reinitialize *(see page 92)*	FBD	This section generates a 500 ms pause and then places the jack back to initial position.

The diagram below shows the program structure after the programming sections have been created:



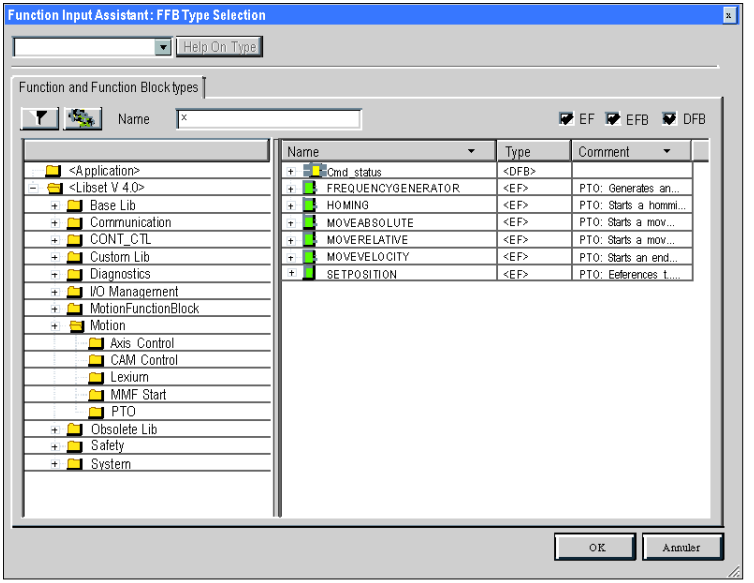
Process Initializing

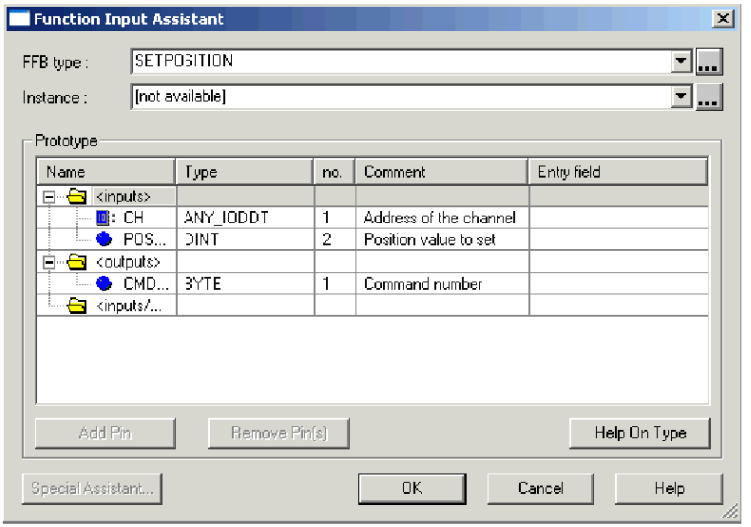
At a glance

This part of the program initializes the axis and references it *(see page 185)*.

Inserting a Block

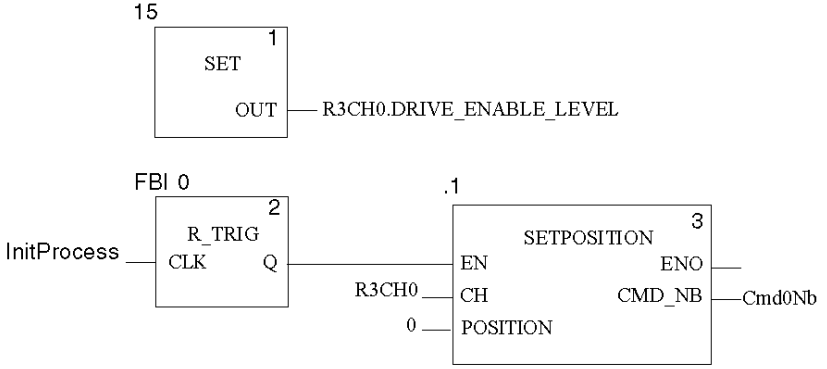
This table describes the procedure for inserting a block in a program section:

Step	Action
1	Right click in an empty field in the FBD section to display the contextual menu.
2	Execute the **FFB Input Assistant..** command in the contextual menu. **Result**: The Function Input Assistant opens.
3	Click on the **...** icon on the **FFB Type** line. **Result**: the **FFB Type Selection** window opens.
4	Expand **Libset V4.0** →**Motion** and click on **PTO**. **Result**: all of the blocks from the **PTO** library are displayed on the right-hand side of the **FFB Type Selection** window. 

Step	Action
5	Confirm the block configuration by clicking on **OK**.  **Result**: the FBD section is displayed again. A symbol is added to the mouse cursor.
6	Click on an empty field in the FBD section. **Result**: the `SETPOSITION` block is inserted in the FBD section.
7	Specify the input and output parameters as defined in the contents.
8	Repeat operation to add the `R_TRIG` block, knowing that it can be found in **Libset V4.0** →**Base Lib** →**Logic** and click on **R_TRIG**

Program

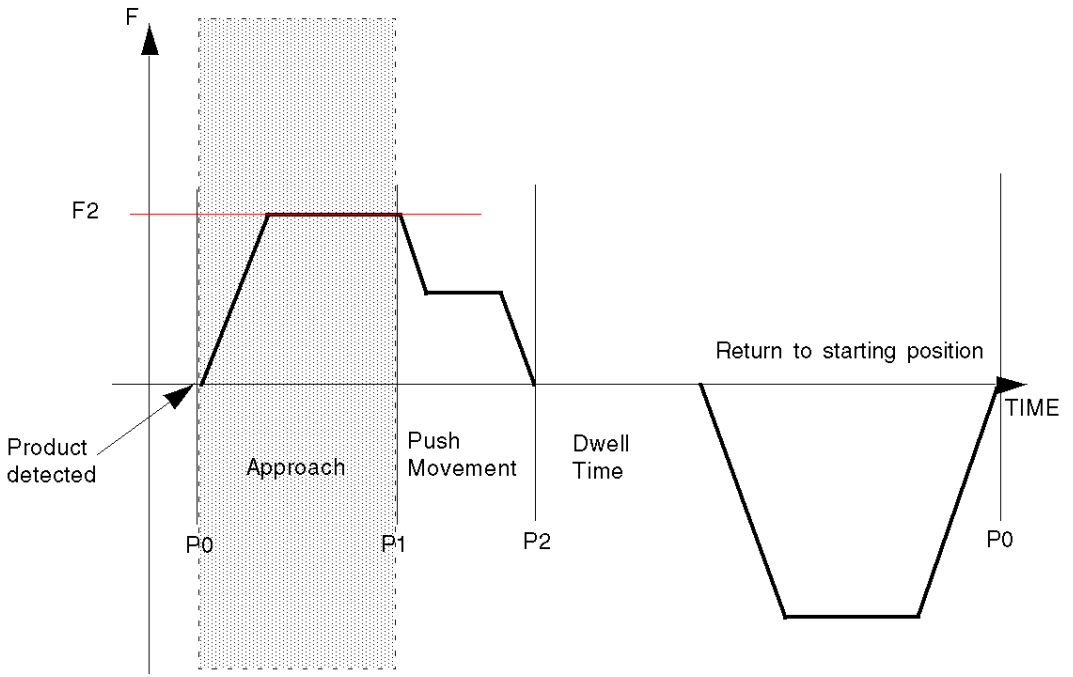
In process initializing section of the example, it is necessary to set D_Enable0 output to 1 either by using the IODDT (DRIVE_ENABLE_LEVEL) or with a program as shown:



Approach

At a glance

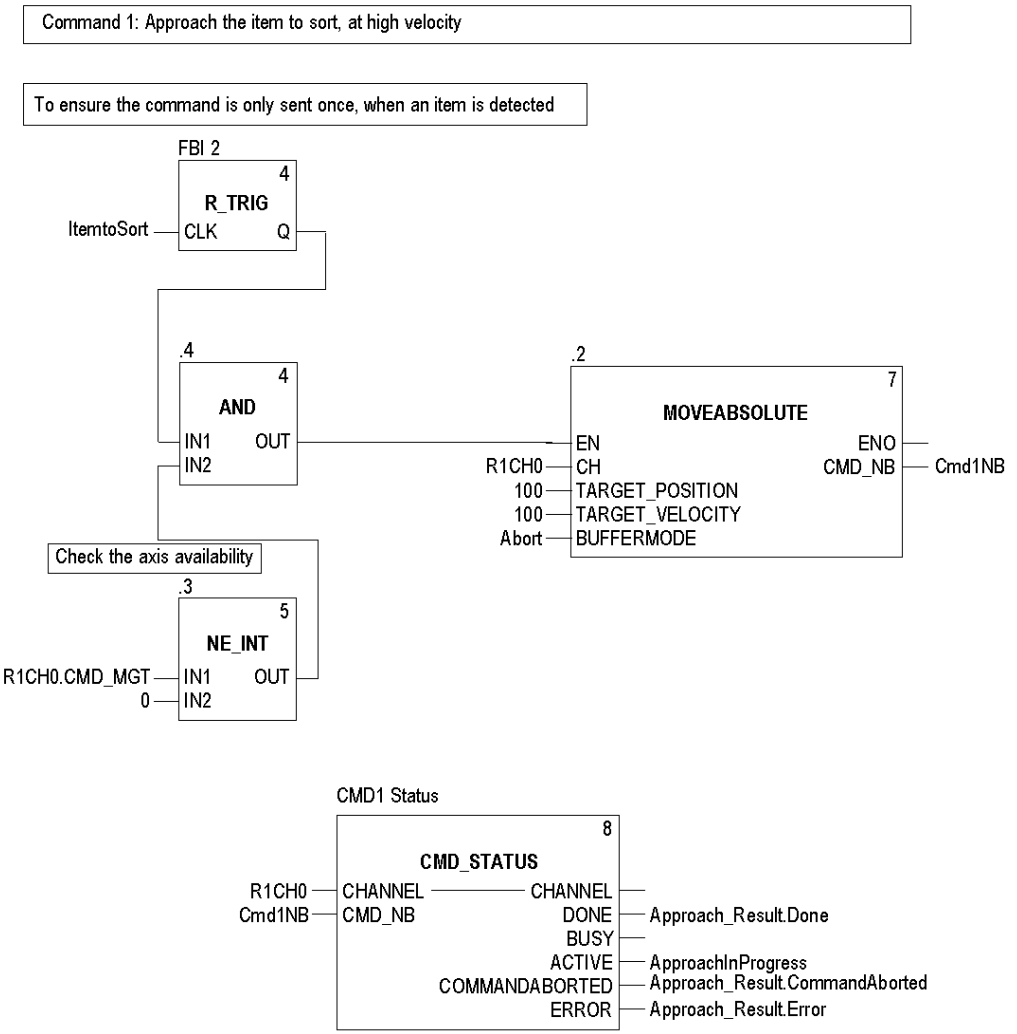
This part of the program is the high speed approach of the product part.



Program

Using the same programming method as in Process Initializing. *(see page 84)*

Command 1: Approach the item to sort at high velocity.



NOTE: TARGET_VELOCITY value is obtained by the following equation: Nb pulses x Gear x 60 / 131072.

To know the Lexium 05 drive movement angle in degree regarding the position degree = Nb Pulses x ratio x 360 (1 turn) / 131072

To know the Lexium 05 drive movement speed regarding the Drive's Velocity Frequency = Frequency Value x ratio x 60 / 131072

Fmax x Ratio = 131072 x Vmax / 60 so the Ratio (Gear) = 131072 x Vmax / 60 x Fmax

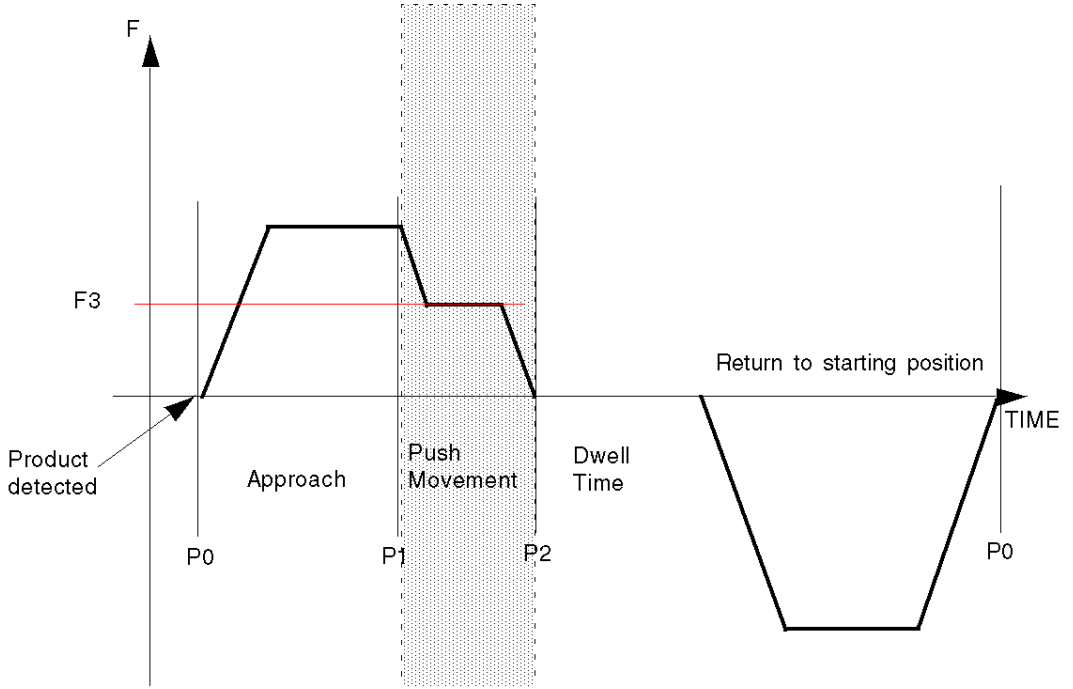
(Fmax (e.g. 200 kHz) must correspond to the drive's Vmax (e.g. 3500 rpm)

Since gear hasn't been modified in our Lexium 05 configuration, it has the default value of 1. This value can be modified with PowerSuite or on the HMI.

Sorting the Product

At a Glance

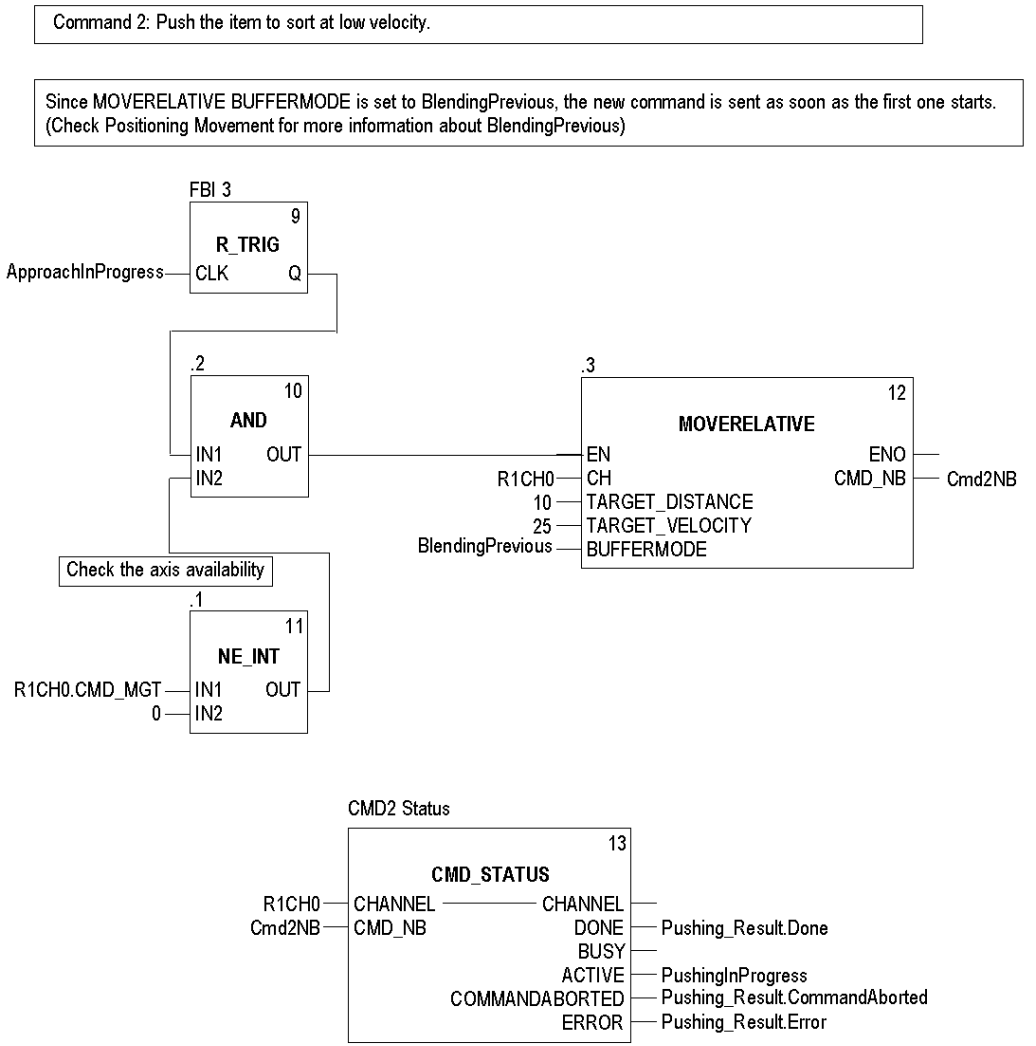
This part of the program is the low speed sorting of the product part.



Program

Using the same programming method as in Process Initializing *(see page 84)*

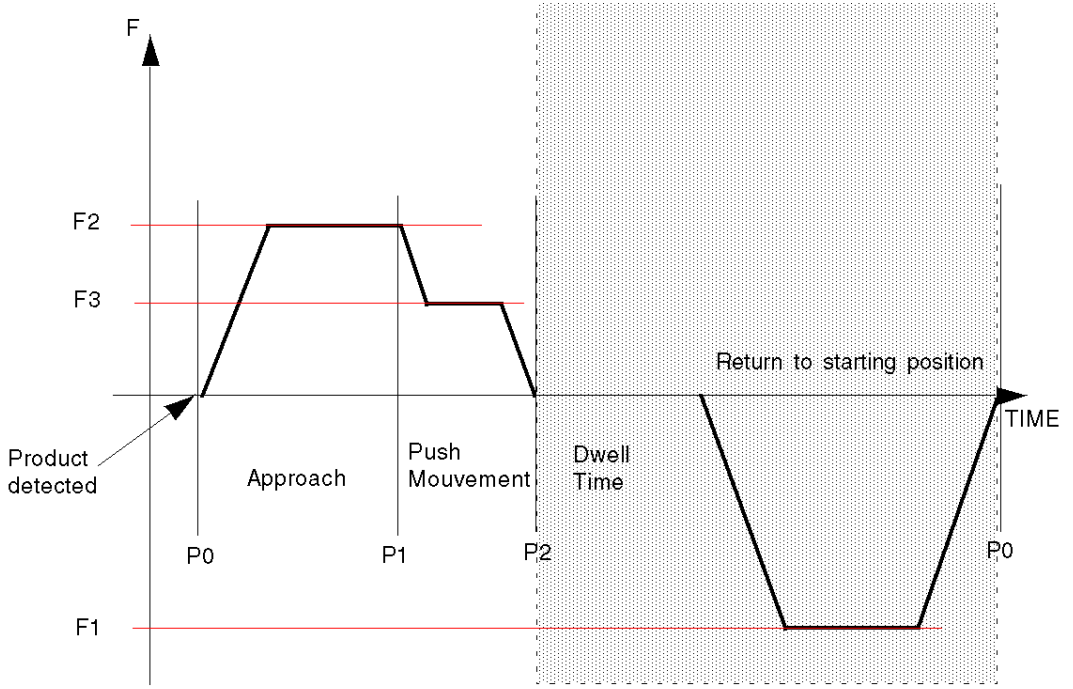
Command 2: Push the item to sort at low velocity.



Temporisation and Position Reinitialization

At a Glance

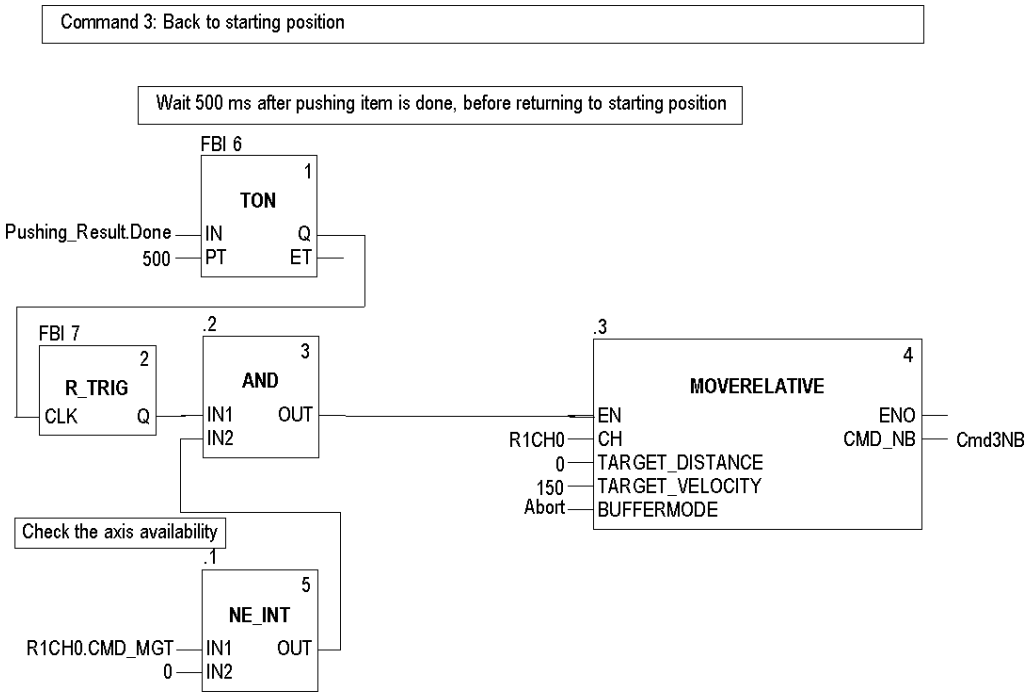
This part of the program is the dwell time and move back movement.



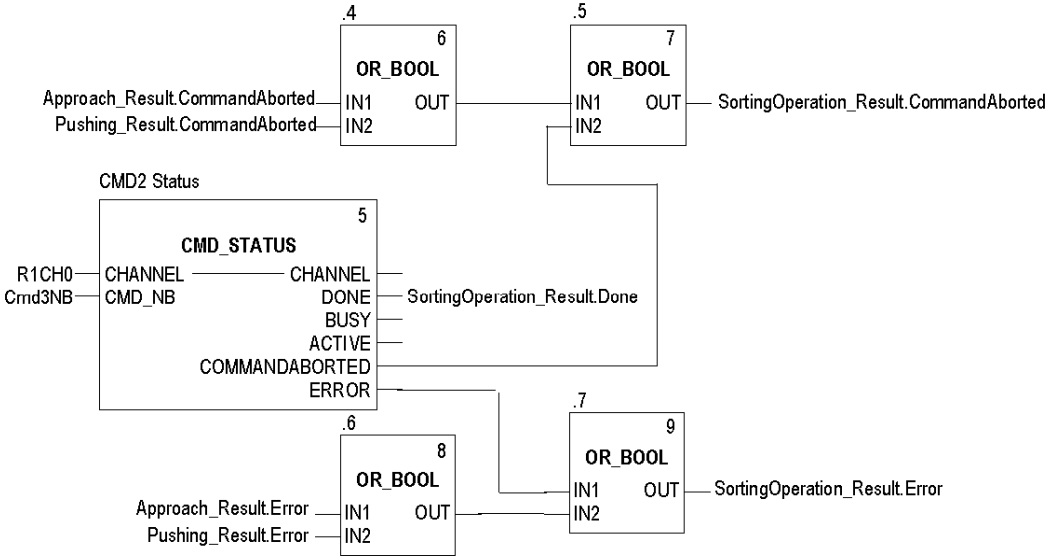
Program

Using the same programming method as in Process Initializing. *(see page 84)*

Command 3: Back to starting position.



This part of the program checks the overall sorting operation result.


.4
6
OR_BOOL
Approach_Result.CommandAborted
Pushing_Result.CommandAborted
IN1
IN2
OUT
.5
7
OR_BOOL
IN1
IN2
OUT
SortingOperation_Result.CommandAborted
CMD2 Status
5
CMD_STATUS
R1CH0
Cmd3NB
CHANNEL
CMD_NB
CHANNEL
DONE
BUSY
ACTIVE
COMMANDABORTED
ERROR
SortingOperation_Result.Done
.6
8
OR_BOOL
Approach_Result.Error
Pushing_Result.Error
IN1
IN2
OUT
.7
9
OR_BOOL
IN1
IN2
OUT
SortingOperation_Result.Error

Transferring the Project between the Terminal and the PLC

At a Glance

Transferring a project allows you to copy the current project from the terminal to the current PLC's memory (PLC that has its address selected).

Project Analysis and Generation

To perform analysis and generation of a project at the same time, carry out the following actions:

Step	Action
1	Activate the **Rebuild All Project** command in the **Build** menu. **Result:** the project is analyzed and generated by the software.
2	Any errors detected are displayed in the information window at the bottom of your screen.

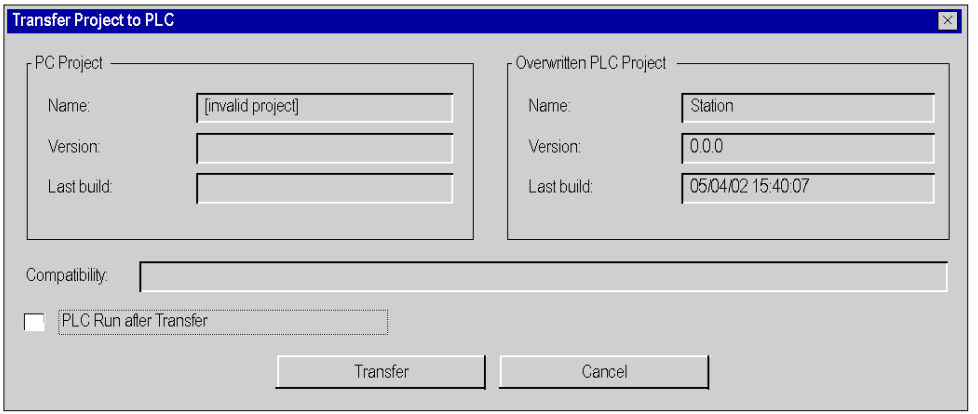
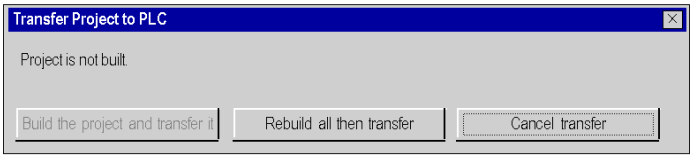
Project Backup

To back up the project, carry out the following actions:

Step	Action
1	Activate the **Save As** command in the **File** menu.
2	If necessary, select the directory to which the project will be saved (disk and path).
3	Enter the file name: **PTO_JackExample**.
4	Confirm with **Save**. **Result:** the project is saved as **PTO_JackExample.STU**.

Transferring the Project to the PLC

You must carry out the following actions to transfer the current project to a PLC:

Step	Action
1	Use the **PLC →Define the address** command. Enter **SYS** if you are using a **USB** media that is directly connected from the PC (terminal) to the PLC.
2	Switch to online mode using the **PLC →Connection** command.
3	Activate the **PLC →Transfer Project to PLC** command. **Result**: the screen used to transfer the project between the terminal and the PLC is displayed: 
4	Activate the **Transfer** command.
5	If the project has not been generated in advance, the screen below will be displayed allowing you to generate it before the transfer (**Rebuild All then Transfer**) or interrupt the transfer (**Cancel Transfer**). 
6	Transfer progress is displayed on screen. At any moment, you can interrupt the transfer by using the **Esc** key. In this case, the PLC project will be invalid. **Note**: In the event that the project is transferred to a Flash Eprom memory card, the transfer can take several minutes.

Example Diagnostic and Debugging

9

Overview

This chapter describes available tools for diagnosing and debugging the application.

What's in this Chapter?

This chapter contains the following topics:

Using Data via the Animation Tables

At a Glance

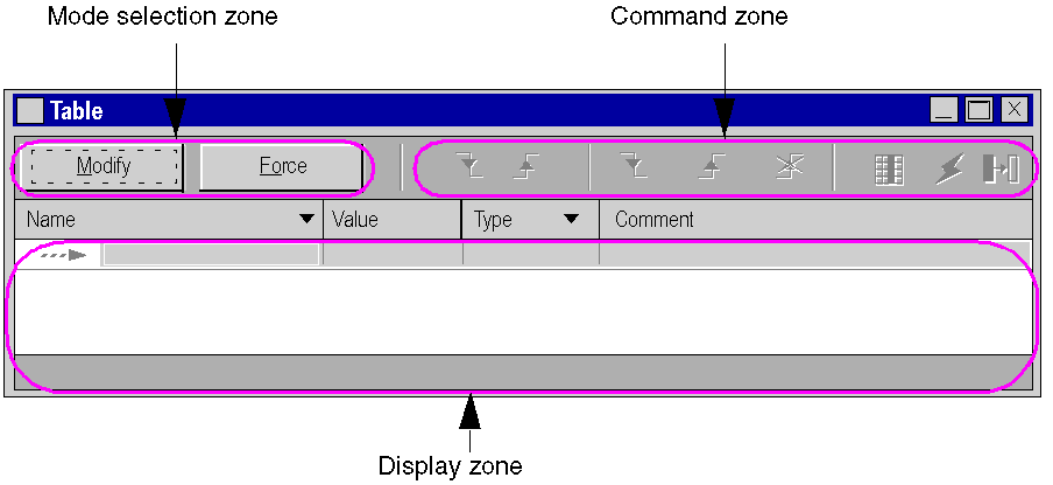
The animation table is the Unity Pro' basic tool for viewing and forcing the status of variables.

NOTE: Unity Pro also offers a graphic tool called **Operator Screens** which is designed to facilitate use of the application. *(see MFB for Modicon M340 using Unity Pro, Start-up Guide)*

An animation table is divided into 3 areas that include:

- the **Mode** area
- the **Command** area
- the **Display** area

Animation table:



Creating an Animation Table

The table below presents the procedure for creating an animation table:

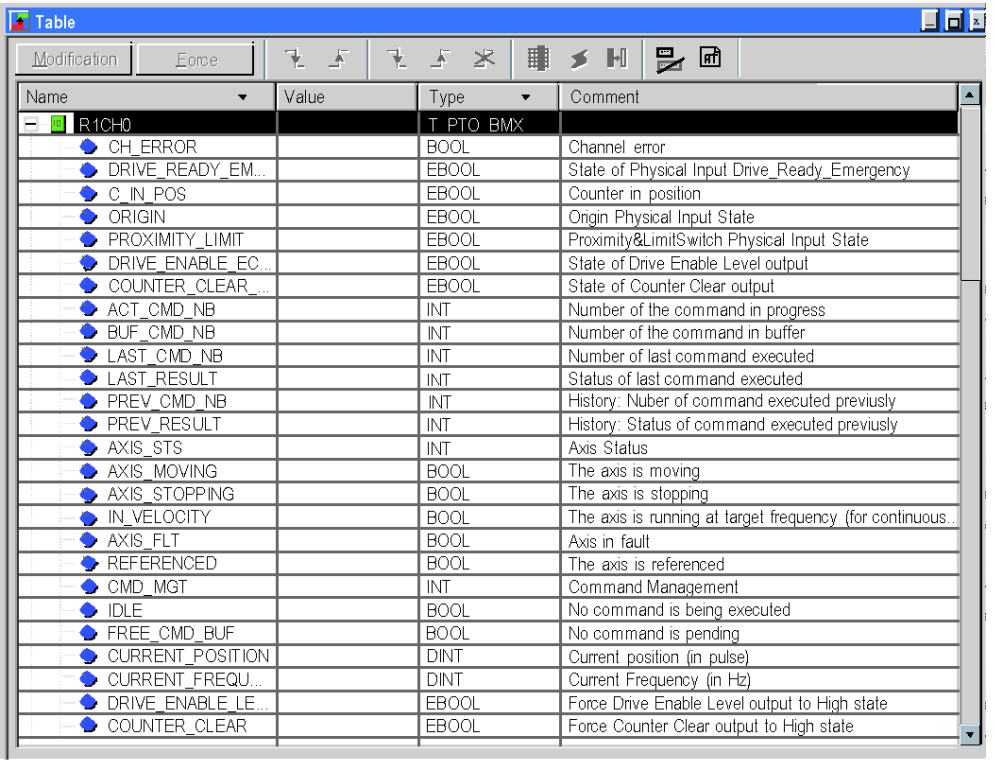
Step	Action
1	Right-click on the **Animation Tables** directory in the project browser. **Result**: the contextual menu is displayed.
2	Select **New Animation Table**. **Result**: a table properties window is displayed.
3	Click on OK to create the table, which is given a default name. **Result**: the animation table is displayed.

Adding Data to the Animation Table

The table below presents the procedure for creating data to view or force in the animation table:

Step	Action
1	In the **Table** window, click on the empty line in the **Name** column.

Step	Action
2	There are two possible ways of adding data: • Enter the variable name directly • Click on the … icon to display the instance selection window in order to select the variable
3	Enter or select the R1CH0 variable and expand it. **Result**: the animation table looks like this.



Name	Value	Type	Comment
R1CH0		T_PTO_BMX	
CH_ERROR		BOOL	Channel error
DRIVE_READY_EM...		EBOOL	State of Physical Input Drive_Ready_Emergency
C_IN_POS		EBOOL	Counter in position
ORIGIN		EBOOL	Origin Physical Input State
PROXIMITY_LIMIT		EBOOL	Proximity&LimitSwitch Physical Input State
DRIVE_ENABLE_EC...		EBOOL	State of Drive Enable Level output
COUNTER_CLEAR_...		EBOOL	State of Counter Clear output
ACT_CMD_NB		INT	Number of the command in progress
BUF_CMD_NB		INT	Number of the command in buffer
LAST_CMD_NB		INT	Number of last command executed
LAST_RESULT		INT	Status of last command executed
PREV_CMD_NB		INT	History: Nuber of command executed previusly
PREV_RESULT		INT	History: Status of command executed previusly
AXIS_STS		INT	Axis Status
AXIS_MOVING		BOOL	The axis is moving
AXIS_STOPPING		BOOL	The axis is stopping
IN_VELOCITY		BOOL	The axis is running at target frequency (for continuous...
AXIS_FLT		BOOL	Axis in fault
REFERENCED		BOOL	The axis is referenced
CMD_MGT		INT	Command Management
IDLE		BOOL	No command is being executed
FREE_CMD_BUF		BOOL	No command is pending
CURRENT_POSITION		DINT	Current position (in pulse)
CURRENT_FREQU...		DINT	Current Frequency (in Hz)
DRIVE_ENABLE_LE...		EBOOL	Force Drive Enable Level output to High state
COUNTER_CLEAR		EBOOL	Force Counter Clear output to High state

Using Data via the Operator Screens

At a Glance

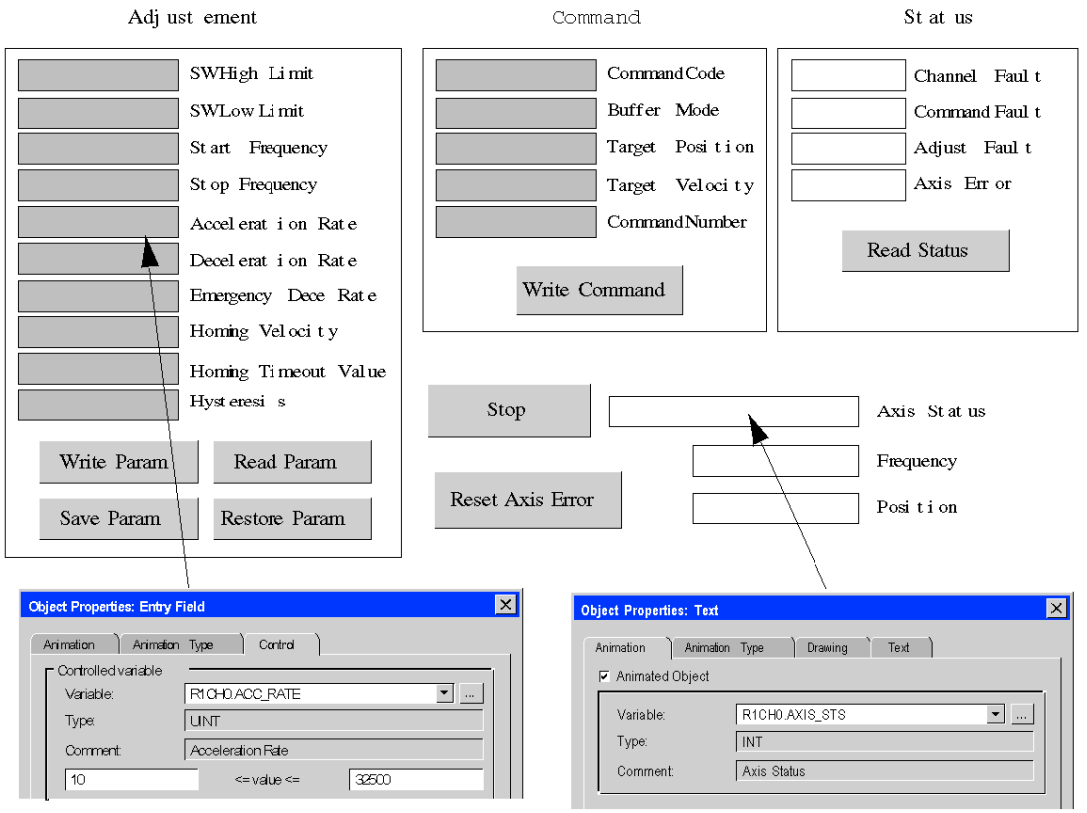
When a project is created, it is common no material is available so to lessen the impact of this problem, Unity Pro gives access to operator screen associated with unlocated bits and words allowing to carry out initial debugging of the program.

In this example, the operator screen is used to:

- View adjustment data
- Write new adjustment parameters
- Send a command
- View status data
- Stop the program
- Clear axis errors

Representation

The representation below symbolizes the operating example which is used to control the axis and indicates the variables to be assigned to the objects (push button, LED and text):



PTO Function

III

Overview

This part describes the features related to Unity Pro for the BMX MSP 0200 PTO module.

What's in this Part?

This part contains the following chapters:

Configuration parameters

10

Overview

This chapter deals with the parameters necessary for configuring the BMX MSP 0200.

What's in this Chapter?

This chapter contains the following topics:

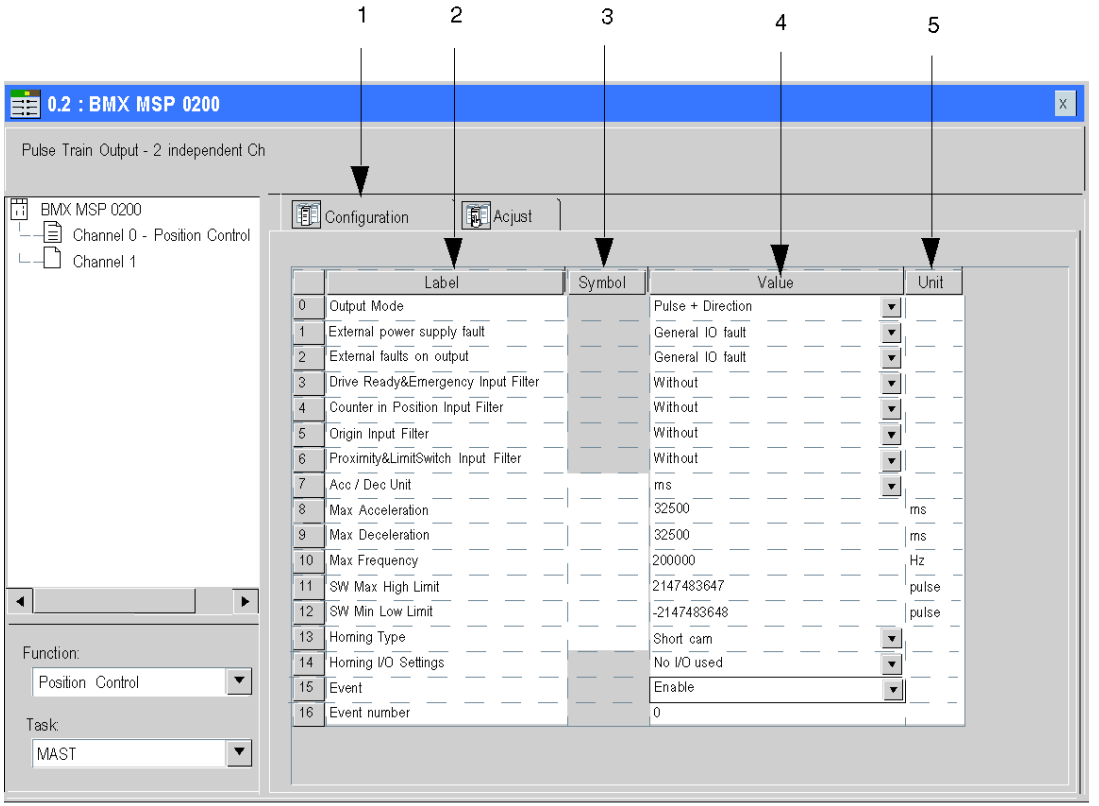
Configuration Screen for the BMX MSP 0200 PTO Module

At a Glance

This section presents the configuration screen for BMX MSP 0200 PTO Module

Illustration

The figure below presents the configuration screen of the BMX MSP 0200 PTO Module in pulse train output mode :



Description of the Screen

The following table presents the various parts of the above screen:

Number	Element	Function
1	Tab	The tab in the foreground indicates the current mode. The current mode is therefore the configuration mode in this example.
2	**Label** field	This field contains the name of each variable that may be configured. This field may not be modified.
3	**Symbol** field	This field contains the address of the variable in the application. This field may not be modified.
4	**Value** field	This field contains a drop-down menu containing all the possible values and the user may then select or directly write the required value of the variable.
5	**Unit** field	This field contains the unit of each variable that may be configured. This field may not be modified.

NOTE: Refer to the desired function *(see page 129)* in order to properly configure the BMX MSP 0200 PTO module

Position Control Mode Configuration

At a Glance

The configuration of a PTO module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topological addressing of the module. Each parameter has the following signification:

- r: represents the rack number,
- m: represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the position control mode configurable elements.

Number	Address in the configuration	Configurable values
Output Mode	%KWr.m.c.1(low byte)	• Pulse + Direction (default value) • CW/CCW • A/B Phases • Pulse + Direction - Reverse • CW/CCW - Reverse • A/B Phases - Reverse
Power Supply Fault	%KWr.m.c.1.8	• General I/O Fault (default) • Local
Output Fault	%KWr.m.c.1.9	• General I/O Fault (default) • Local
Drive Ready & Emergency Input Filter	%KWr.m.c.2(low byte)	• Without (default) • Low • Medium • High
Counter in position Input Filter	%KWr.m.c.2(high byte)	• Without (default) • Low • Medium • High
Origin Input Filter	%KWr.m.c.3(low byte)	• Without (default) • Low • Medium • High
Proximity&LimitSwitch Input Filter	%KWr.m.c.3(high byte)	• Without (default) • Low • Medium • High
Acc / Dec Unit	%KWr.m.c.1.12	• ms (default) • Hz/2ms
Max Acceleration	%KWr.m.c.4	10 to 32,500 (default value = 32,500)

Number	Address in the configuration	Configurable values
Max Deceleration	%KWr.m.c.5	10 to 32,500 (default value = 32,500)
Max Frequency	%KDr.m.c.6	0 to 200,000 (default value = 200,000)
SW Max High Limit	%KDr.m.c.8	-2,147,483,647 to 2,147,483,647 (default value = 2,147,483,647)
SW Min Low Limit	%KDr.m.c.10	-2,147,483,648 to 2,147,483,646 (default value = 2,147,483,648)
Homing Type	%KWr.m.c.12	• Short Cam (default) • Long Cam Positive • Long Cam Negative • Short Cam with Positive Limit • Short Cam with Negative Limit • Short Cam with Marker
Homing I/O Settings	%KWr.m.c.1.10-11	• No I/O used (default) • With Counter Clear Output • With Counter in Position Input
Event	%KWr.m.c.0 (high byte)	• Disable (default) • Enable
Event number	%KWr.m.c.0 (high byte)	Event Nb (Default: First free EVT)

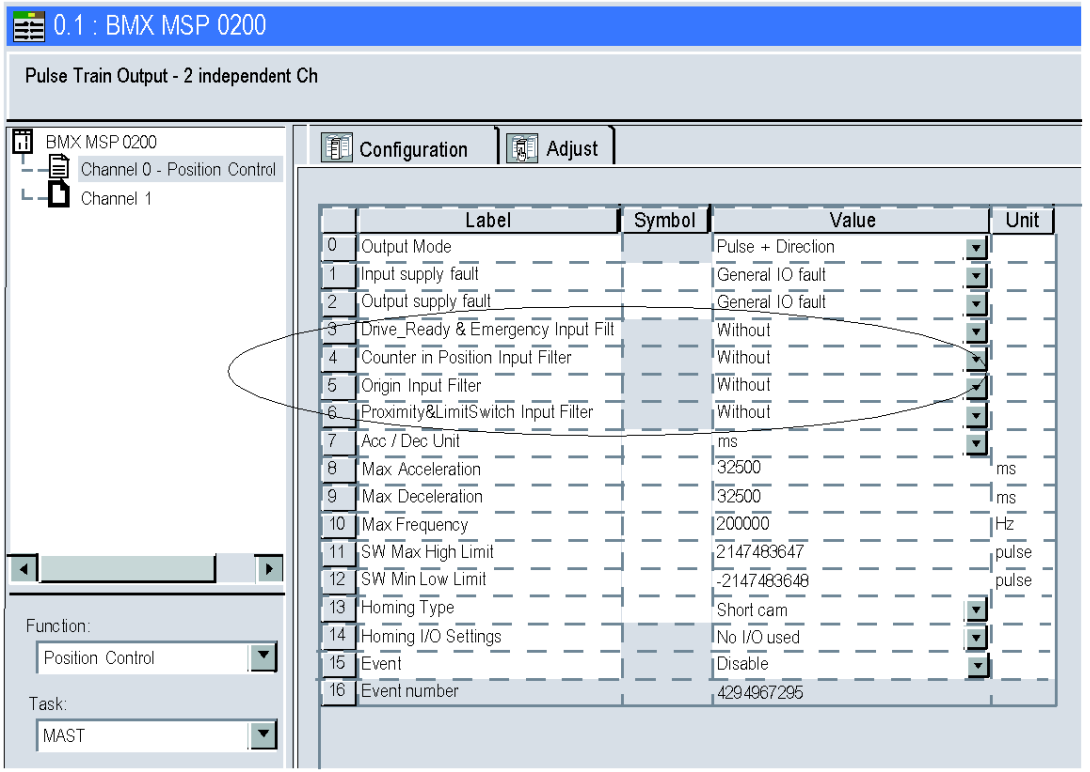
NOTE: For better accuracy of the PTO, set Acc/Dec parameter to Hz/2ms.

NOTE: Physical output are refreshed when PLC is in RUN state only.In STOP state, previous value are maintained.

Programmable Input Filtering

Overivew

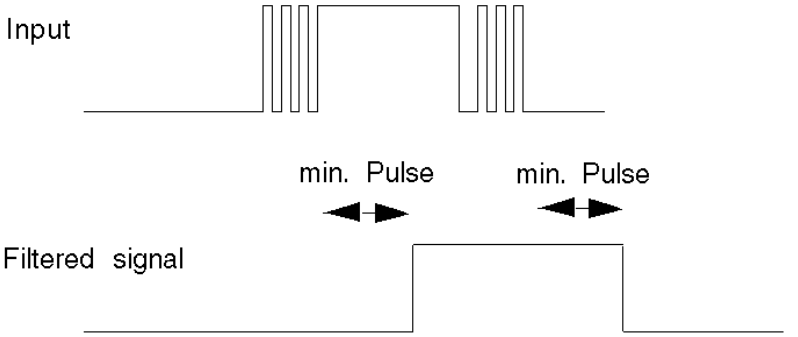
Each of the BMX MSP 0200 PTO module inputs allows input filtering. There are four levels of filtering available (low, medium, high and none), that can be configured in the configuration screen, as shown:



Description

The filtering used is a programmable bounce filter, which operates as follows:

Bounce rejection diagram



In bounce rejection mode, the system delays all transitions until the signal remains stable for the duration defined for the filter level.

Bounce rejection levels

Input	Filter Level	Min Pulse
Drive_Ready&Emergency, Counter_In_Position	No filter	2.3 ms
	Low (For Bounces > 2 kHz)	2.7 ms
	Medium (For Bounces > 1 kHz)	3.5 ms
	High (For Bounces > 250 Hz)	6.3 ms
Proximity&LimitSwitch used as LimitSwitch	No filter	2.1 ms
	Low (For Bounces > 2 kHz)	2.45 ms
	Medium (For Bounces > 1 kHz)	3.25 ms
	High (For Bounces > 250 Hz)	6.3 ms
Origin, Proximity&LimitSwitch used for homing	No filter	60 µs
	Low (For Bounces > 2 kHz)	450 µs
	Medium (For Bounces > 1 kHz)	1.25 ms
	High (For Bounces > 250 Hz)	4.1 ms

For each input , the bounce level to be applied is independently configurable by the user through the configuration parameters %KWr.m.c.2 and %KWr.m.c.3.

Event Sending to Application

Summary

The PTO channels can send events to the application.

To do so in UnityPro configuration screen, enable the event functionality and specify the number of the event task that will be triggered.

PTO channels support 2 sources of events:
- Position reached
- Referencing done

All the events sent by the unit, regardless of the source, call the same single event task in the PLC.

There is only one type of event signaled per call.

The source producing the call is determined in the event task via the Event Sources variable (%IWr.m.c.12).

This variable is updated at the beginning of the event task processing.

NOTE: It is not recommended to send new PTO commmands in Event Task, as they may be rejected.

Enabling

A source will produce its events if the corresponding enable bit is set to 1.

This event source enabling is done through the implicit command object %QWr.m.c.0.

Any event occurring while its source is disabled will be lost. When the source is enabled again, only new event occurrences will be produced.

Object	Type	Symbol	Value
%QWr.m.c.0	INT	Enable Evt Source	One bit per source 1: Enable / 0: Disable
x0	bit		Position reached
x1	bit		Referencing done

Limitations

Each PTO channel can produce a maximum of one event per 2 ms, but this flow may be slowed by the simultaneous transmission of events by several units on the rack bus.

Special Input Interface

The event has a unique input interface; this is only updated at the beginning of event task processing. This interface includes:

- Events Source variable (%Iwr.m.c.10).
- Position: the current position on event time.

Programming Features

11

Overview

This chapter describes the programming features associated to the BMX MSP 0200.

What's in this Chapter?

This chapter contains the following sections:

11.1 General Command Programming

Overview

This section deals with general programming features concerning the BMX MSP 0200 motion functions.

What's in this Section?

This section contains the following topics:

Elementary function description

Elementary Functions

There are 6 basic Motion Commands, which are sent by explicit exchanges:

- FrequencyGenerator *(see page 131)*
- MoveVelocity *(see page 137)*
- MoveAbsolute *(see page 154)*
- MoveRelative *(see page 159)*
- Homing *(see page 185)*
- SetPosition *(see page 200)*

NOTE: The Stop command is sent by implicit exchanges. *(see page 202)*

Command Mechanism

Overview

There are two ways to send motion commands (other than Stop) from the user application:

- Using the specific Elementary Functions (EFs), in the Unity library
- Using the WRITE_CMD instruction

PTO Elementary Functions

The PTO EF family contains 6 instructions:

Name	Input CH	Input 1	Input 2	Input 3
unsigned short FrequencyGenerator	ANY_IODDT %CH	DINT Target_Frequency		
unsigned short MoveVelocity	ANY_IODDT %CH	DINT Target_Velocity		
unsigned short MoveAbsolute	ANY_IODDT %CH	DINT Target_Position	DINT Target_Velocity	BYTE BufferMode
unsigned short MoveRelative	ANY_IODDT %CH	DINT Target_Distance	DINT Target_Velocity	BYTE BufferMode
unsigned short Homing	ANY_IODDT %CH	DINT Position	DINT Velocity	
unsigned short SetPosition	ANY_IODDT %CH	DINT Position		

Stop Command

There is a specific mechanism to send Stop commands, which uses implicit exchanges.

When the axis needs to be stopped, the specific implicit command object: "Stop Level" (%Qr.m.c.2) must be set to 1.

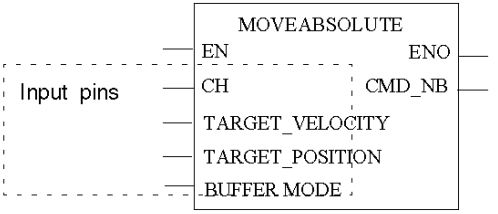
A Stop command takes precedence over any other motion commands: any command sent while the axis is stopping will be rejected.

Motion Command Using FBD

At a Glance

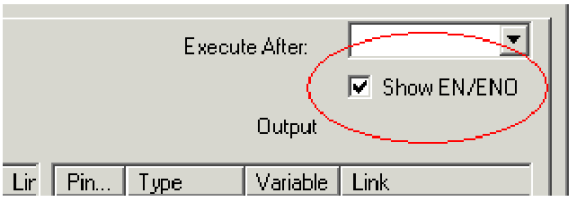
The first way to send a motion command is by using the specific Elementary Functions (EFs), in the Unity library

For example: the EF MoveAbsolute



EN/ENO Pins

In order to make the EN and ENO pin appear in the FBD representation double click on the FBD representation (or right click and select `properties`) and check the Show EN/ENO checkbox.



EN and ENO are general pins used by all EFs. The ENO pin is computed only if EN is set to 1, otherwise its value is undefined.

The output pin CMD_NB is computed internally. There are 3 different cases:

- If the command has been correctly sent and accepted, this object will give a command number (between 0x01 and 0x7F), and can be used to follow the status of the command through the implicit status objects (%IWr.m.c.0 to %IWr.m.c.5). The ENO output of the EF is set to 1.
- If the command has been correctly sent but rejected, CMD_NB takes the value of the command number for the first 7 bits, but its most significant bit will be set to 1 (value between 0x81 and 0xFF). The ENO output of the EF is set to 1
- If the command has been incorrectly sent, CMD_NB will remain at 0. The ENO output of the EF is set to 0

In the last two cases, an error notification will be reported through the CMD_ERR system object (%MWr.m.c.1.1).

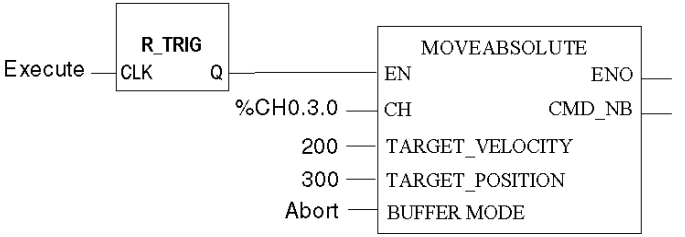
NOTE: It is necessary to have EN set to 1 to change command parameter values.

⚠ WARNING

UNEXPECTED PARAMETER CHANGES

Command parameters will change on each PLC cycle if EN is set to 1.

Add a rising edge detection (`R_Trig`) as shown on the diagram below:



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

Other Pins

The input pins correspond to all command parameters associated with this specific command. (except the command code)

When the command is sent through the PTO EF, the %MWr.m.c.13 object takes the same value as CMD_NB.

Motion Command using Write_CMD

At a Glance

It is also possible to directly write the parameter values into the corresponding %MWCmd objects and then trigger the execution of the motion command by sending a WRITE_CMD instruction.

Description

The behavior is similar to the one with EFs. However, it is necessary to specify what kind of command is to be executed with the command code byte. If this parameter is not valid, the command will be rejected and the detected error will be reported in the CMD_CODE_INV status object (%MWr.m.c.3.2).

When sending a command through WRITE_CMD, the command object %MWr.m.c.13 is computed internally. Its behavior is exactly the same as the CMD_NB output pin of the EF when the command is sent by EF.

This mechanism can be used to send motion commands from Unity Operator Screens *(see Unity Pro, Operating Modes,)*, which can't be done with only EFs.

NOTE: A command example, written in ST representation is given for each EF. *(see page 129)*

Command Mechanism Sending Rules

At a Glance

Independent of the method used to send a command, certain constraints must be taken into account:

- Only one command can be sent at a time (at most one command per PLC cycle). The previous command needs to be received by the channel before sending a new one.
  Any command sent while another one is being exchanged with the channel will be ignored.
  The availability can be checked on the bus rack through the system bit CMD_IN_PROGR (%MWr.m.c.0.1).
- The channel can receive two commands in succession. One will be executed, while the second is in buffer, waiting for the first one to be completed. This is valid for positioning commands only, and the chosen buffer mode must be Buffered or BlendingPrevious.
- When a command is being executed, and another one is already in buffer, the channel cannot accept a third command. Check the availability of the channel before sending any command.
  If a command is sent while the channel is not available, it will be rejected, all commands in the channel will be aborted, the axis will be stopped and the corresponding error notification will be reported in the BUFFER_FULL status object (%MWr.m.c.3.4).

Module Availability to Commands

The value of implicit status objects: **Idle** and **FreeCmdBuf** allows to check if the module is available for a new command.

The following table details the different cases:

Idle	FreeCmdBuf	Meaning
0	0	Two cases: ● A command is being sent ● A command is being executed, and another one is in buffer In both cases, no command should be sent.
0	1	A command is being executed, but the command buffer is free. A new command can be sent and will be kept in the command buffer; FreeCmdBuf is set to 0.
1	0	No significance
1	1	The buffer is free and no command is being executed. A new command can be sent.

Parameter Description

Overview

Each command has its related command parameters, setting parameters and adjustment parameters (refer to each function for more details).

Command Parameters

Command parameters can be set in the application:

- directly in the interface objects, previous to executing the Write_Cmd instruction
- by executing EFs

NOTE: Sending a new command of the same type aborts the active command.

NOTE: It is not possible to modify the command parameters of a Homing command, since it does not support the succession of several commands. *(see page 122)*

Setting Parameters

Setting Parameters are only managed through the Unity Pro configuration tool.

Adjustment Parameters

Adjustment Parameters are managed through the Unity Pro Adjust tool.

They can be read by executing the Read_Param instruction and their initial values can be set to their current values by executing the Save_Param instruction.

They can be set by

- modifying %M.. objects and executing the Write_Param instruction
- executing the Restore_Param instruction to set them to their initial values.

When accessing the Adjustment Parameters:

- through the IODDTs or the Adjustment screen, it is possible to directly write the unsigned values.
- through their topological addresses, only signed types are accepted. Converting the unsigned value into a signed value before writing in the %MWr.m.c object is necessary.

If Adjustment Parameters are changed while the PTO channel is running, this change will take effect on next commands.

Limit Parameters

These are objects used to define valid ranges of values for command parameters.

Configuration Parameters			
Object	**Type**	**Symbol**	**Description**
%KWr.m.c.4	UINT	Max Acceleration	Acceleration Rate Maximum Value
%KWr.m.c.5	UINT	Max Deceleration	Deceleration Rate Maximum Value
%KDr.m.c.6	UDINT	Max Frequency	Maximum Frequency (in Hz)
%KDr.m.c.8	DINT	SW Max High Limit	Software Pulse Number Maximum High Limit
%KDr.m.c.10	DINT	SW Min Low Limit	Software Pulse Number Minimum Low Limit

Adjustment Parameters			
Object	**Type**	**Symbol**	**Description**
%MDr.m.c.14	UDINT	SW High Limit	Software Pulse Number High Limit
%MDr.m.c.16	UDINT	SW Low Limit	Software Pulse Number Low Limit

Any command sent with parameters that are inconsistent with the specified limits will be rejected.

Constraints on Configuration and Adjustment Parameters:

The following rules of consistency between configuration and adjustment parameters must be observed:

- SW High Limit ≤SW Max High Limit
- SW Max High Limit > SW Min Low Limit
- SW High Limit > SW Low Limit
- SW Low Limit ≥ SW Min Low Limit
- Start Frequency ≤Max Frequency
- Stop Frequency ≤Max Frequency
- Homing Velocity ≤Max Frequency
- Start Frequency ≤Homing Velocity if Start Frequency enabled
- Stop Frequency ≤Homing Velocity if Stop Frequency enabled
- Acceleration Rate ≤Max Acceleration
- Deceleration Rate ≤Max Deceleration
- Emergency Deceleration Rate ≤Max Deceleration

If a setting parameter or initial parameter does not respect one of these rules, the configuration will not be accepted.

NOTE: Unity Pro Initial parameters respect all the rule above.

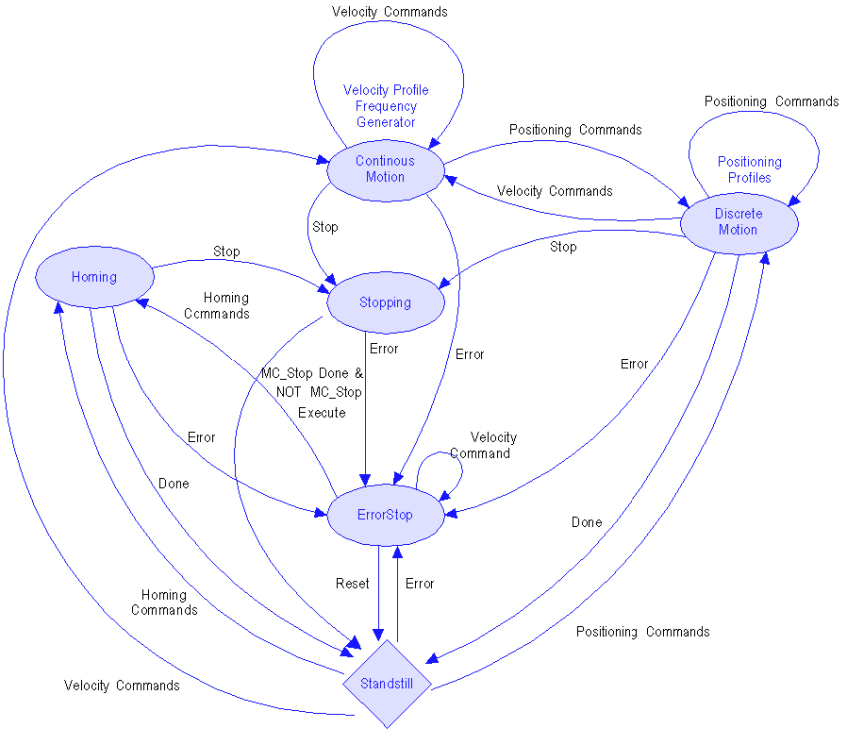
If an adjustment with an invalid parameter is set:

- The parameter will be rejected
- The previous values will be maintained
- The detected error will be reported in the ADJUST_FLT status word (%MWr.m.c.4)

Sequence of commands

Motion State Diagram

Any sequence of commands must respect the following state diagram:



Allowed Sequence of Commands

The PTO channel can accept the following sequence of command:

Current Command								
		No Command	**Frequency Generator**	**Move Velocity**	**Move Absolute**	**Move Relative**	**Homing**	**Set Position**
Next Command	**No Command**	Reject	Reject	Reject	Reject	Reject	Reject	Reject
	Frequency Generator	Accept	Accept	Accept	Accept	Accept	Reject	Reject
	MoveVelocity	Accept	Accept	Accept	Accept	Accept	Reject	Reject
	MoveAbsolute (Abort)	Accept	Accept	Accept	Accept	Accept	Reject	Reject
	MoveAbsolute (Buffered/Blending)	Accept	Reject	Reject	Accept	Accept	Reject	Reject
	MoveRelative (Abort)	Accept	Accept	Accept	Accept	Accept	Reject	Reject
	MoveRelative (Buffered/Blending)	Accept	Reject	Reject	Accept	Accept	Reject	Reject
	Homing	Accept	Reject	Reject	Reject	Reject	Reject	Reject
	SetPosition	Accept	Reject	Reject	Reject	Reject	Reject	Reject

Reject:

- The sequence of commands described in the cell is not supported. The new command will be rejected.
- All commands in progress will be aborted, the axis will be stopped and an error notification will be reported in the CMD_SEQ_INV status object (%MWr.m.c.3.3).

Accept:

- The sequence of commands described in the cell is supported.
- The new command is accepted. Its execution starts either immediately, or after the completion of current command, depending upon the set buffer mode.

The BufferMode command parameter is used to determine how a sequence of commands will be executed:

- Abort: the new command aborts the current command.
- Buffered: the new command is executed after the current command is completed.
- BlendingPrevious: the two commands are merged at the target velocity of the first command.

For each buffer mode, the behavior is detailed in MoveRelative description.
(see page 159)

Axis Status Information

At a Glance

In order to know which PLCopen state the axis is in, check the value of the AXIS_STS object (%IWr.m.c.6).

Axis Status

This word does not describe all the PLCopen states that appear in the state diagram, but it indicates which of the following 4 states the axis is in:

STANDSTILL state is described with the following set of information:	• bit0 (MOVING) = 0 • bit1 (STOPPING) = 0 • bit3 (AXIS_FLT) = 0 • %IWr.m.c.0 = 0 & %IWr.m.c.7.bit0 = 1 (no command in execution) • %IWr.m.c.1 = 0 & %IWr.m.c.7.bit1 = 1 (no command in buffer)
STOPPING state is described with the following set of information:	• bit1 (STOPPING) = 1 • bit3 (AXIS_FLT) = 0 • %IWr.m.c.0 = 0 & %IWr.m.c.7.bit0 = 1 (no command in execution) • %IWr.m.c.1 = 0 & %IWr.m.c.7.bit1 = 1 (no command in buffer)
ERROR_STOP state is described with the following set of information:	• bit1 (STOPPING) = 1 • bit3 (AXIS_FLT) = 1 • %IWr.m.c.0 = 0 & %IWr.m.c.7.bit0 = 1 (no command in execution) • %IWr.m.c.1 = 0 & %IWr.m.c.7.bit1 = 1 (no command in buffer)
Command in execution. This is not a PLCopen state but includes several of them. It is described with the following set of information:	• bit1 (STOPPING) = 0 • bit3 (AXIS_FLT) = 0 • %IWr.m.c.0 $\neq$ 0 & %IWr.m.c.7.bit0 = 0 (command in execution) • bit0 (MOVING) = 1

This word (%IWr.m.c.0) indicates the exact PLCopen state:

Each command sent has an allocated number and can be read through the CMD_SENT_NB (%MWr.m.c.13) object or the EF output.

Knowing these two numbers, it is possible to identify which command and which type of profile is currently being executed and which state the axis is in (CONTINUOUS MOTION, DISCRETE MOTION and HOMING). This information can also be obtained using the Cmd_Status function. *(see page 203)*

NOTE: when Drive_Enable is disabled, axis referenced bit will be cleared, and any command can be accepted.

11.2 Positioning Function Description

Overview

The BMX MSP 0200 can use a library of 7 basic Motion Commands which are described in this section.

What's in this Section?

This section contains the following topics:

Topic	Page
STOP	202
Command Status Follow-Up	203

Frequency Generator

Description

The PTO channel provides a pulse output signal at a specified frequency.

Physical Inputs/Output

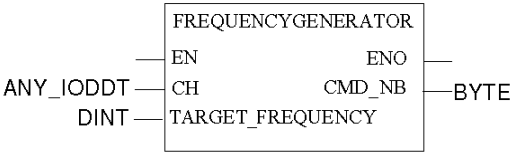
Input/Output	Description
Drive_Ready&Emergency input (optional)	The pulse output is generated as long as a current goes through Drive_Ready&Emergency input. *(see page 224)*
Proximity&LimitSwitch input (optional)	Used as a LimitSwitch. *(see page 224)*
Drive_Enable output	To be connected to the corresponding input of the drive. Enables the drive when active. This output is directly controlled through an implicit command object (%Qr.m.c.0).

Configuration Parameters

Parameter	Valid Values
PTO Output Mode	Value 0: Pulse + Direction (Default) Value 1: CW/CCW Value 2: A/B Phases Value 3: Pulse + Direction – Reverse Value 4: CW/CCW – Reverse Value 5: A/B Phases – Reverse

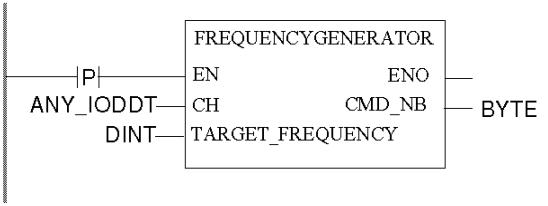
FBD Representation

Representation:



LD Representation

Representation:



⚠ WARNING

UNINTENDED INVALID COMMAND

Commands will be sent on every PLC cycle if EN is set to 1. *(see page 119)*

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

Representation in IL

Representation:

```
FREQUENCYGENERATOR (CH := (*ANY_IODDT*), TARGET_FREQUENCY :=
(*DINT*))
ST (*BYTE*)
```

Representation in ST

Representation:

```
(*BYTE*) := FREQUENCYGENERATOR (CH := (*ANY_IODDT*),
TARGET_FREQUENCY := (*DINT*));
```

Command example using the WRITE_CMD command mechanism in ST representation:

```
if (ChangeFreq = True) then  %CH0.1.0.CMD_CODE := 1;
%CH0.1.0.TGT_VELOCITY := 5000; WRITE_CMD(%CH0.1.0);
ChangeFreq := False; end_if;
```

Command Specific Parameters

Parameter	Valid Values
Target Velocity (in Hz)	-200 kHz to 200 kHz Absolute value limited by Max Frequency

Overall Parameters

This table describes all the functional parameters associated to the function.

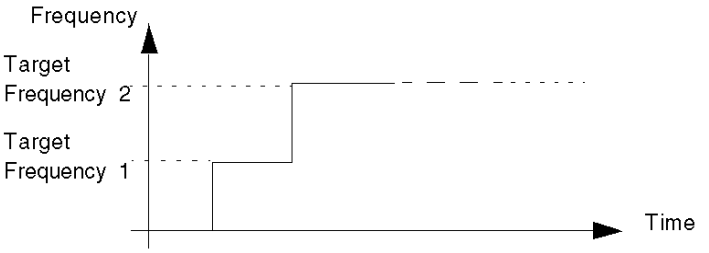
Explicit Command Parameters		Setting Parameters		Adjustment Parameters	
Address	**Parameter**	**Address**	**Parameter**	**Address**	**Parameter**
%MWr.m.c.6 (byte 0)	Command Code (=1)	%KWr.m.c.1(byte 0)	Output Mode	%MWr.m.c.25	Hysteresis
%MDr.m.c.10	Target Frequency	%KDr.m.c.6	Max Frequency		

Frequency Generator Complex Profile

At a Glance

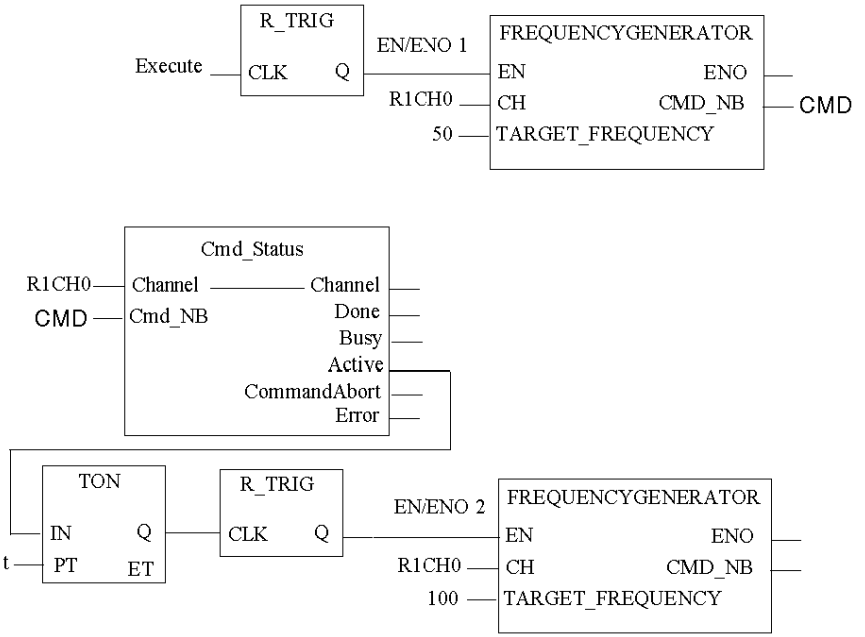
When a frequency generator command is running, it is possible to modify the target frequency, such as shown by the figure below:

Frequency generator - change of frequency



FBD program

Program to obtain the above profile:

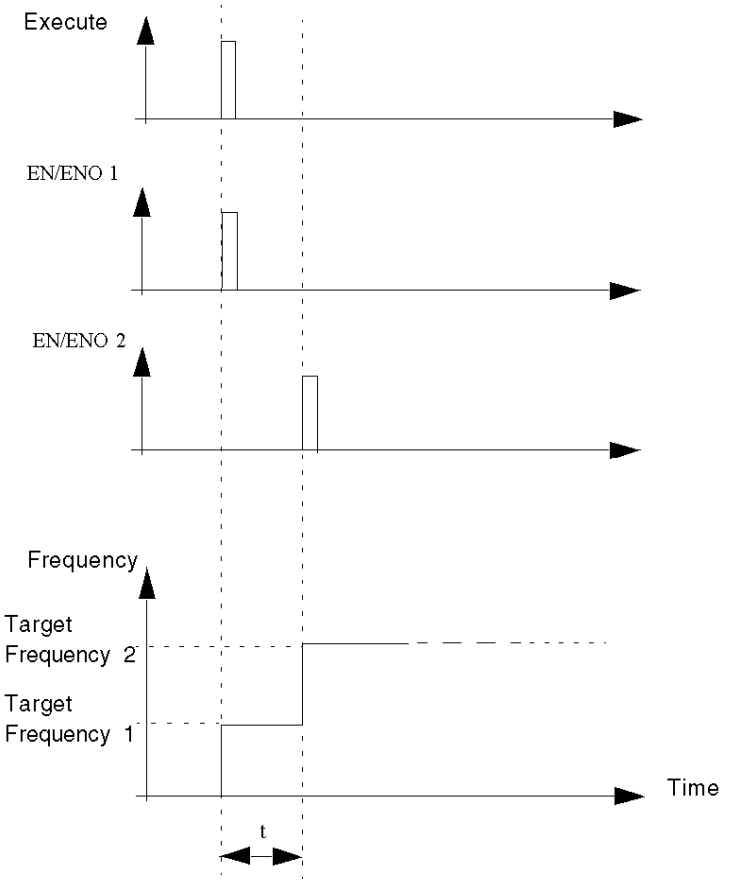


R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Cmd_Status is the command status follow up function. *(see page 203)*

Time Diagram

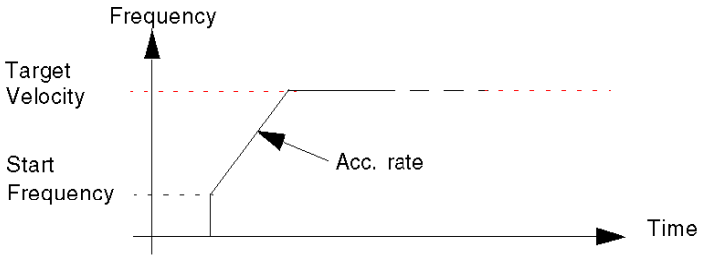
Time diagram of the frequency generators Input / Output



Move Velocity

Description

This function is used to generate a pulse output at a specified frequency, by reaching this frequency smoothly through an acceleration ramp.



Physical Inputs/Output

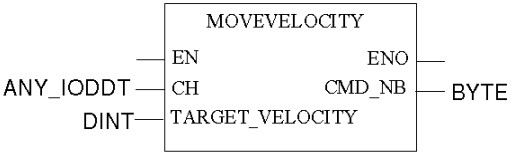
Input/Output	Description
Drive_Ready&Emergency input (optional)	The pulse output is generated as long as a current goes through Drive_Ready&Emergency input. *(see page 224)*
Proximity&LimitSwitch input (optional)	Used as LimitSwitch. *(see page 224)*
Drive_Enable output:	To be connected to the corresponding input of the drive. Enables the drive when active. This output is directly controlled by the user through an implicit command object (%Qr.m.c.0).

Configuration Parameters

Parameter	Valid Values
PTO Output Mode	Value 0: Pulse + Direction (Default) Value 1: CW/CCW Value 2: A/B Phases Value 3: Pulse + Direction – Reverse Value 4: CW/CCW – Reverse Value 5: A/B Phases – Reverse
Acceleration / Deceleration Unit	ms or Hz/2ms Default is ms

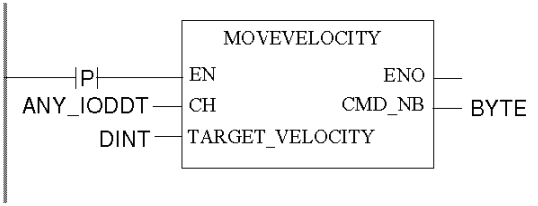
Representation in FBD

Representation:



Representation in LD

Representation:



⚠ WARNING

UNINTENDED INVALID COMMAND

Commands will be sent on every PLC cycle if EN is set to 1. *(see page 119)*

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

Representation in IL

Representation:

```
MOVEVELOCITY (CH := (*ANY_IODDT*), TARGET_VELOCITY :=
(*DINT*))
ST (*BYTE*)
```

Representation in ST

Representation:

```
(*BYTE*) := MOVEVELOCITY (CH := (*ANY_IODDT*), TARGET_VELOCITY
:= (*DINT*));
```

Command example using the WRITE_CMD command mechanism in ST representation:

```
if (ChangeVel = True) then  %CH0.1.0.CMD_CODE := 2;
%CH0.1.0.TGT_VELOCITY := 5000; WRITE_CMD(%CH0.1.0); ChangeVel
:= False; end_if;
```

Command Specific Parameters

Parameter	Valid Values
Target Velocity (in Hz)	-200 kHz to 200 kHz Absolute value limited by Max Frequency

Adjustment Parameters

Parameter	Valid Values
Start Frequency (in Hz)	0 Hz to 65,535 Hz, default is 0Hz, limited by Max Frequency
Stop Frequency (in Hz)	0 Hz to 65,535 Hz, default is 0Hz, limited by Max Frequency
Acceleration Rate	10 to 32,500, default is 100, limited by Max Acceleration
Deceleration Rate	10 to 32,500, default is 100, limited by Max Deceleration
Emergency Deceleration Rate	10 to 32,500, default is 100, limited by Max Deceleration

Overall Parameters

This table describes all the functional parameters associated to the function.

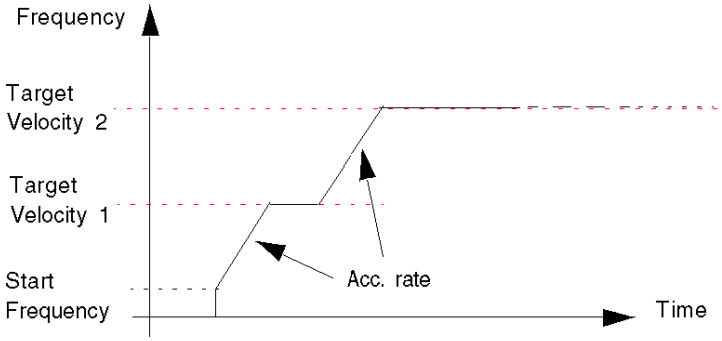
Explicit Command Parameters		Setting Parameters		Adjustment Parameters	
Address	**Parameter**	**Address**	**Parameter**	**Address**	**Parameter**
%MWr.m.c.6 (byte 0)	Command Code (=2)	%KWr.m.c.1(byte 0)	Output Mode	%MWr.m.c.18	Start Frequency
%MDr.m.c.10	Target Velocity	%KWr.m.c.1(byte 12)	Acc/Dec Unit	%MWr.m.c.19	Stop Frequency
		%KWr.m.c.4	Acc Max	%MWr.m.c.20	Acceleration Rate
		%KWr.m.c.5	Dec Max	%MWr.m.c.21	Deceleration Rate
		%KDr.m.c.6	FMax	%MWr.m.c.25	Hysteresis

Move Velocity Complex Profile 1

At a Glance

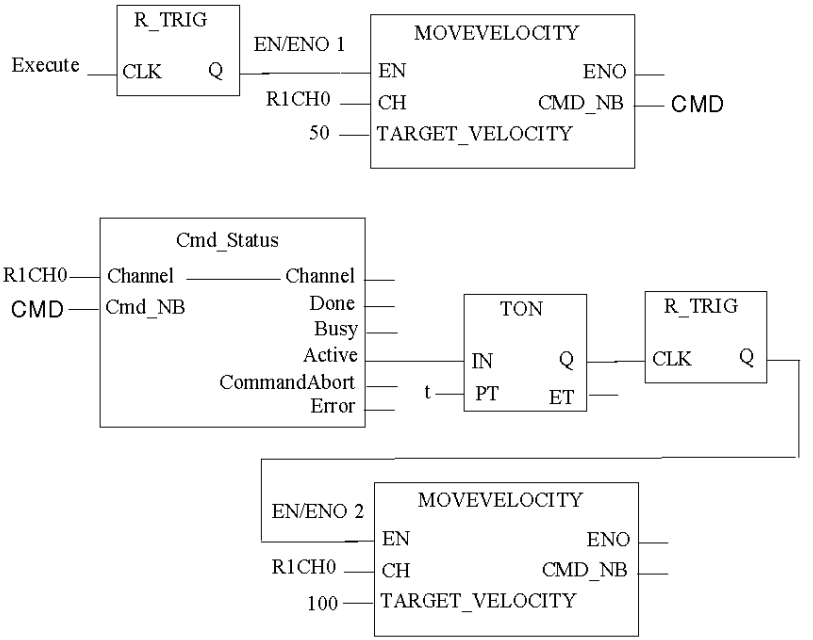
When a velocity profile is being output, it is possible to modify the target velocity to a higher or a lower value, such as shown by the figures below:

MoveVelocity - change of velocity



FBD Program

Program to obtain the profile

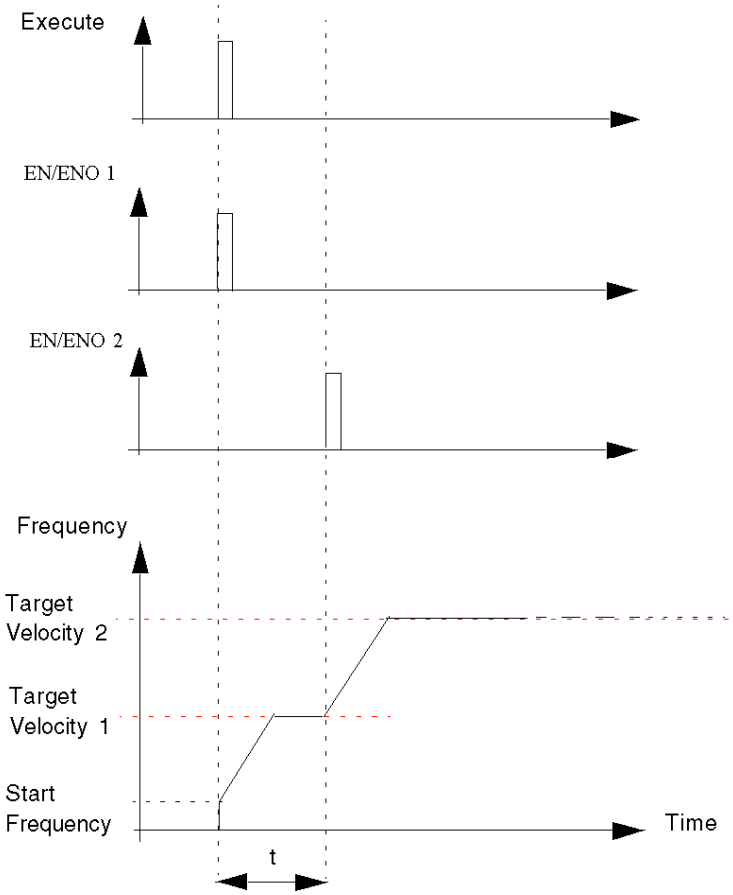


R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Cmd_Status is the command status follow up function. *(see page 203)*

Time Diagram

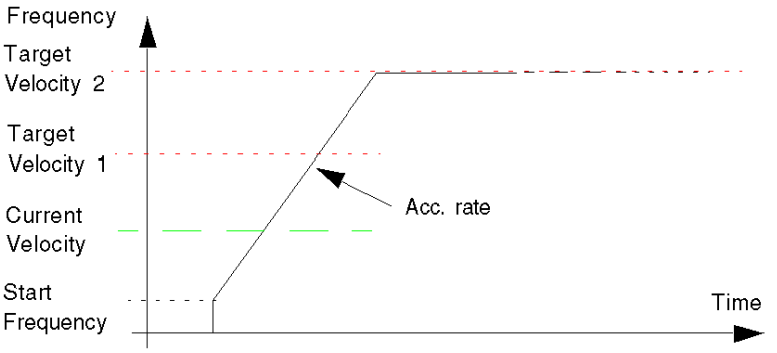
Time diagram of the MOVEVELOCITY Input / Output



Move Velocity Complex Profile 2

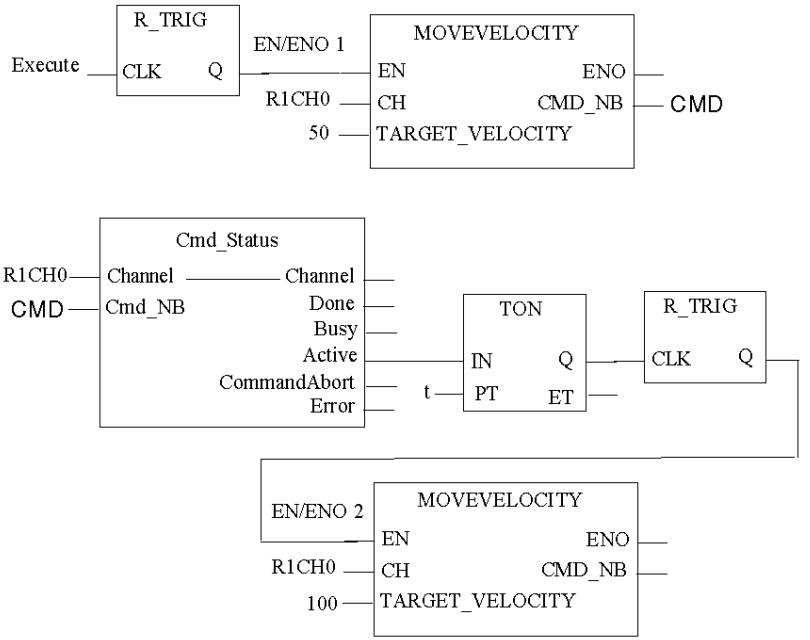
At a Glance

If the first target velocity has not been reached, the target velocity can be changed during acceleration/deceleration phase):



FBD Program

Program to obtain the profile

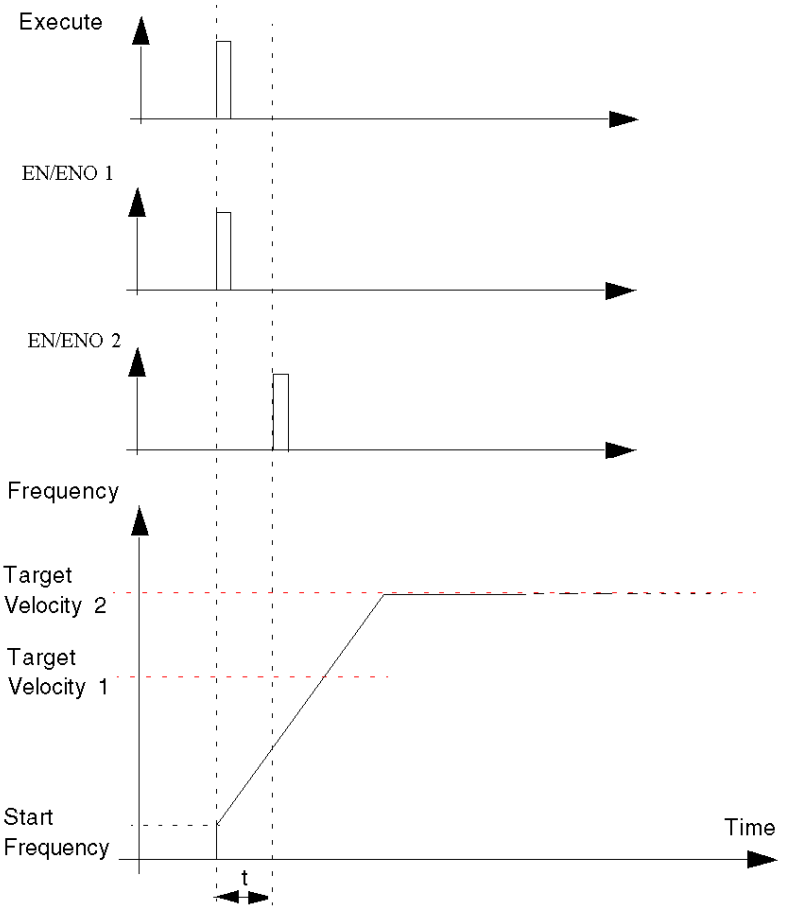


R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Cmd_Status is the command status follow up *(see page 203)* function.

Time Diagram

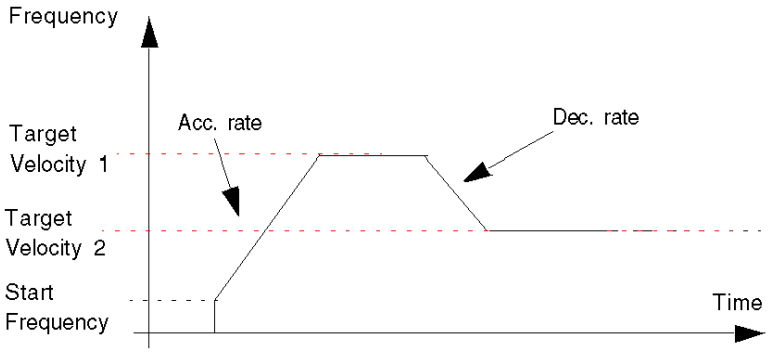
This is the time diagram of the MOVEVELOCITY Input / Output:



Move Velocity Complex Profile 3

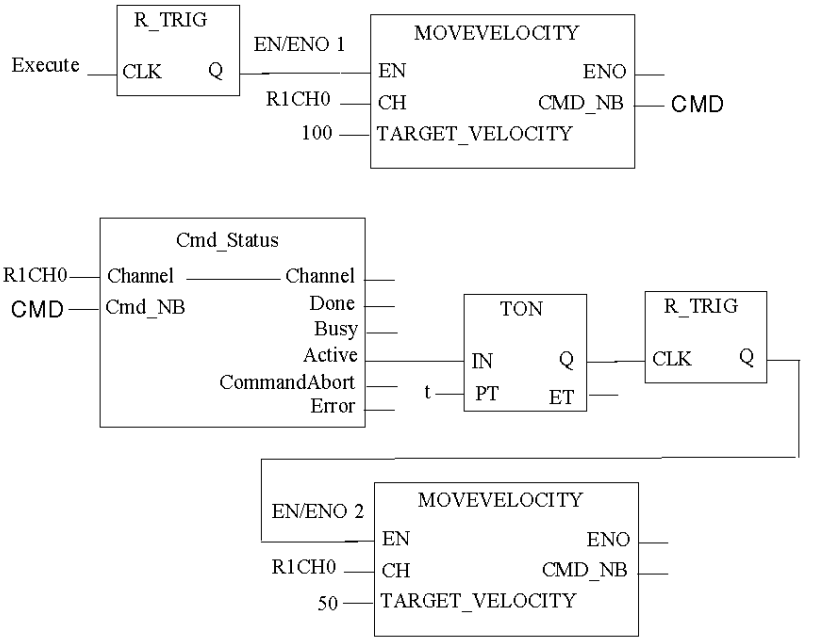
At a Glance

If the new target velocity is lower than the previous one, there will be a deceleration ramp.



FBD Program

Program to obtain the profile

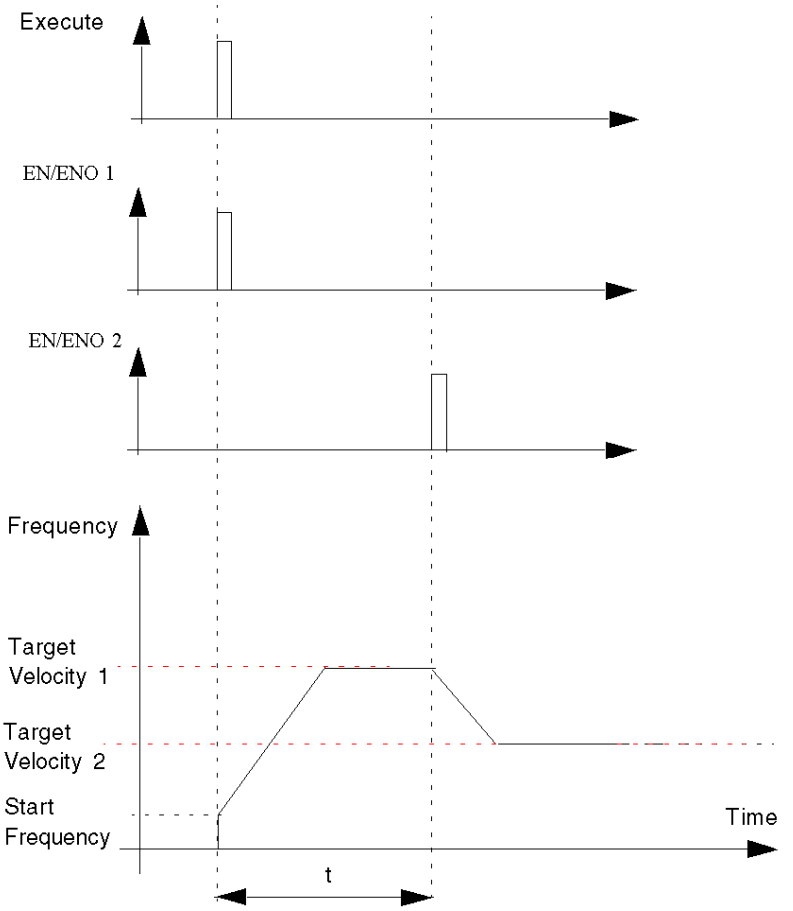


R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Cmd_Status is the command status follow up function. *(see page 203)*

Time Diagram

This is the time diagram of the MOVEVELOCITY Input / Output:

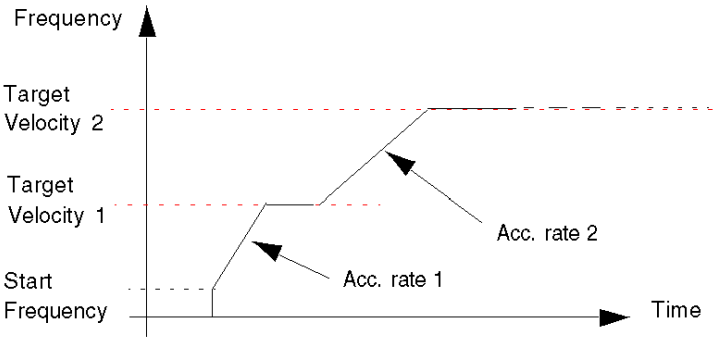


Move Velocity Complex Profile 4

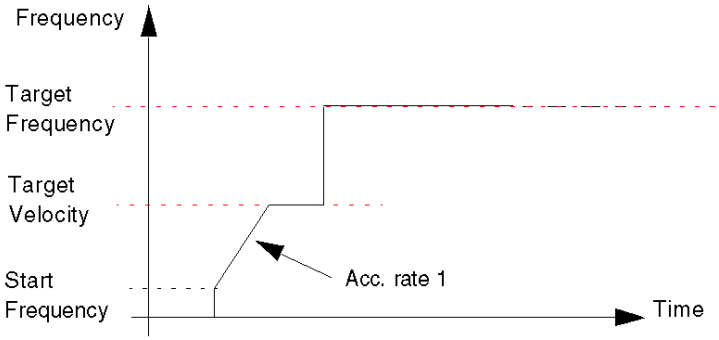
At a Glance

If a velocity profile is being output, a new continuous motion command can be sent to the channel and abort the current command, whether the target velocity has been reached or not. The new command can be:

Case 1: a velocity profile command with possible different acceleration/deceleration rates:

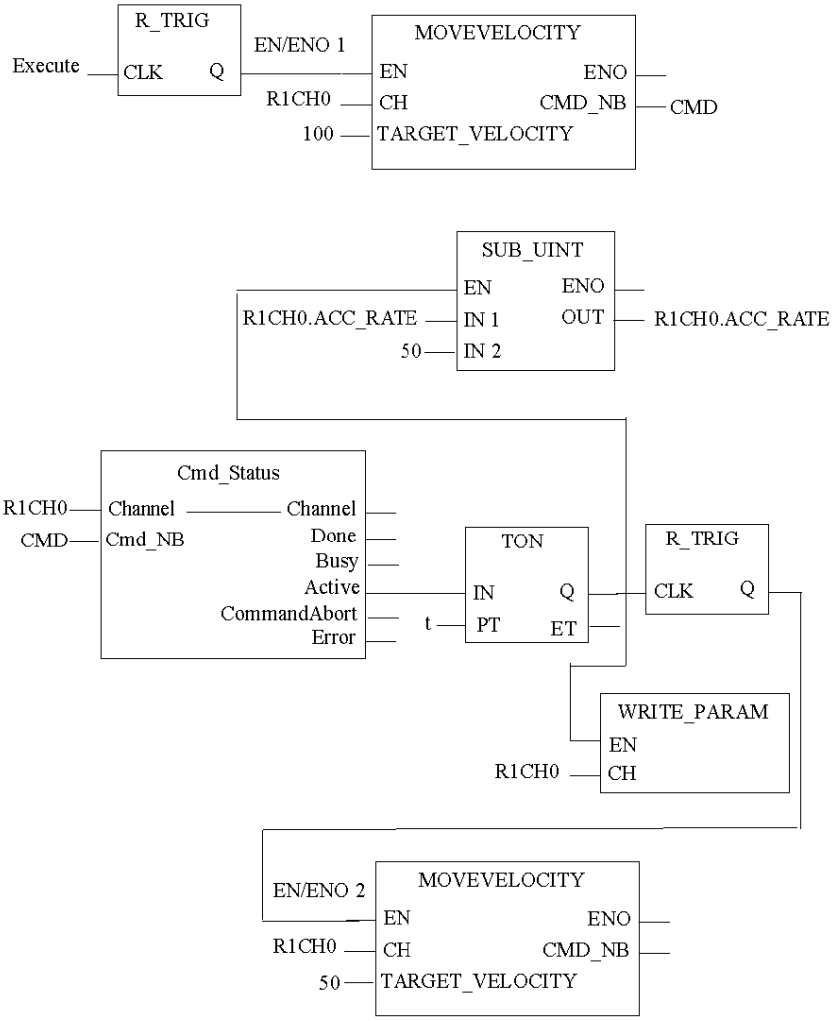


Case 2: a FrequencyGenerator command:



FBD Program Case 1

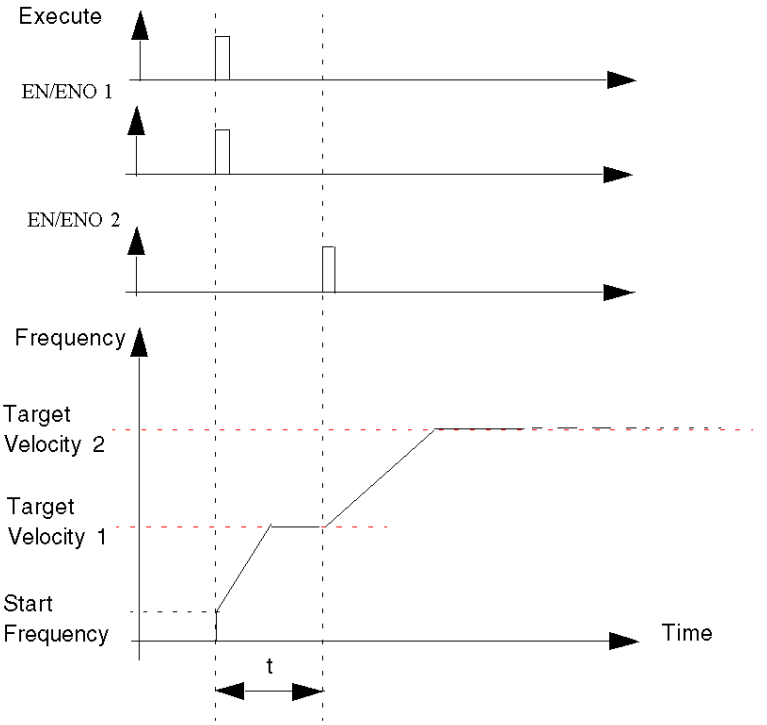
Program to obtain the profile in case 1:



R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

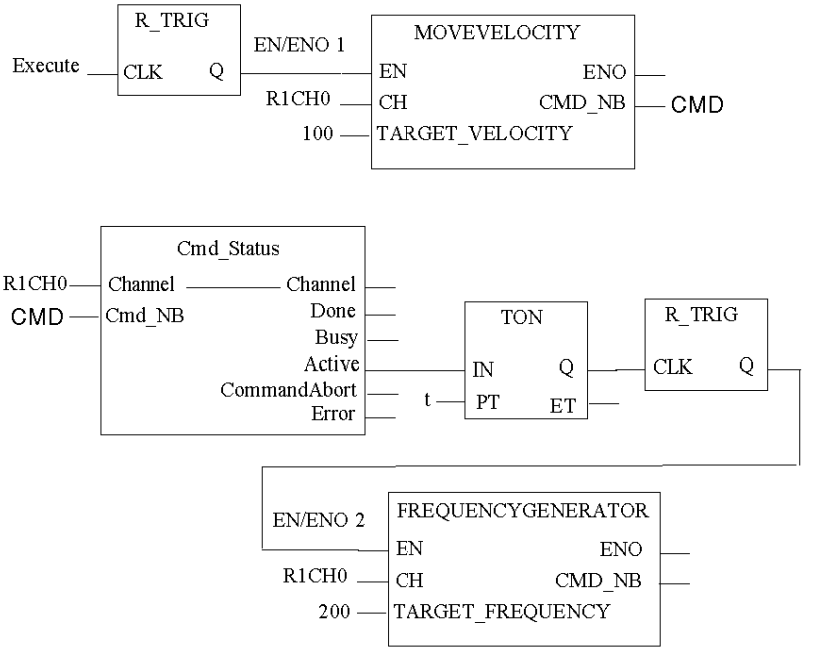
Time Diagram Case 1

Time diagram of the MoveVelocity Input / Output for case 1:



FBD Program Case 2

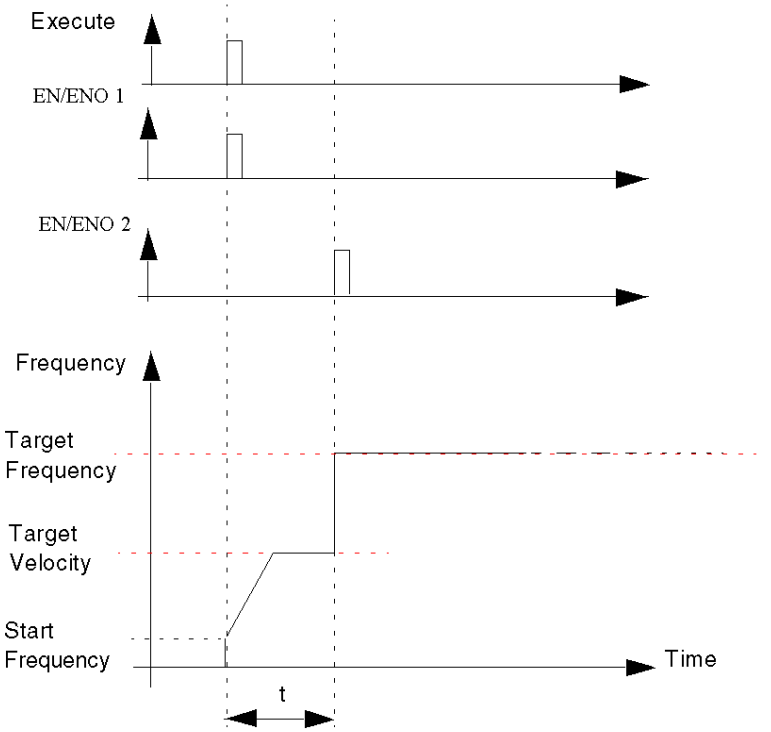
Program to obtain the profile in case 2:



R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Time Diagram Case 2

Time diagram of the MoveVelocity Input / Output for case 2:



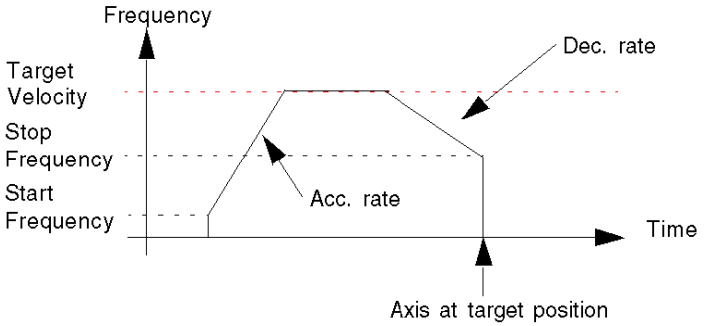
Absolute Positioning: Move Absolute

Description

This function is used to manage a complete movement of the axis from the current position to a specified target position.

The target position is directly specified with its coordinate, in pulses, relative to a previously set origin.

The velocity of the axis will follow a trapezoidal profile:



NOTE: No absolute positioning command can be performed while "REFERENCED" is low. Any absolute positioning command sent while REFERENCED is low will be rejected and an error notification is reported in the CMD_FLT status word (%MWr.m.c.3.5).

"REFERENCED" is an implicit bit (%IWr.m.c.6.7) which reports information on whether the axis is referenced or not.This bit will be set to 1 by the module when a referencing command (Homing or SetPosition) is completed

It will return to 0:

- Each time synchronization is lost between the PTO channel and the drive (Drive_Ready input is off.)
- At the beginning of each new homing command.

Physical Inputs/Output

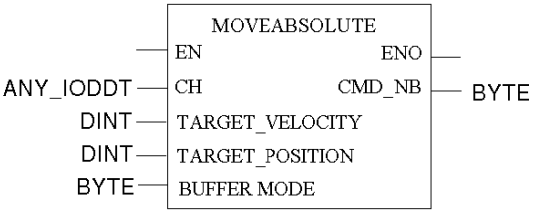
Input/Output	Description
Drive_Ready&Emergency input (optional)	The pulse output is generated as long as a current goes through Drive_Ready&Emergency input. *(see page 224)*
Proximity&LimitSwitch input (optional)	Used as a LimitSwitch. *(see page 224)*
Counter_in_Position input (optional)	Only for information. Input from the drive goes high when positioning movement is completed (the drive's error counter is empty).
Drive_Enable output:	To be connected to the corresponding input of the drive. Enables the drive when active. This output is directly controlled by the user through an implicit command object (%Qr.m.c.0).

Configuration Parameters

Parameter	Valid Values
PTO Output Mode	Value 0: Pulse + Direction (Default) Value 1: CW/CCW Value 2: A/B Phases Value 3: Pulse + Direction – Reverse Value 4: CW/CCW – Reverse Value 5: A/B Phases – Reverse
Acceleration / Deceleration Unit	ms or Hz/2ms Default is ms

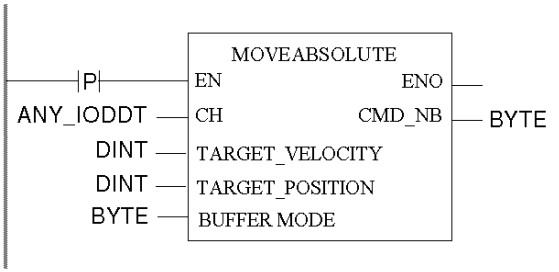
Representation in FBD

Representation:



Representation in LD

Representation:



⚠ WARNING

UNINTENDED INVALID COMMAND

Commands will be sent on every PLC cycle if EN is set to 1. *(see page 119)*

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

Representation in IL

Representation:

```
MOVEABSOLUTE (CH := (*ANY_IODDT*), TARGET_POSITION :=
(*DINT*), TARGET_VELOCITY := (*DINT*), BUFFERMODE := (*BYTE*))
ST (*BYTE*)
```

Representation in ST

Representation:

```
(*BYTE*) := MOVEABSOLUTE (CH := (*ANY_IODDT*), TARGET_POSITION
:= (*DINT*), TARGET_VELOCITY := (*DINT*), BUFFERMODE :=
(*BYTE*));
```

Command example using the WRITE_CMD command mechanism in ST representation:

```
if (ChangePos = True) then  %CH0.1.0.CMD_CODE := 3;
%CH0.1.0.TGT_VELOCITY := 5000; %CH0.1.0.TGT_POSITION := 50000;
%CH0.1.0.BUFFER_MODE :=1; WRITE_CMD(%CH0.1.0); ChangePos :=
False; end_if;
```

Command Specific Parameters

Parameter	Valid Values
Target position (in pulses)	- 2,147,483,648 to 2,147,483,647 Must be enclosed between SW Low Limit and SW High Limit
Target Velocity (in Hz)	1 Hz to 200 kHz Absolute value limited by Max Frequency
Buffer mode	Value 0: Abort Value 1: Buffered Value 2: BlendingPrevious

Parameters

Parameter	Valid Values
Hysteresis (Slack)	0 to 255 pulses, default is 0 For A/B Phase output mode only (Normal or Reverse)
Start Frequency (in Hz)	0 Hz to 65,535 Hz Default is 0Hz, limited by Max Frequency
Stop Frequency (in Hz)	0 Hz to 65,535 Hz Default is 0Hz, limited by Max Frequency
Acceleration Rate	10 to 32,500, default is 100, limited by Max Acceleration
Deceleration Rate	10 to 32,500 Default is 100, limited by Max Deceleration
Emergency Deceleration Rate	10 to 32,500 Default is 100, limited by Max Deceleration
Software High Limit (in pulses)	-2,147,483,647 to 2,147,483,647 Default is 2,147,483,647 Must be between SW Low Limit and SW Max High Limit
Software Low Limit (in pulses)	-2,147,483,648 to 2,147,483,646 Default is - 2,147,483,648 Must be enclosed between SW Min Low Limit and SW High Limit

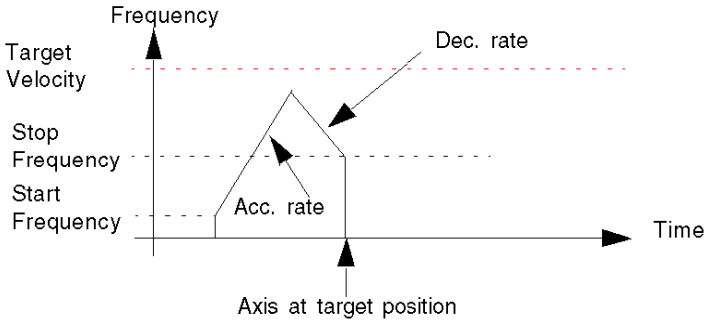
Debugging Parameters

This table describes all the functional parameters associated to the function.

Explicit Command Parameters		Setting Parameters		Adjustment Parameters	
Address	**Parameter**	**Address**	**Parameter**	**Address**	**Parameter**
%MWr.m.c.6 (byte 0)	Command Code (=3)	%KWr.m.c.1 (byte 0)	Output Mode	%MWr.m.c.18	Start Frequency
%MDr.m.c.10	Target Velocity	%KWr.m.c.1 (byte 12)	Acc/Dec Unit	%MWr.m.c.19	Stop Frequency
		%KWr.m.c.4	Acc Max	%MWr.m.c.20	Acceleration Rate
		%KWr.m.c.5	Dec Max	%MWr.m.c.21	Deceleration Rate
		%KDr.m.c.6	FMax	%MWr.m.c.25	Hysteresis

Special cases

If the set target velocity cannot be reached before attaining the target position, the axis velocity will then follow a triangular profile:



Complex Profiles

Complex profiles for MOVEABSOLUTE Position are the same as for MOVERELATIVE

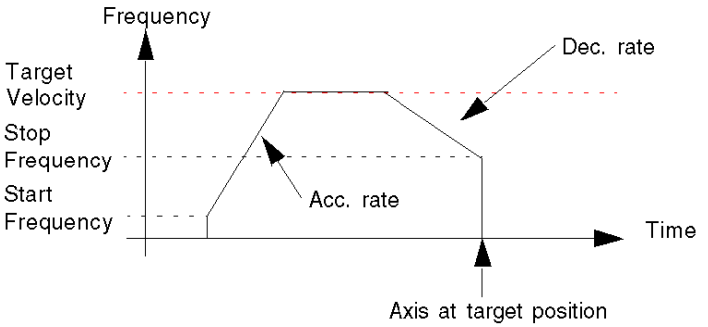
Relative Positioning: Move Relative

Description

This function is used to manage a complete movement of the axis from the current position to a specified target position.

The target position is directly specified by its distance, in pulses, from the current position of the axis at the time of execution.

The velocity of the axis will follow a trapezoidal profile:



NOTE: If a move relative command is sent while the axis is not referenced, the command is accepted and the position is first set to 0 before executing the command. However, the axis remains unreferenced.

Physical Inputs/Output

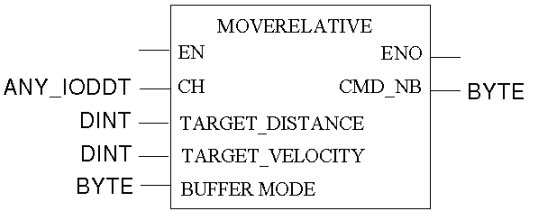
Input/Output	Description
Drive_Ready&Emergency input (optional)	The pulse output is generated as long as a current goes through Drive_Ready&Emergency input. *(see page 224)*
Proximity&LimitSwitch input (optional)	Used as a LimitSwitch. *(see page 224)*
Counter_in_Position input (optional)	Only for information. Input from the drive goes high when positioning movement is completed (the drive's error counter is empty).
Enable_Drive output:	To be connected to the corresponding input of the drive. Enables the drive when active. This output is directly controlled by the user through an implicit command object (%Qr.m.c.0).

Configuration Parameters

Parameter	Valid Values
PTO Output Mode	Value 0: Pulse + Direction (Default) Value 1: CW/CCW Value 2: A/B Phases Value 3: Pulse + Direction – Reverse Value 4: CW/CCW – Reverse Value 5: A/B Phases – Reverse
Acceleration / Deceleration Unit	ms or Hz/2ms Default is ms

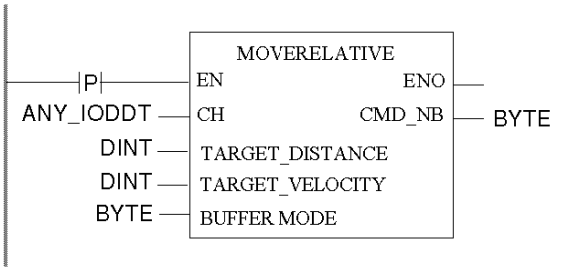
Representation in FBD

Representation:



Representation in LD

Representation:



⚠ WARNING

UNINTENDED INVALID COMMAND

Commands will be sent on every PLC cycle if EN is set to 1. *(see page 119)*

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

Representation in IL

Representation:

```
MOVERELATIVE (CH := (*ANY_IODDT*), TARGET_DISTANCE :=
(*DINT*), TARGET_VELOCITY := (*DINT*), BUFFERMODE := (*BYTE*))
ST (*BYTE*)
```

Representation in ST

Representation:

```
(*BYTE*) := MOVERELATIVE (CH := (*ANY_IODDT*), TARGET_DISTANCE
:= (*DINT*), TARGET_VELOCITY := (*DINT*), BUFFERMODE :=
(*BYTE*));
```

Command example using the WRITE_CMD command mechanism in ST representation:

```
if (ChangePos = True) then  %CH0.1.0.CMD_CODE := 4;
%CH0.1.0.TGT_VELOCITY := 5000; %CH0.1.0.TGT_POSITION := 50000;
%CH0.1.0.BUFFER_MODE :=1; WRITE_CMD(%CH0.1.0); ChangePos :=
False; end_if;
```

Command Specific Parameters

Parameter	Valid Values
Target Distance (in pulses)	- 2,147,483,648 to 2,147,483,647 Must be enclosed between SW Low Limit and SW High Limit
Target Velocity (in Hz)	1 Hz to 200 kHz Absolute value limited by Max Frequency
Buffer mode	Value 0: Abort Value 1: Buffered Value 2: BlendingPrevious

Adjustment Parameters

Parameter	Valid Values
Hysteresis (Slack)	0 to 255 pulses, default is 0 For A/B Phase output mode only (Normal or Reverse)
Start Frequency (in Hz)	0 Hz to 65,535 Hz Default is 0Hz, limited by Max Frequency
Stop Frequency (in Hz)	0 Hz to 65,535 Hz default is 0Hz, limited by Max Frequency
Acceleration Rate	10 to 32,500 Default is 100, limited by Max Acceleration
Deceleration Rate	10 to 32,500 Default is 100, limited by Max Deceleration
Emergency Deceleration Rate	10 to 32,500 Default is 100, limited by Max Deceleration
Software High Limit (in pulses)	-2,147,483,647 to 2,147,483,647 Default is 2,147,483,647 Must be between SW Low Limit and SW Max High Limit
Software Low Limit (in pulses)	-2,147,483,648 to 2,147,483,646 Default is - 2,147,483,648 Must be enclosed between SW Min Low Limit and SW High Limit

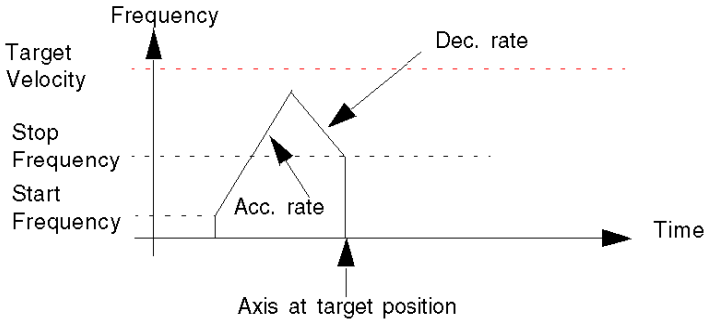
Overall Parameters

This table describes all the functional parameters associated to the function.

Explicit Command Parameters		Setting Parameters		Adjustment Parameters	
Address	**Parameter**	**Address**	**Parameter**	**Address**	**Parameter**
%MWr.m.c.6 (byte 0)	Command Code (=4)	%KWr.m.c.1 (byte 0)	Output Mode	%MWr.m.c.18	Start Frequency
%MWr.m.c.7 (byte 0)	Buffer Mode	%KWr.m.c.1 (byte 12)	Acc/Dec Unit	%MWr.m.c.19	Stop Frequency
%MDr.m.c.8	Target Distance	%KWr.m.c.4	Acc Max	%MWr.m.c.20	Acceleration Rate
%MDr.m.c.10	Target Velocity	%KWr.m.c.5	Dec Max	%MWr.m.c.21	Deceleration Rate
		%KDr.m.c.6	FMax	%MWr.m.c.25	Hysteresis

Special cases

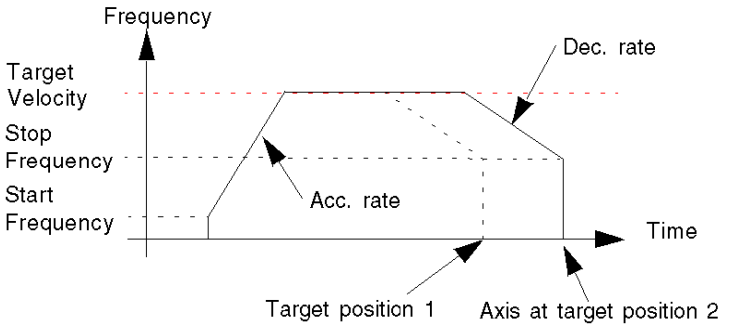
If the set target velocity cannot be reached before attaining the target position, the axis velocity will then follow a triangular profile:



Positioning Complex Profile 1

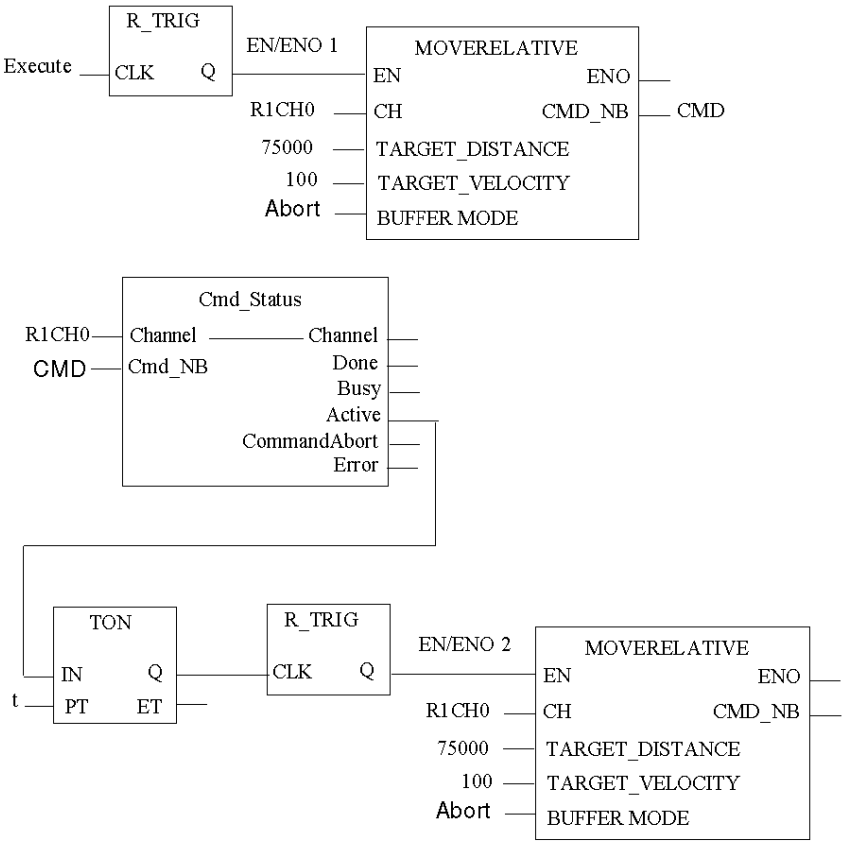
At a Glance

While executing a positioning command, it is possible to modify the target position on the fly:



FBD Program

Program to obtain the above profile

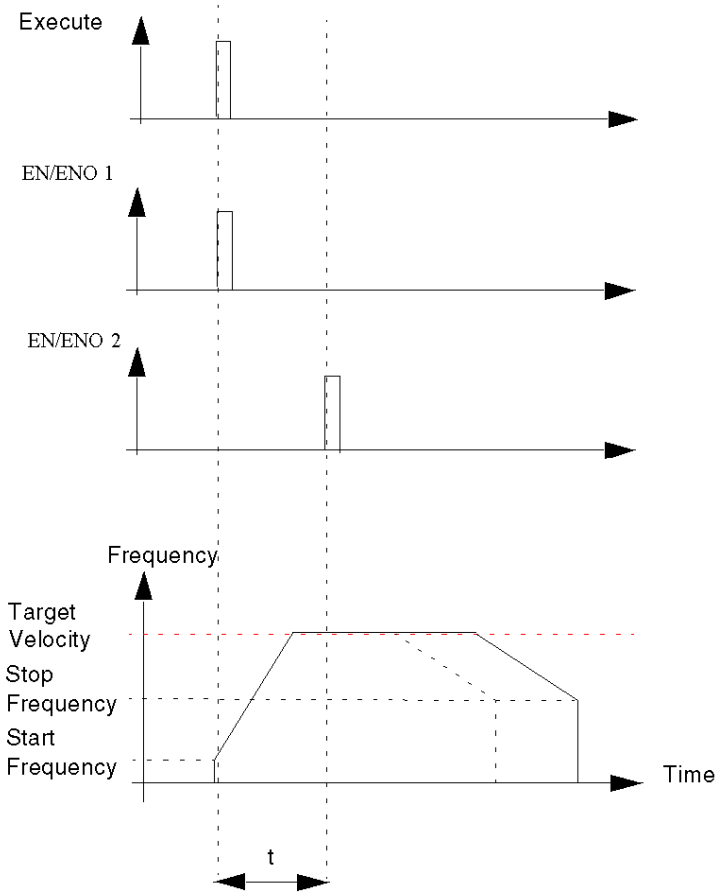


R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Cmd_Status is the command status follow up *(see page 203)* function.

Time Diagram

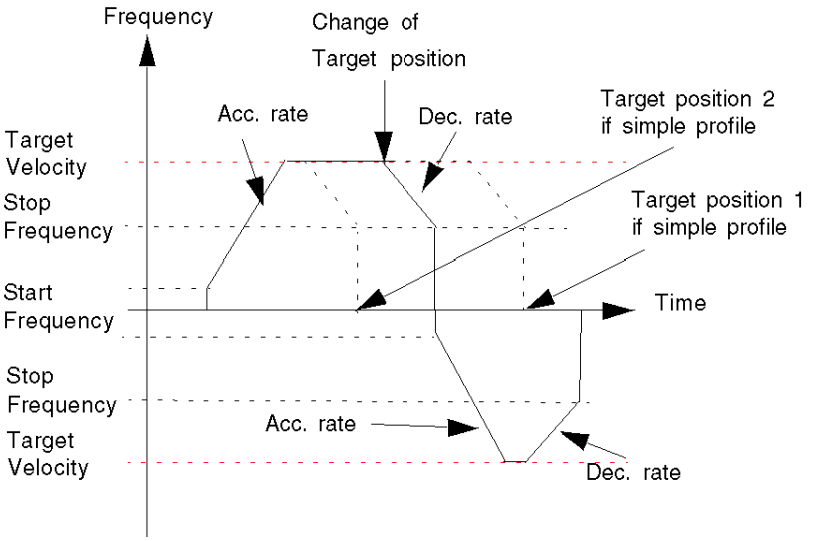
Time diagram of the MOVERELATIVE Input / Output:



Positioning Complex Profile 2

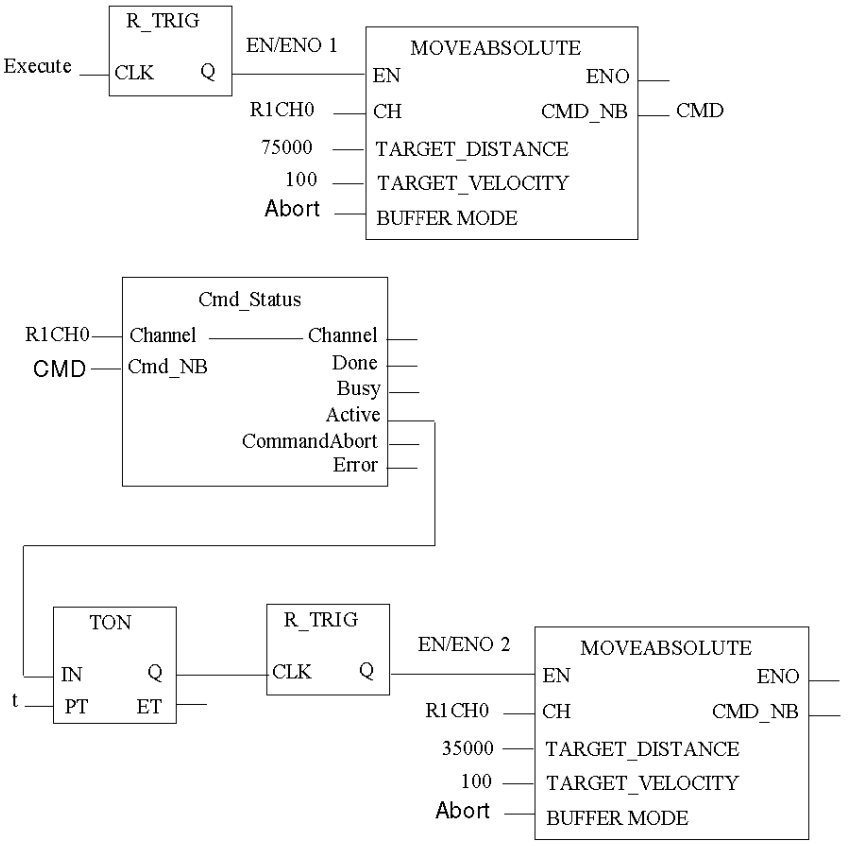
At a Glance

In some cases, the axis has already gone past the new target position, this will require the axis to stop and change direction:



FBD Diagram

Program to obtain the above profile

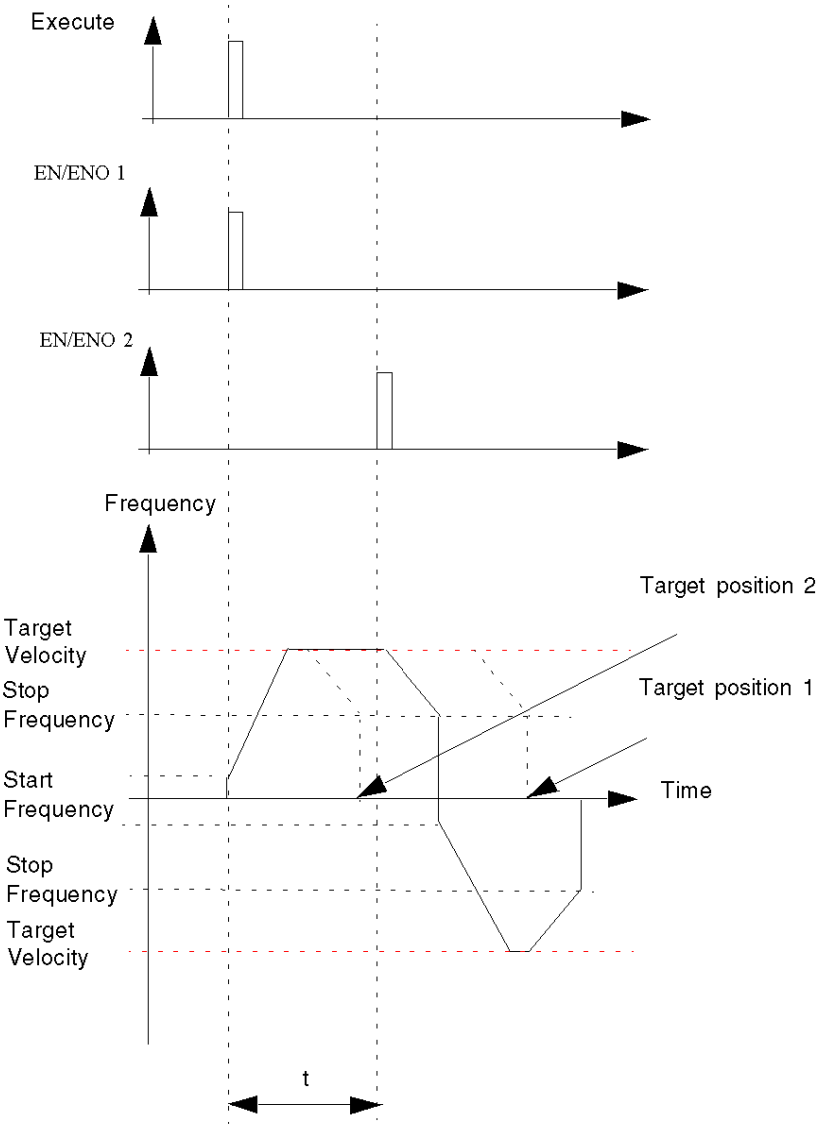


R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Cmd_Status is the command status follow up function. *(see page 203)*

Time Diagram

Time diagram of the MOVERELATIVE Input / Output:



Positioning Buffer Mode Management

At a Glance

While a positioning command is running, it is possible to send a new command. The sequence of those two commands can be managed in three different ways according to the BufferMode parameter of the new command:

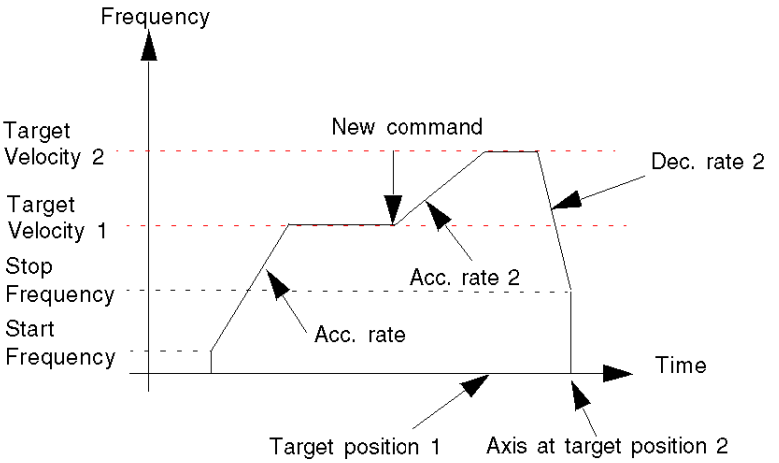
- Abort: the new command aborts the previous one and is executed immediately.
- Buffered: the new command is put into a buffer and executed only once the current command is completed. The current command ends normally (stops when reaching the target position).
- BlendingPrevious: the new command is put into a buffer and executed only once the target position of the current command is reached. However, the axis does not stop between both commands and the velocity is blended with the target velocity of the current command (see diagram below).

Positioning Buffer Mode Abort Case

At a Glance

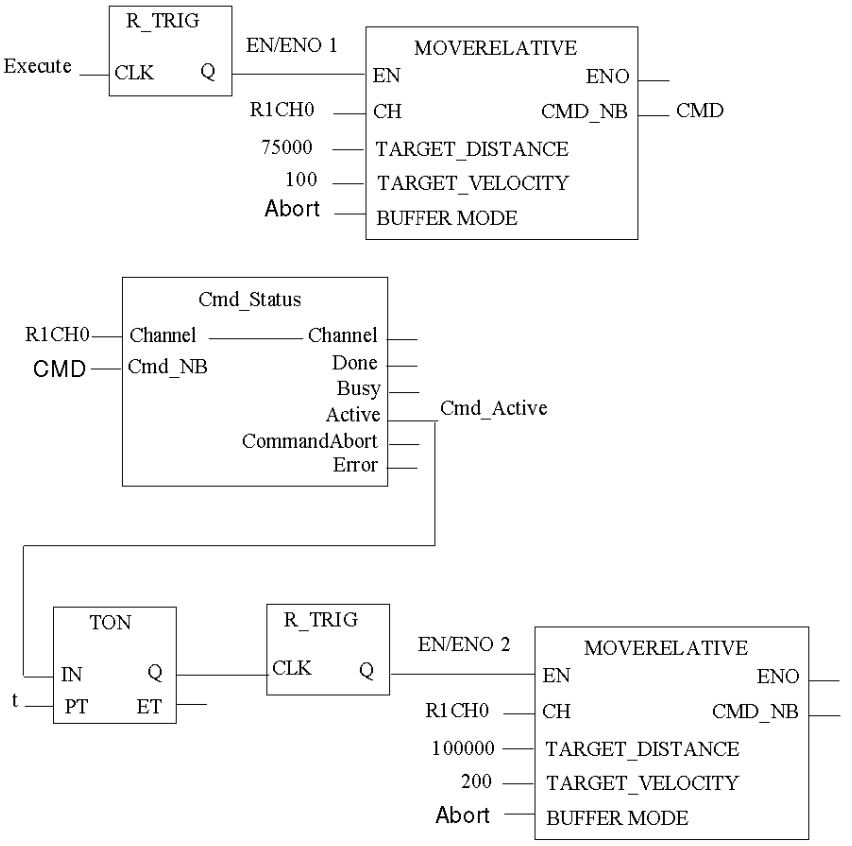
The new command aborts the previous one and is executed immediately.

Case of Abortion



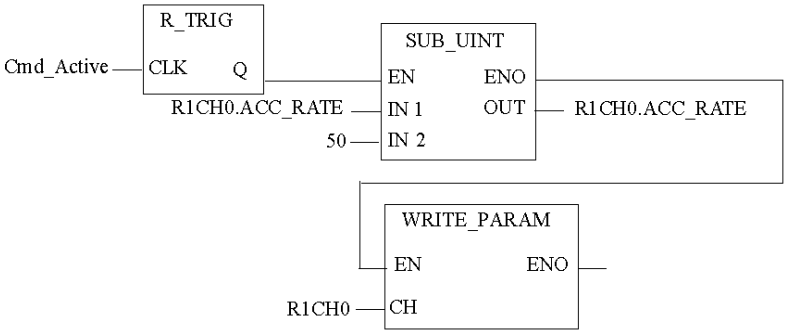
FBD Program

Program to obtain the above profile



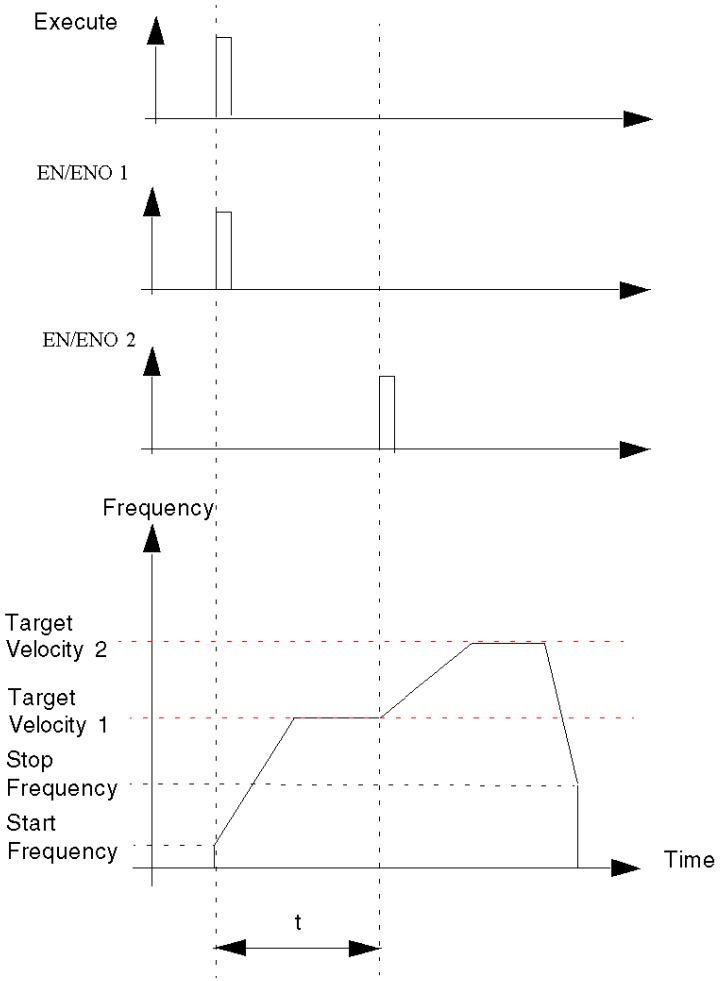
R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Cmd_Status is the command status follow up function. *(see page 203)*


R_TRIG
Cmd_Active
CLK
Q
SUB_UINT
EN
ENO
R1CH0.ACC_RATE
IN 1
OUT
R1CH0.ACC_RATE
50
IN 2
WRITE_PARAM
EN
ENO
R1CH0
CH

Time Diagram

Time diagram of the MOVERELATIVE Input / Output:

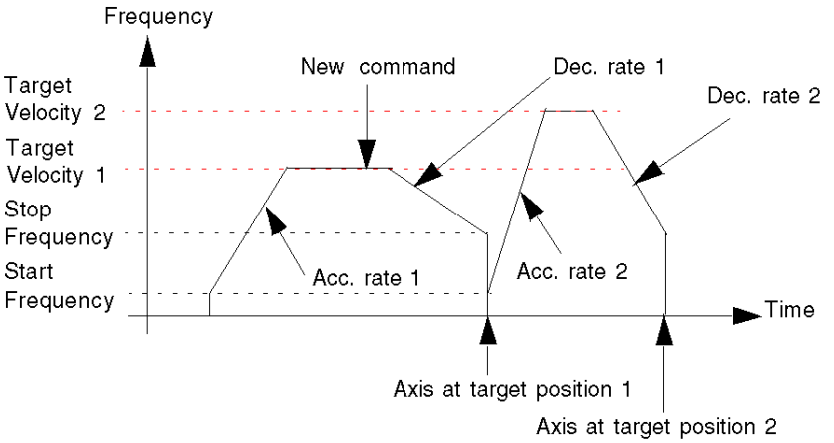


Positioning Buffer Mode Buffered Case

At a Glance

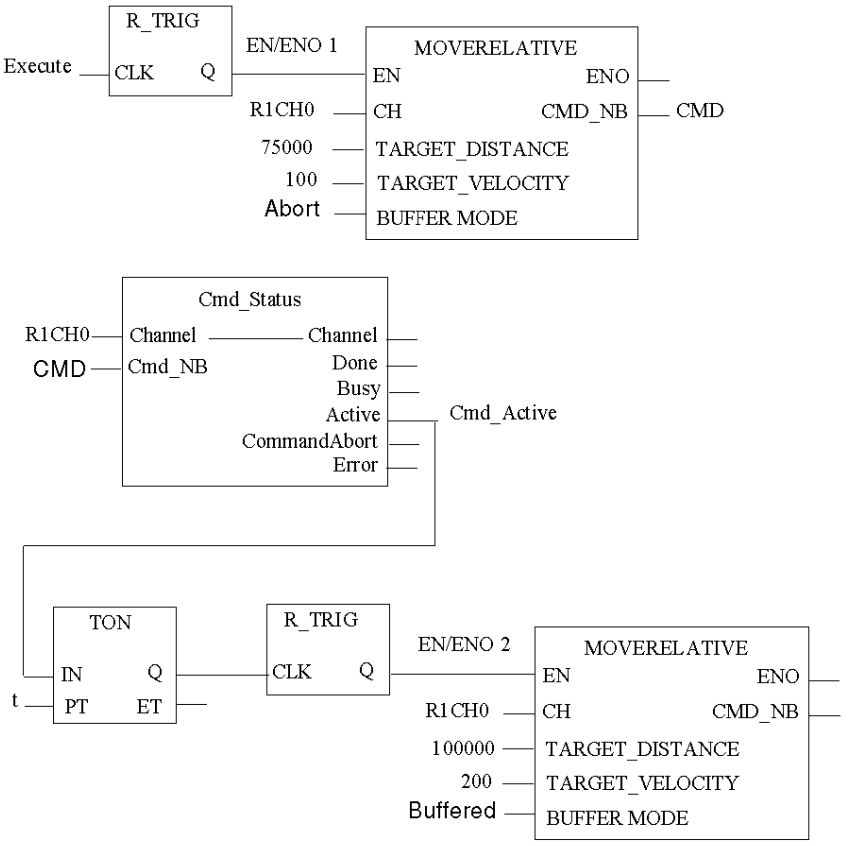
The new command is put into a buffer and executed only after the current command is completed. The current command ends normally (stops when reaching the target position).

Case of Bufferizing



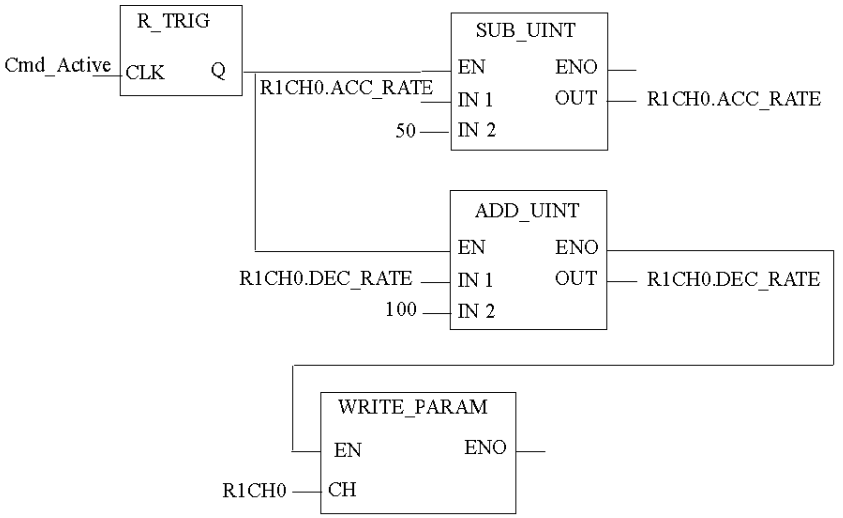
FBD Program

Program to obtain the above profil



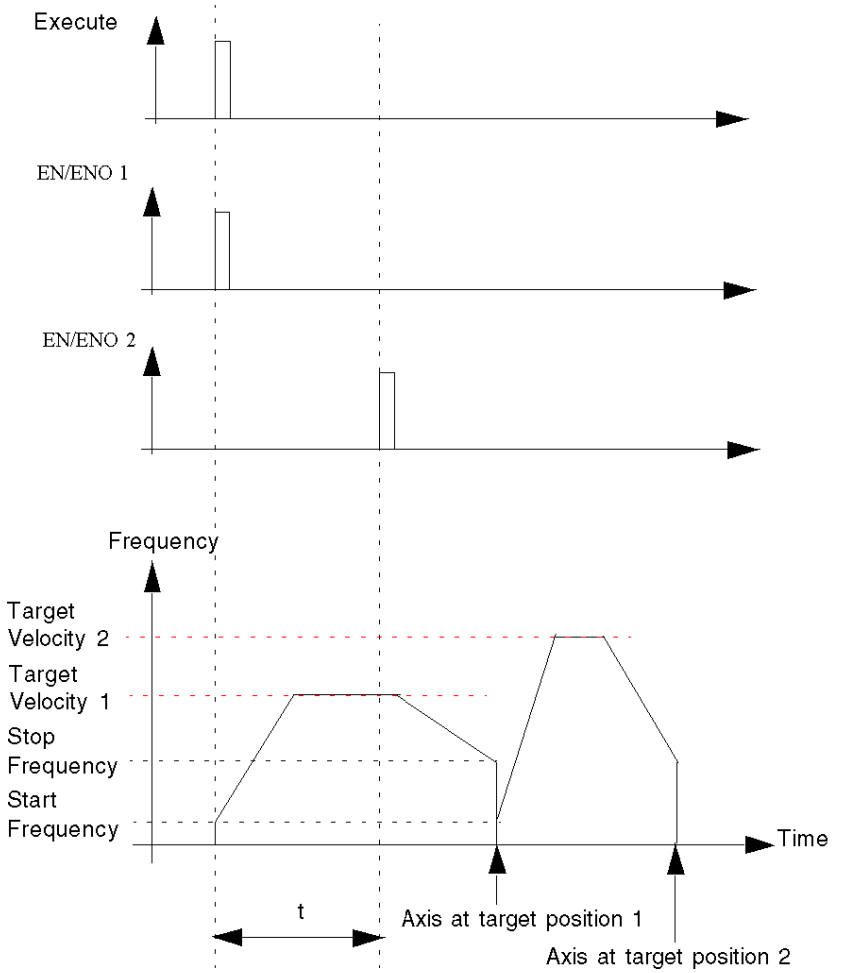
R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Cmd_Status is the command status follow up function. *(see page 203)*


R_TRIG
Cmd_Active
CLK
Q
SUB_UINT
EN
ENO
R1CH0.ACC_RATE
IN 1
OUT
R1CH0.ACC_RATE
50
IN 2
ADD_UINT
EN
ENO
R1CH0.DEC_RATE
IN 1
OUT
R1CH0.DEC_RATE
100
IN 2
WRITE_PARAM
EN
ENO
R1CH0
CH

Time Diagram

Time diagram of the MOVERELATIVE Input / Output



Positioning Buffer Mode Case of BlendingPrevious

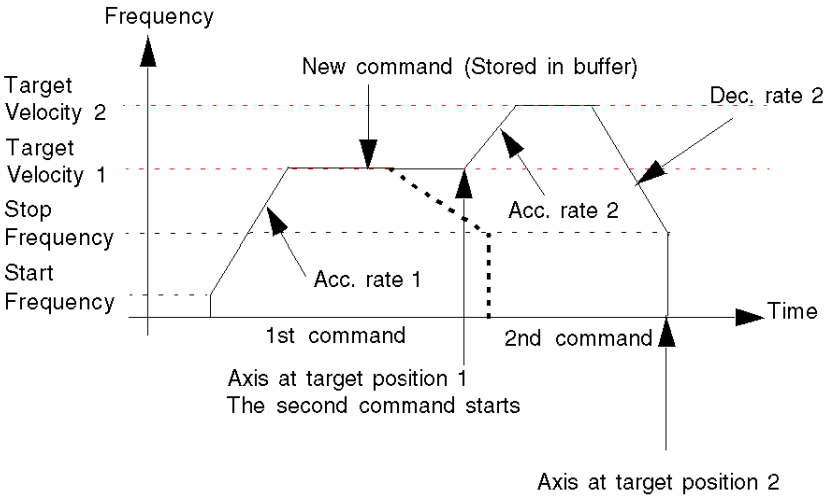
At a Glance

For the BlendingPrevious buffer mode, there can be two different cases:

- the second command is received during the acceleration or constant velocity phase of the previous command
- the second command is received during the stopping phase of the previous command

1st Case Overview

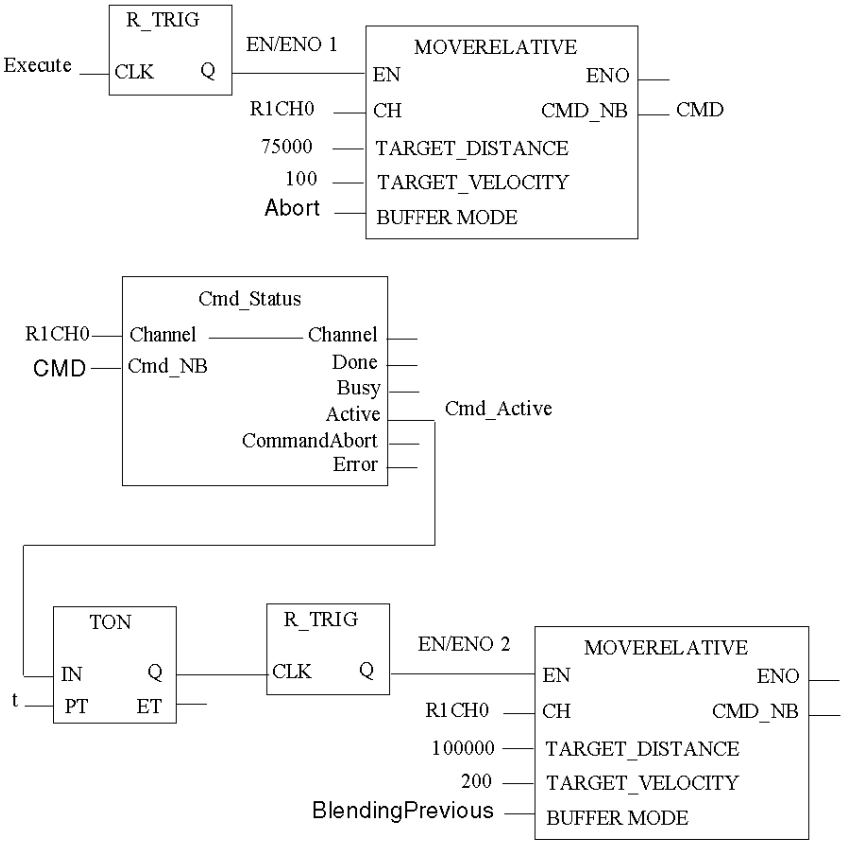
The new command is received by the PTO module during the acceleration phase or constant velocity phase of the previous command. As soon as the first target position is reached, the execution of the second command starts at the Target_Velocity of the previous command:



If there was no second command, the frequency profile would have followed the thick dotted line.

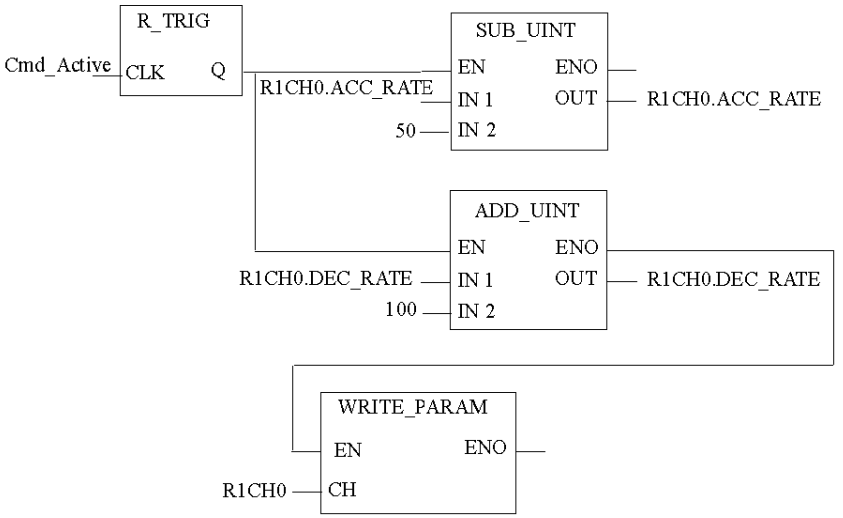
1st Case FBD Diagram

Program to obtain the above profile



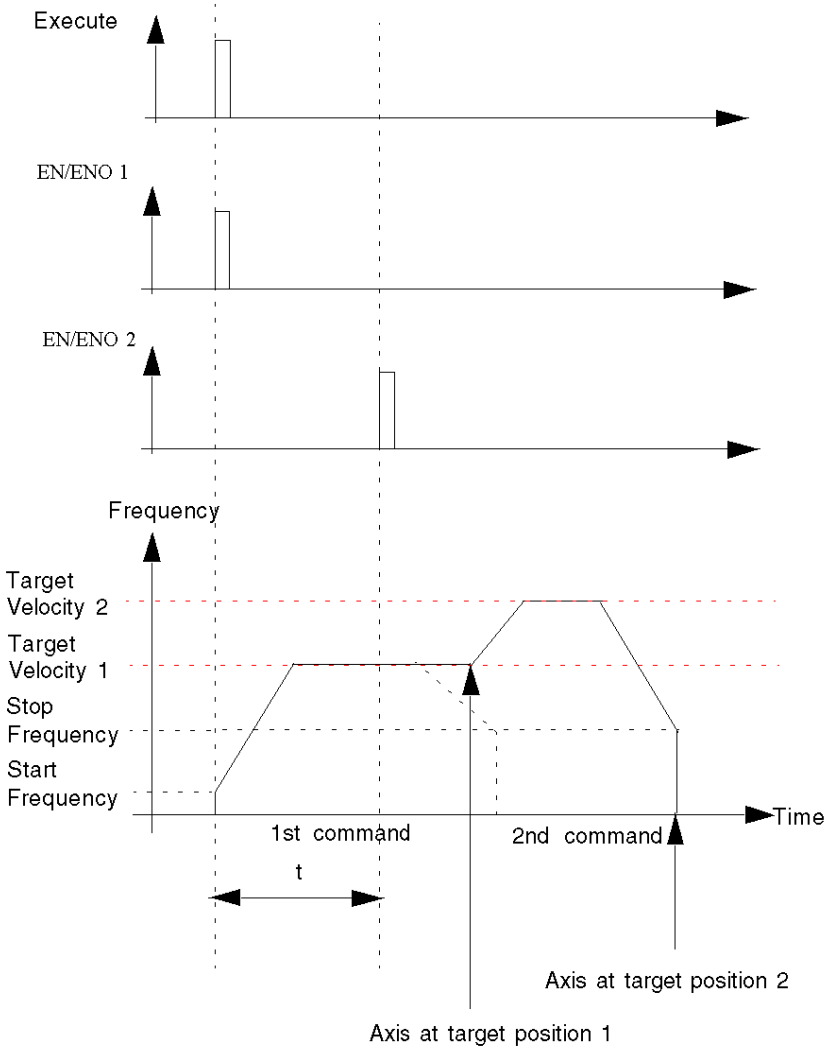
R1CH0 = %CH0.1.0
(PTO module on rack 1, channel 0 configured for position control)

Cmd_Status is the command status follow up function. *(see page 203)*


R_TRIG
Cmd_Active
CLK
Q
SUB_UINT
EN
ENO
R1CH0.ACC_RATE
IN 1
OUT
R1CH0.ACC_RATE
50
IN 2
ADD_UINT
EN
ENO
R1CH0.DEC_RATE
IN 1
OUT
R1CH0.DEC_RATE
100
IN 2
WRITE_PARAM
EN
ENO
R1CH0
CH

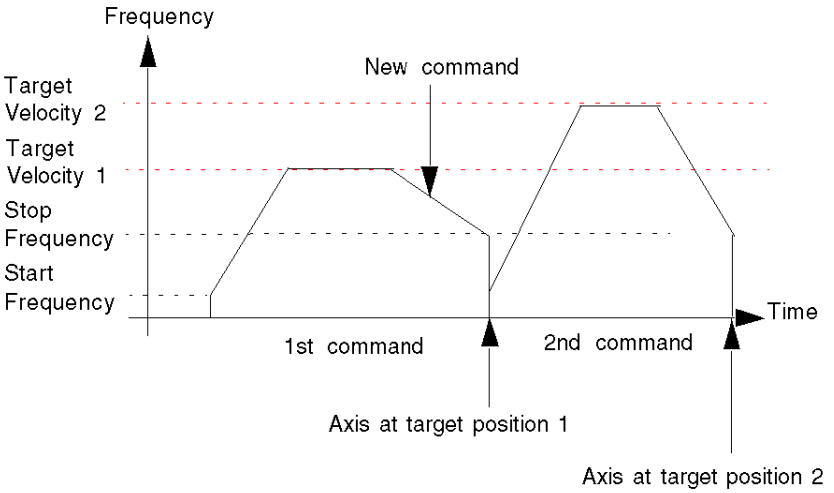
1st Case Time Diagram

Time diagram of the MOVERELATIVE Input / Output



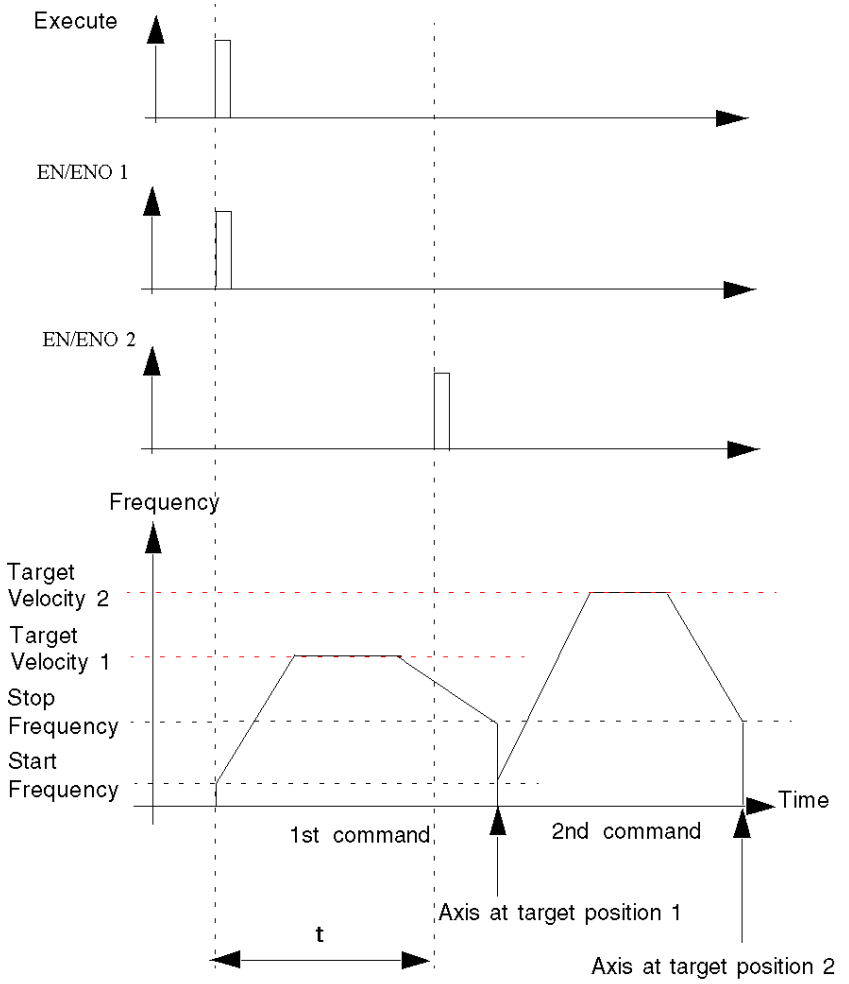
2nd Case Overview

If the new command is received by the PTO channel during the stopping phase of the previous command, the sequence of the two commands is executed as "Buffered".



2nd Case Time Diagram

Time diagram of the MOVERELATIVE Input / Output



Homing

Description

This function commands the axis to search for a reference point set by input signals, and to stop at this reference point.

When the homing sequence is completed:

- The reference point's coordinate is set to the position value (parameter of the homing command)
- The channel "REFERENCED" status bit is set to 1 which activates software limits if not disabled.

There are different homing modes, depending on the physical configuration of the controlled machine. The mode to be used is chosen via the "Homing Type" parameter (cf. description of each type below).

Physical Inputs/Outputs

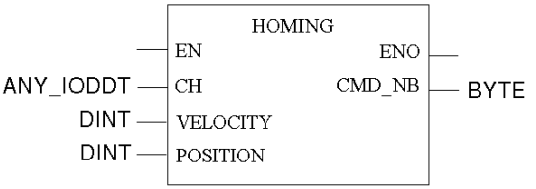
Input/Output	Description
Drive_Ready&Emergency input (optional)	The pulse output is generated as long as a current goes through Drive_Ready&Emergency input. *(see page 224)*
Proximity&LimitSwitch input (optional)	This input can be used in two ways: • as proximity signal for the homing profile and detailed below within the description of each homing mode: • as a LimitSwitch. *(see page 224)*
Counter_in_Position input (optional)	For information, input from the drive goes high when positioning movement is completed (the drive's error counter is empty). According to configuration, this input can also be used for the homing process. See below Homing I/O Settings description.
Origin Input	Detailed within the description of each homing mode.
Drive_Enable output:	To be connected to the corresponding input of the drive. Enables the drive when active. This output is directly controlled via an implicit command object (%Qr.m.c.0).
Counter_Clear output	See Homing I/O Settings description To be connected to the corresponding input of the drive. Orders a reset of the drive internal error counter

Configuration Parameters

Parameter	Valid Values
PTO Output Mode	Value 0: Pulse + Direction (Default) Value 1: CW/CCW Value 2: A/B Phases Value 3: Pulse + Direction – Reverse Value 4: CW/CCW – Reverse Value 5: A/B Phases – Reverse
Acceleration / Deceleration Unit	ms or Hz/2ms Default is ms
Homing Type	Value 0: Short Cam (Default) Value 1: Long Cam Positive Value 2: Long Cam Negative Value 3: Short Cam with Positive Limit Value 4: Short Cam with Negative Limit Value 5: Short Cam with Marker
Homing I/O Settings	Value 0: No I/O used (Default) Value 1: With Counter_Clear Output Value 2: With Counter_in_Position Input

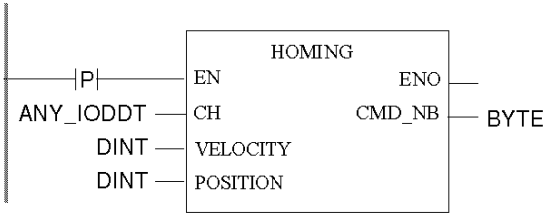
Representation in FBD

Representation:



Representation in LD

Representation:



⚠ WARNING

UNINTENDED INVALID COMMAND

Commands will be sent on every PLC cycle if EN is set to 1. *(see page 119)*

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

Representation in IL

Representation:

```
HOMING (CH := (*ANY_IODDT*), POSITION := (*DINT*), VELOCITY :=
(*DINT*))
ST (*BYTE*)
```

Representation in ST

Representation:

```
(*BYTE*) := HOMING (CH := (*ANY_IODDT*), POSITION := (*DINT*),
VELOCITY := (*DINT*));
```

Command Specific Parameters

Parameter	Valid Values
Target position (in pulses)	- 2,147,483,648 to 2,147,483,647 Must be enclosed between SW Low Limit and SW High Limit
Velocity (in Hz)	-200 kHz to 200 kHz (≠0) Absolute value limited by Max Frequency

Adjusment Parameters

Parameter	Valid Values
Hysteresis (Slack)	0 to 255 pulses Default is 0 For A/B Phase output mode only (Normal or Reverse)
Start Frequency (in Hz)	0 Hz to 65,535 Hz Default is 0Hz, limited by Max Frequency
Stop Frequency (in Hz)	0 Hz to 65,535 Hz Default is 0Hz, limited by Max Frequency
Acceleration Rate	10 to 32,500 Default is 100, limited by Max Acceleration
Deceleration Rate	10 to 32,500 Default is 100, limited by Max Deceleration
Emergency Deceleration Rate	10 to 32,500 Default is 100, limited by Max Deceleration
Software High Limit (in pulses)	-2,147,483,647 to 2,147,483,647 Default is 2,147,483,647 Must be between SW Low Limit and SW Max High Limit
Software Low Limit (in pulses)	-2,147,483,648 to 2,147,483,646 Default is - 2,147,483,647 Must be enclosed between SW Min Low Limit and SW High Limit
Homing Velocity (in Hz)	1 Hz to 65,535 Hz Default is 1Hz, limited by Max Frequency Must be ≥ Start Frequency (if enabled) Must be ≥ Stop Frequency (if enabled)
Homing Time Out Value	0 to 65,535 ms Default is 65,535 ms

NOTE: For a detailed explanation on how to keep consistency between parameters, please refer to parameter description section. *(see page 123)*

Overall Parameters

Explicit Command Parameters		Setting Parameters		Adjustment Parameters	
Adress	**Parameter**	**Adress**	**Parameter**	**Adress**	**Parameter**
%MWr.m.c.6 (byte 0)	Command CodeValue (=5)	%KWr.m.c.1 (byte 0)	Output Mode	%MDr.m.c.14	SW High Limit
%MDr.m.c.8	Target Position	%KWr.m.c.1 (byte 10 & 11)	Homing I/O Settings	%MDr.m.c.16	SW Low Limit
%MDr.m.c.10	Target Velocity	%KWr.m.c.1 (byte 12)	Acc/Dec Unit	%MWr.m.c.18	Start Frequency
		%KWr.m.c.4	Acc Max	%MWr.m.c.19	Stop Frequency
		%KWr.m.c.5	Dec Max	%MWr.m.c.20	Acceleration Rate
		%KDr.m.c.6	FMax	%MWr.m.c.21	Deceleration Rate
		%KDr.m.c.8	SW Max High Limit	%MWr.m.c.23	Homing Velocity
		%KDr.m.c.10	SW Min Low Limit	%MWr.m.c.24	Homing Time Out Value
		%KWr.m.c.12	Homing Type	%MWr.m.c.25	Hysteresis

General Homing Features

At a Glance

There are 6 homing modes:

- Short Cam *(see page 191)*
- Long Cam Positive *(see page 192)*
- Long Cam Negative *(see page 193)*
- Short Cam with Positive Limit *(see page 194)*
- Short Cam with Negative Limit *(see page 196)*
- Short Cam with Marker *(see page 198)*

Each homing mode has two velocities: a high velocity, which is set as a command parameter (Velocity), and a low velocity, used to get to the referenced point, set by adjustment (Homing Velocity).

Homing I/O Settings

Homing I/O settings

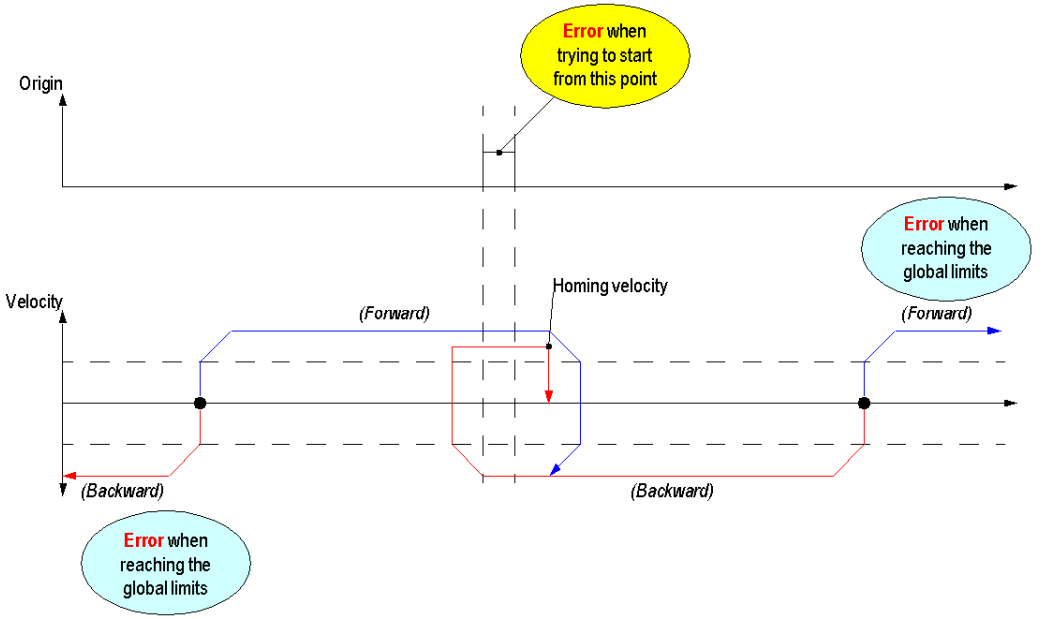
- When the Counter_Clear output is enabled (value 1):
  In order to synchronize the PTO channel and the drive, a pulse is sent on the Counter_Clear output.
  When the homing condition is reached, the channel's internal counter is set to the specified position value and the output frequency is stopped.
  The channel "REFERENCED" status bit is then set to 1.
- When the Counter_in_Position input is enabled (value 2):
  After the homing condition is reached, the output frequency is stopped.
  In order to synchronize the PTO channel and the PTO drive, the homing command remains running (BUSY state) until a rising edge of the Counter_in_Position input is detected. The channel's internal counter is then set to the specified position value and the channel "REFERENCED" status bit is set to 1.
  A homing function error is reported if Counter_in_Position remains low after a certain duration (time-out value to be configured in setting parameters) by rising the HOMING_FLT bit (%MWr.m.c.5.4) and the AXIS_FLT bit (%IWr.m.c.6.3).
- When no specific I/O are used for the homing process (value 0):
  When the homing condition is reached, the channel's internal counter is set to the specified position value and the output frequency is stopped.
  The channel "REFERENCED" status bit is then set to 1.
  Synchronization between the PTO channel and the PTO drive cannot be assumed because the end of the homing process is defined internally in the module, independently from any feedback from the drive.

For all homing modes described in the following sections, the direction (FORWARD, BACKWARD) is given by the sign of Velocity, specified in the homing command.

Homing Mode: Short Cam

Short Cam

In the Short Cam homing mode, the reference point is preset at the negative side of the cam, when coming in positive direction (off cam) at low velocity.



Inputs used:

- The Short Cam homing mode only uses the Origin input (Cam).

Detected errors that can be encountered:

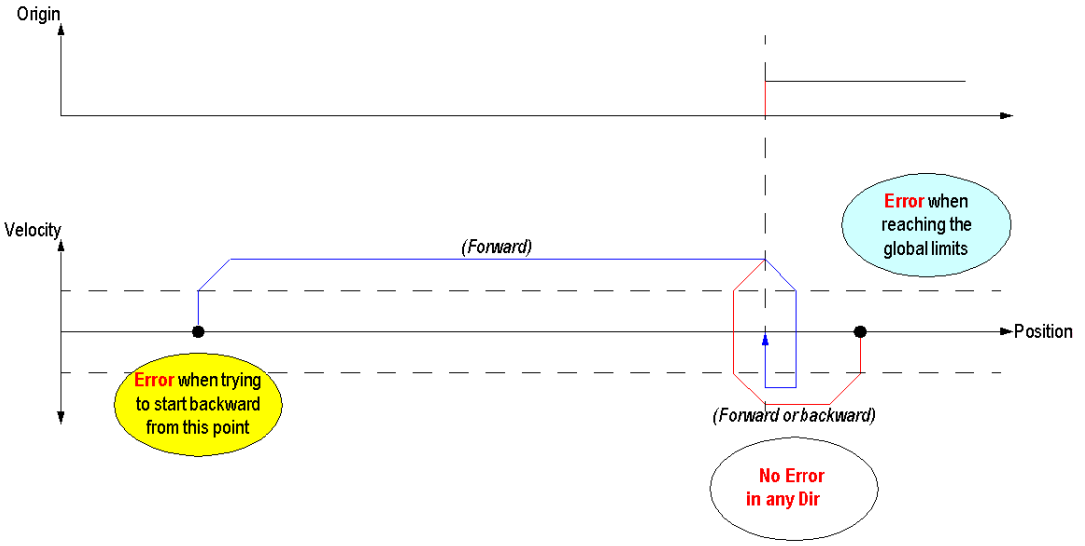
- If a limit is bypassed and detected with Proximity&LimitSwitch input (if not disabled), the detected error is reported in the LIMIT_FLT status object (%MWr.m.c.5.1).
- If the axis is already on the cam at start, the homing function will not be executed and the detected error is reported in the HOMING_FLT status object (%MWr.m.c.5.4).
- If Drive_Ready&Emergency goes off (if not disabled), the detected error is reported in the DRIVE_KO status object (%MWr.m.c.5.0).

The detected errors are also reported in the AXIS_FLT implicit status object (%IWr.m.c.6.3).

Homing Mode: Long Cam Positive

Long Cam Positive

In Long Cam Positive homing mode, the reference point is preset at the negative side of the cam, when coming in negative direction (from the cam) at low velocity.



Inputs used:

- The Long Cam Positive homing mode only uses the Origin input (Cam).

Detected errors that can be encountered:

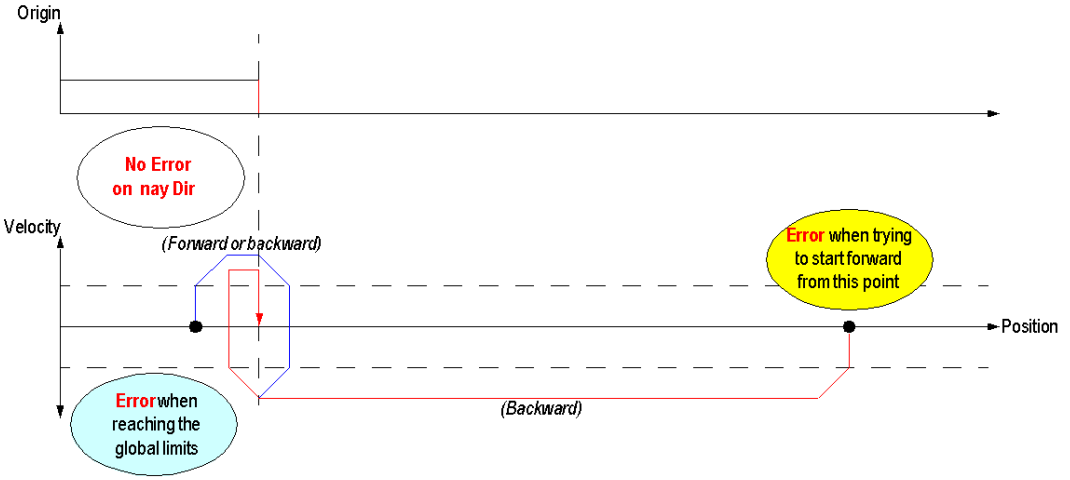
- If a limit is bypassed and detected with Proximity&LimitSwitch input (if not disabled), the detected error is reported in the LIMIT_FLT status object (%MWr.m.c.5.1).
- If the axis is off the cam and direction is set backward (negative velocity), the homing function will not be executed and the detected error will be reported in the HOMING_FLT status object (%MWr.m.c.5.4).
- If Drive_Ready&Emergency goes off (if not disabled), the detected error is reported in the DRIVE_KO status object (%MWr.m.c.5.0).

The detected errors are also reported in the AXIS_FLT implicit status object (%IWr.m.c.6.3).

Homing Mode: Long Cam Negative

Long Cam Negative

In the Long Cam Negative homing mode, the reference point is preset at the positive side of the cam, when coming in positive direction (from the cam) at low velocity.



Inputs used:

- The Long Cam Negative homing mode only uses the Origin input (Cam).

Errors that can be encountered:

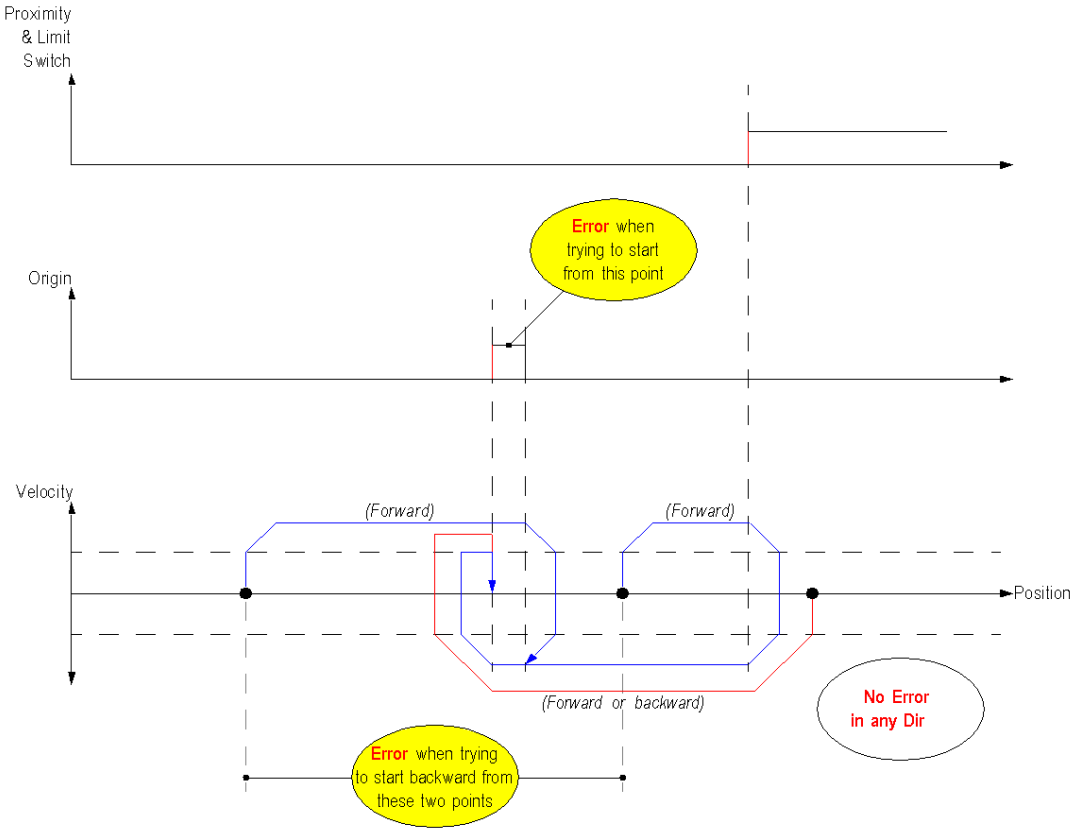
- If a limit is bypassed and detected with Proximity&LimitSwitch input (if not disabled), an error is reported in the LIMIT_FLT status object (%MWr.m.c.5.1).
- If the axis is off the cam and direction is set forward (positive velocity), the homing function will not be executed and an error will be reported in the HOMING_FLT status object (%MWr.m.c.5.4).
- If Drive_Ready&Emergency goes off (if not disabled), an error is reported in the DRIVE_KO status object (%MWr.m.c.5.0).

The error is also reported in the AXIS_FLT implicit status object (%IWr.m.c.6.3).

Homing Profile: Short Cam with Positive Limit

Short Cam with Positive Limit

In the Short Cam with Positive Limit homing mode, the reference point is preset at the negative side of the cam, when coming in positive direction (off cam) at low velocity.



The Short Cam with Positive Limit homing mode uses the two homing-specific inputs:

- The Proximity&LimitSwitch input: used as the positive limit signal. On the rising edge of the signal (negative side), the axis decelerates to change direction.
- The Origin (Cam) input.

Detected errors that can be encountered:

- If the axis is already on the cam at start, the homing function will not be executed and the detected error is reported in the HOMING_FLT status object (%MWr.m.c.5.4).
- When the axis is inside the working area (delimited by LimitSwitch signal) and direction is set backward (negative velocity), the homing function will not be executed and the detected error will be reported in the HOMING_FLT status object (%MWr.m.c.5.4).
- If Drive_Ready&Emergency goes off (if not disabled and Drive_Enable output is active), the detected error is reported in the DRIVE_KO status object (%MWr.m.c.5.0).

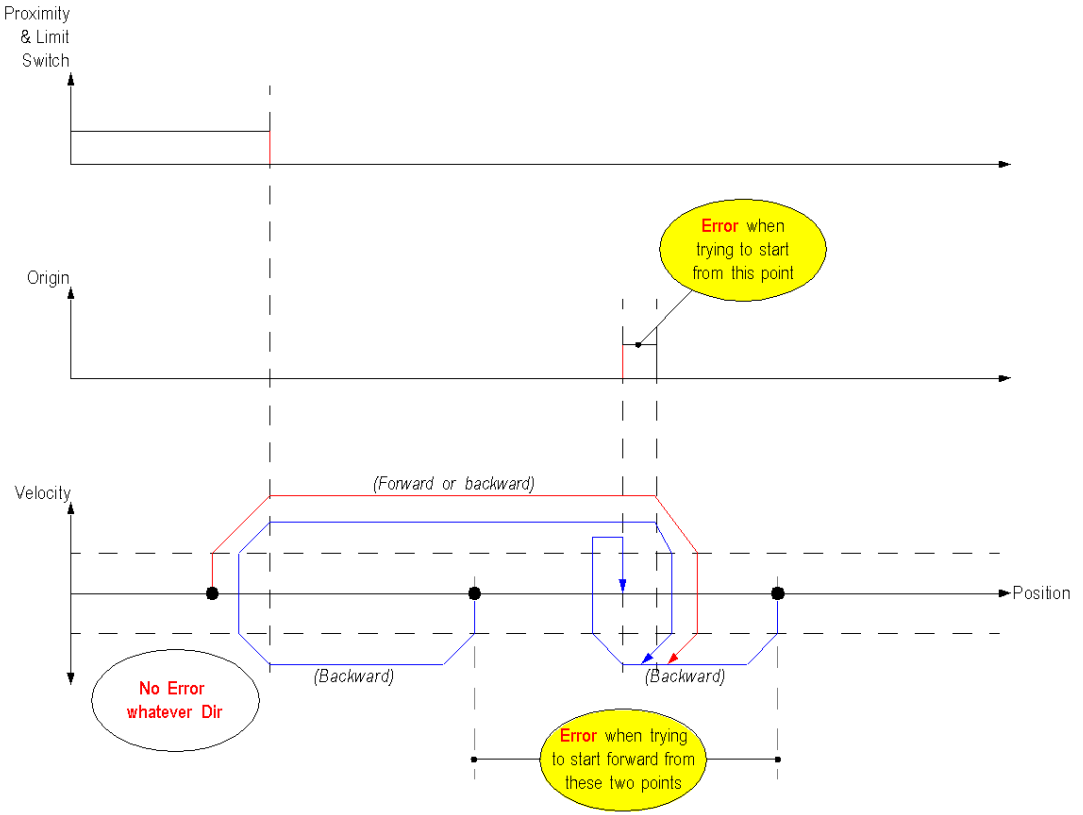
The detected errors are also reported in the AXIS_FLT implicit status object (%IWr.m.c.6.3).

NOTE: During the homing process, the Proximity&LimitSwitch input will not be used as Limit Switch (no detection of limit crossing). For any other command, this input can still be used as Limit Switch input.

Homing Mode: Short Cam with Negative Limit

Short Cam with Negative Limit

In the Short Cam with Negative Limit homing mode, the reference point is preset at the negative side of the cam, when coming in positive direction (off cam) at low velocity.



The Short Cam with Negative Limit homing mode uses the two homing-specific inputs:

- The Proximity&LimitSwitch input: used as the negative limit signal. On the rising edge of the signal (positive side), the axis decelerates to change direction.
- The Origin (Cam) input.

Detected errors that can be encountered:

- If the axis is already on the cam at start, the homing function will not be executed and the detected error is reported in the HOMING_FLT status object (%MWr.m.c.5.4).
- When the axis is inside the working area (delimited by LimitSwitch signal) and direction is set forward (positive velocity), the homing function will not be executed and the detected error will be reported in the HOMING_FLT status object (%MWr.m.c.5.4).
- If Drive_Ready&Emergency goes off (if not disabled and Drive_Enable output is active), the detected error is reported in the DRIVE_KO status object (%MWr.m.c.5.0).

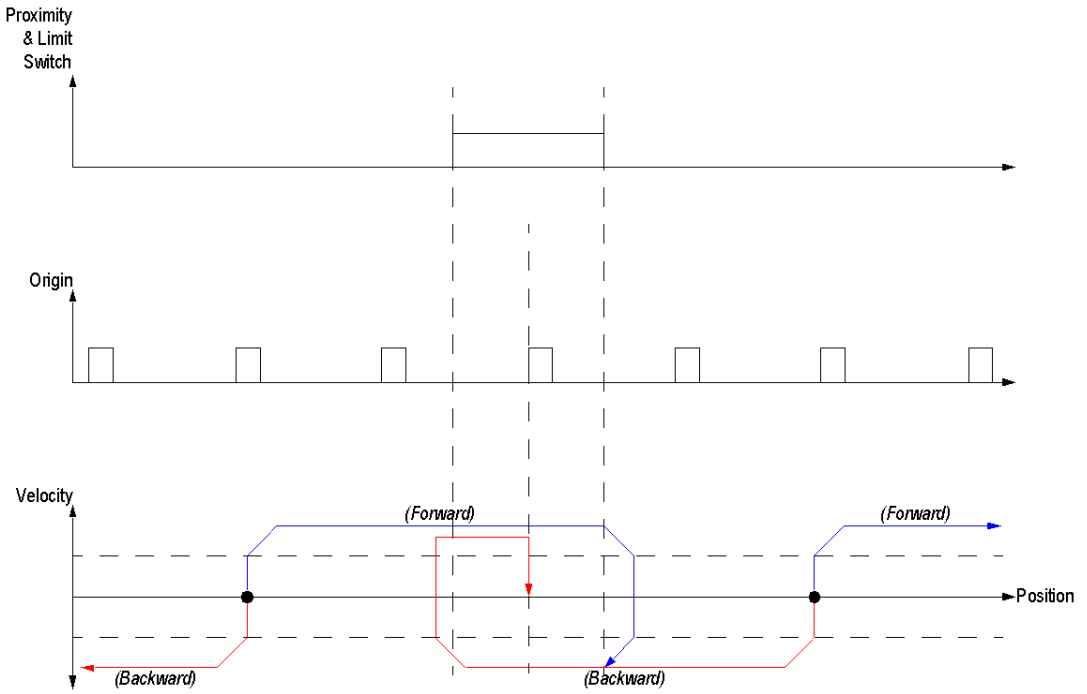
The detected errors are also reported in the AXIS_FLT implicit status object (%IWr.m.c.6.3).

NOTE: During the homing process, the Proximity&LimitSwitch input will not be used as Limit Switch (no detection of limit crossing). For any other command, this input can still be used as Limit Switch input.

Homing Mode: Short Cam with Marker

Short Cam with Marker

In the Short Cam with Marker homing mode, the reference point is preset at the negative side of the zero marker, when coming in positive direction at low velocity.



The Short Cam with Zero Marker homing mode uses the two homing-specific inputs:

- The Proximity&LimitSwitch input: used as the proximity signal. On the falling edge of the signal, the axis decelerates to change direction.
- The Origin input used as Zero Marker signal.

The detected errors that can be encountered:

- If Drive_Ready&Emergency goes off (if not disabled and Drive_Enable output is active), the detected error is reported in the DRIVE_KO status object (%MWr.m.c.5.0).

The detected errors are also reported in the AXIS_FLT implicit status object (%IWr.m.c.6.3).

Limit crossing detection: The Proximity&LimitSwitch input can not be used as a Limit Switch input, either for homing commands or any other command. Instead use the Drive_Ready&Emergency input in order to detect a limit-crossing event.
(see page 32)

Set Position

Description

Contrary to the other motion functions, this function does not impact the physical pulse outputs of the channel, and does not generate any motion profiles.

Like the homing function, it defines an origin and a reference position of the axis by assigning an absolute coordinate to the current position of the axis and setting to 1 the channel "REFERENCED" status bit.

This function can only be used when the axis is in STANDSTILL state.

Physical Inputs/Output

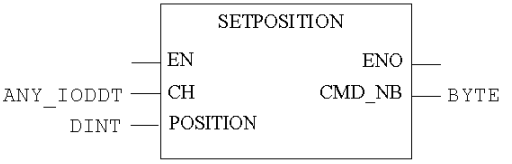
Input/Output	Description
Counter_Clear output	To be connected to the corresponding input of the drive. When the Counter_Clear output is enabled, the Set Position function also orders the drive to reset its internal counter.

Configuration Parameters

Parameter	Valid Values
Homing I/O Settings	Value 0: No I/O used (Default) Value 1: With Counter_Clear Output Value 2: With Counter_in_Position Input: not used with SetPosition command.

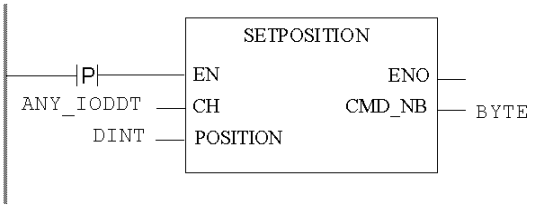
Representation in FBD

Representation:



Representation in LD

Representation:



⚠ WARNING

UNINTENDED INVALID COMMAND

Commands will be sent on every PLC cycle if EN is set to 1. *(see page 119)*

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

Representation in IL

Representation:

```
(*BYTE*) := SETPOSITION (CH := (*ANY_IODDT*), POSITION :=
(*DINT*));
```

Representation in ST

Representation:

```
SETPOSITION (CH := (*ANY_IODDT*), POSITION := (*DINT*)) ST
(*BYTE*)
```

Command example using the WRITE_CMD command mechanism in ST representation:

```
if (SetPos = True) then  %CH0.1.0.CMD_CODE := 6;
%CH0.1.0.TGT_POSITION := 50000; WRITE_CMD(%CH0.1.0); SetPos :=
False; end_if;
```

Command Specific Parameters

Parameter	Valid Values
Position (in Pulses)	- 2,147,483,648 to 2,147,483,647 (Enclosed between SW Low Limit and SW High Limit)

STOP

Description

Whatever the motion in progress, and at whatever stage of the movement, the user can order the axis to stop, smoothly, by going through a deceleration phase. It is also possible to STOP the axis by setting to 0 the Drive ENABLE command, then the moving part is forced to stop through a deceleration phase (equal to Stop command)

Configuration Parameters

Parameter	Valid Values
PTO Output Mode	Value 0: Pulse + Direction (Default) Value 1: CW/CCW Value 2: A/B Phases Value 3: Pulse + Direction – Reverse Value 4: CW/CCW – Reverse Value 5: A/B Phases – Reverse
Deceleration Unit	ms (default) or Hz/2ms

Representation

The stop function does not have any program representation, it can be activated via the debugging screen *(see page 216)* (Stop Level Cmd %Qr.m.c.2).

Adjustment Parameters

Parameter	Valid Values
Stop Frequency (in Hz)	0 Hz to 65,535 Hz, default is 0 Hz, limited by Max Frequency
Deceleration Rate	10 to 32,500, default is 100, limited by Max Deceleration
Emergency Deceleration Rate	10 to 32,500, default is 100, limited by Max Deceleration

Command Status Follow-Up

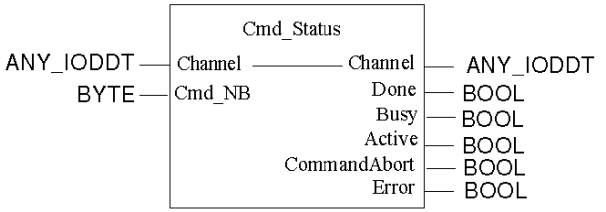
Description

There are two ways for the user to get the information about the status of a command:

- directly through the implicit objects %IWr.m.c.0 to %IWr.m.c.5.
- via the Cmd_Status DFB

Representation in FBD

Representation:

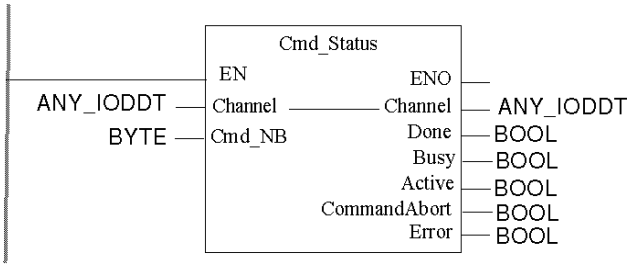


NOTE: The command status follow-up is the only PTO function which doesn't need to be enabled (via EN input) in FBD representation.

CAUTION
RISK OF UNINTENDED EQUIPMENT The direct link between the CMD_NB output of the command bloc function and the corresponding input of the CMB_Status bloc does not work. It's required to us an intermediate static byte value (CMB_NB) between the output of the motion bloc and the associeted input of the DFB CMB_status. **Failure to follow these instructions can result in equipment damage.**

Representation in LD

Representation:



Representation in IL

Representation:

```
CAL FBI_x (Channel := (*T_PTO_BMX*), Cmd_Nb := (*BYTE*), Done
=> (*BOOL*), Busy => (*BOOL*), Active => (*BOOL*),
CommandAborted => (*BOOL*), Error => (*BOOL*))
```

where x is a number.

Representation in ST

Representation:

```
FBI_x (Channel := (*T_PTO_BMX*), Cmd_Nb := (*BYTE*), Done =>
(*BOOL*), Busy => (*BOOL*), Active => (*BOOL*), CommandAborted
=> (*BOOL*),Error => (*BOOL*));
```

where x is a number.

Input/Output Description

Inputs description:

Name	Type	Description
Channel	T_PTO_BMX	The IODDT of the PTO channel to which the command has been sent. This pin is also repeated as an output of the block.
Cmd_Nb	BYTE	The number of the command. This object corresponds either to: ● The output of a PTO EF ● The CMD_SENT_NB (%MWr.m.c.13) object – converted to BYTE type - after use of the WRITE_CMD instruction.

Outputs description:

Name	Type	Description
Done	BOOL	The command has been executed and completed successfully
Busy	BOOL	The command has been accepted by the PTO channel but is not completed yet.
Active	BOOL	The command is being executed.
CommandAborted	BOOL	The command has been aborted before completion.
Error	BOOL	An error has been detected before the command completion.

The boolean outputs "Done", "Busy", "CommandAborted" and "Error" indicate the current status of the command. As required by the PLCopen standard, these outputs are mutually exclusive: only one will be set TRUE at a given time.

NOTE: If Cmd_Nb is different from 0, at least one of these outputs will be TRUE, except during one PLC cycle when all outputs will be FALSE, immediatly after the Cmb_Nb input value is modified.

For buffered commands:

- when the command is in buffer (not yet in execution), Busy is TRUE.
- when the command is being executed, Active is TRUE.

For non-buffered commands, the values for Active and Busy are TRUE when the command is being executed.

NOTE: The DFB outputs will remain unchanged as long as there is no change in the status of the specified command or up to the moment the command number is re-used by another command. If, after a periode of time a new command is sent that has the same command number, the outputs of the DFB will then change to reflect the status of this new command.

Adjustment

12

Overview

This chapter provides necessary information to adjust the BMX MSP 0200 module.

What's in this Chapter?

This chapter contains the following topics:

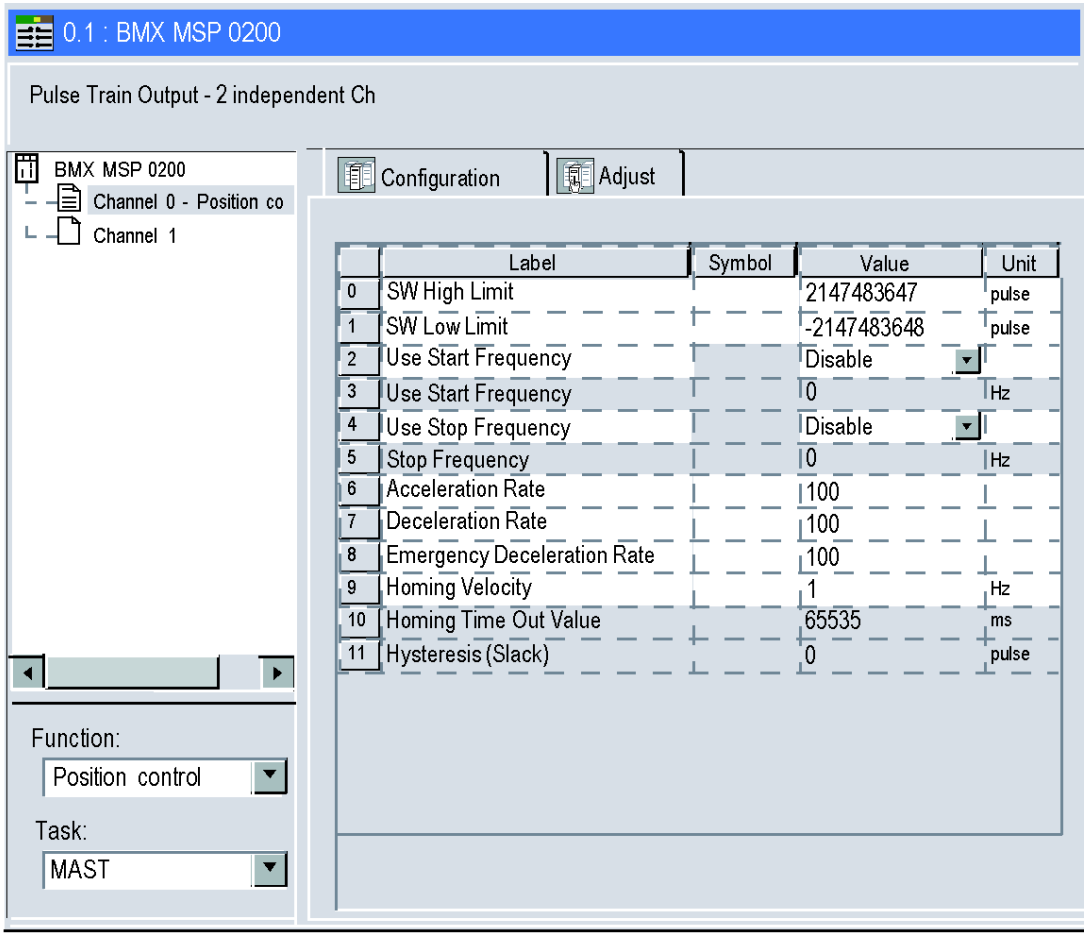
Adjust Screen for BMX MSP 0200 PTO module

At a Glance

This section presents the adjust screen for the BMX MSP 0200 PTO module.

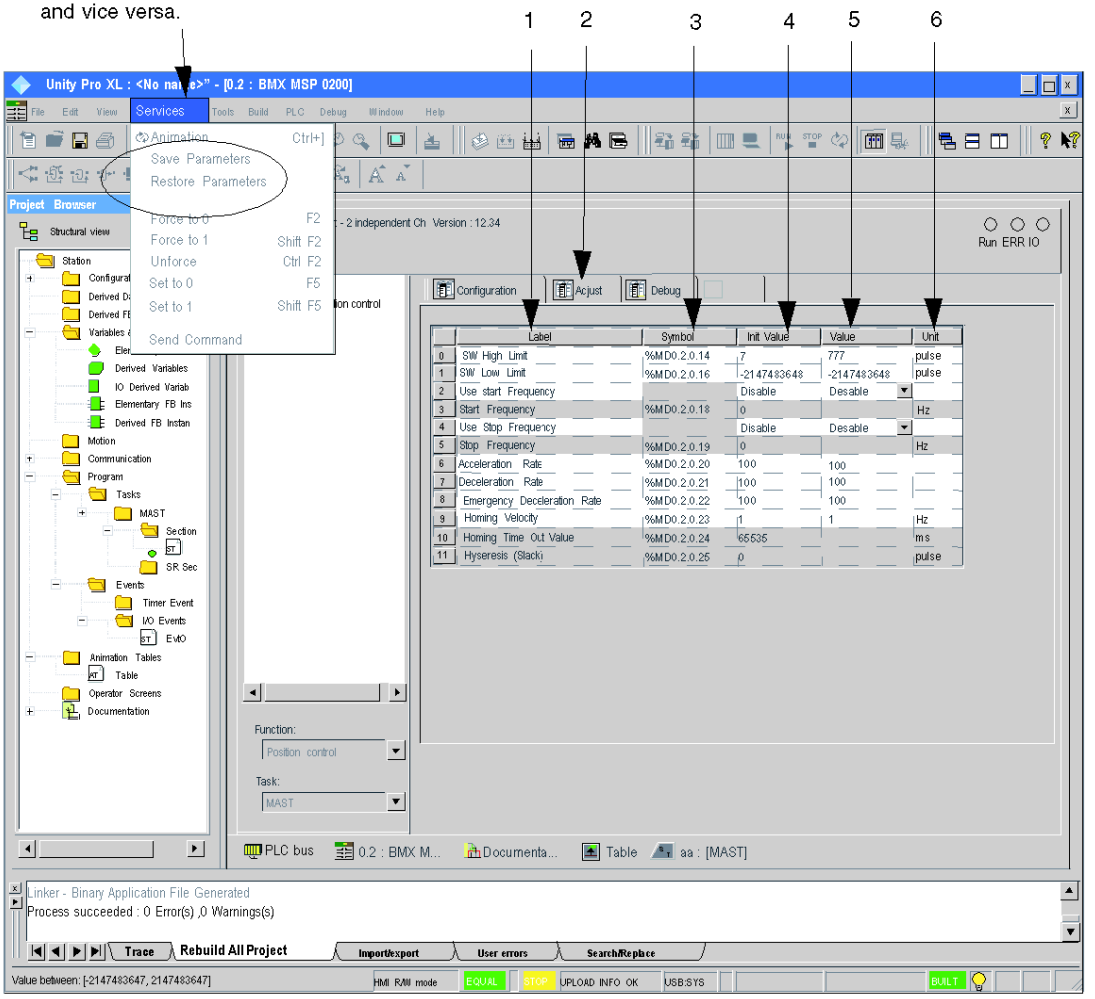
Illustration

The figure below presents the adjust screen offline for the BMX MSP 0200 PTO module in position control mode:



The figure below presents the adjust screen online for the BMX MSP 0200 PTO module in position control mode:

use save and restore parameter from the service menu to copy initial value to value field and vice versa.



Description of the Screen

The following table presents the various parts of the above screen:

Number	Element	Function
1	**Label** field	This field contains the name of each variable that may be adjusted. This field cannot be modified.
2	Tab	The tab in the foreground indicates the current mode. The current mode is therefore the adjust mode in this example.
3	**Symbol** field	This field contains the mnemonics of the variable. This field cannot be modified.
4	**Initial value** field	This field displays the value of the variable that has been adjusted in the "value" column in offline mode.
5	**Value** field	The function of this field depends on the mode in which the user is working: ● **In offline mode**: initial value of the variable can be adjusted. ● **In online mode**: the current value of the variable can be displayed and adjusted. Modifying a value requires a validation action.
6	**Unit** field	This field contains the unit of each variable that may be configured. This field cannot be modified.

Position Control Mode Adjustment

At a Glance

The adjustment values of a BMX MSP 0200 PTO module are stored in 2 areas:

- %MWadjust for current values,
- %KP for initial values.

The parameters r,m and c shown in the following tables represent the topological addressing of the module. Each parameter has the following signification:

- r: represents the rack number,
- m:represents the position of the module on the rack,
- c: represents the channel number.

Adjustment Objects

The table below presents the position control mode configurable elements.

Number	Address in the configuration	Configurable values
SW High Limit	%MDr.m.c.14	-2,147,483,647 to 2,147,483,647 (default value = 2,147,483,6437 or SW Max High Limit if lower)
SW Low Limit	%MDr.m.c.16	-2,147,483,648 to 2,147,483,646 (default value = 2,147,483,648 or SW Min Low Limit if higher)
Use Start Frequency	%MWr.m.c.18	• Disable (default) • Enable
Start Frequency	%MWr.m.c.18	1 to 65,535 (default 1)
Use Stop Frequency	%MWr.m.c.19	• Disable (default) • Enable
Stop Frequency	%MWr.m.c.19	1 to 65,535 (default 1)
Acceleration Rate	%MWr.m.c.20	10 to 32,500 (default value = 100 or Max Acceleration if lower)
Deceleration Rate	%MWr.m.c.21	10 to 32,500 (default value = 100 or Max Deceleration if lower)
Emergency Deceleration Rate	%MWr.m.c.22	10 to 32,500 (default value = 100 or Max Deceleration if lower)
Homing Velocity	%MWr.m.c.23	1 to 65,535 (default 1)
Homing Time Out Value	%MWr.m.c.24	1 to 65,535 (default 65,535)
Hysteresis (Slack)	%MWr.m.c.25	0 to 255 (default value = 0)

The values have value restrictions that needs to be respected. *(see page 123)*

Slack Correction

At a Glance

The adjustment parameter Hysteresis (Slack) is used to define the number of output pulses to ignore from the position after every change of direction.

Configuration Procedure

To apply a slack correction, it is necessary to follow this procedure in order to configure it properly:

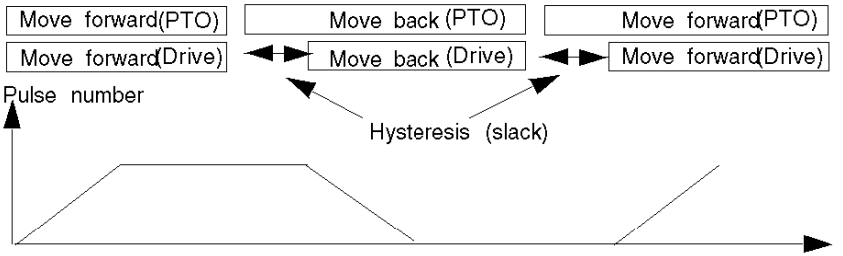
Step:	Action:
1	Set the Slack Correction value and validate the change. The Slack Correction will be activated if value is different than 0.
2	Before sending a command, it is necessary to reference the axis (SETPOSITION is not enought).
3	The system will automatically take in account the slack value for the following commands.

Illustration

When the configured pulse output mode is A/B phases (either normal or reverse), a hysteresis can be applied when changing direction.

The behavior will then be as follows:

Slack correction:



Diagnostic and debugging the BMX MSP 0200 PTO module

13

At a Glance

This chapter provides necessary information to diagnose and debug the BMX MSP 0200 module.

What's in this Chapter?

This chapter contains the following topics:

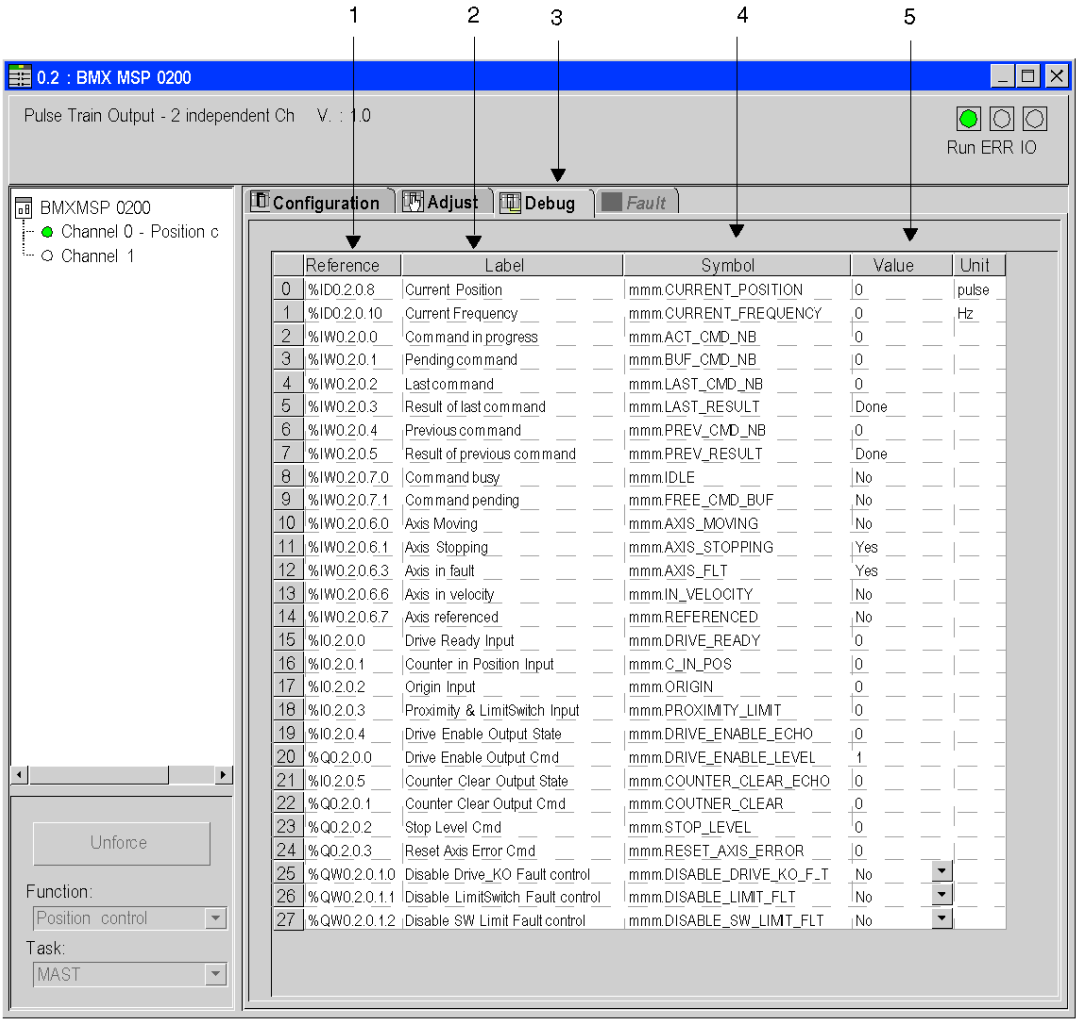
Debug Screen for BMX MSP 0200 PTO Module

At a Glance

This section presents the debug screen for BMX MSP 0200 PTO module. A module's debug screen can only be accessed in online mode.

Illustration

The screen presents the debug screen for the BMX MPS 0200 PTO module:



Description of the Screen

The following table presents the various parts of the above screen:

Number	Element	Function
1	**Reference** field	This field contains the address of the variable in the application. This field may not be modified.
2	**Label** field	This field contains the name of each variable that may be configured. This field may not be modified.
3	**Tab**	The tab in the foreground indicates the current mode. The current mode is therefore the debug mode in this example.
4	**Symbol** field	This field contains the mnemonics of the variable. This field may not be modified.
5	**Value** field	This field contains a drop-down menu containing all the possible values. If there is no downward pointing arrow, this field simply displays the current value of the variable.

Debugging Parameter Description

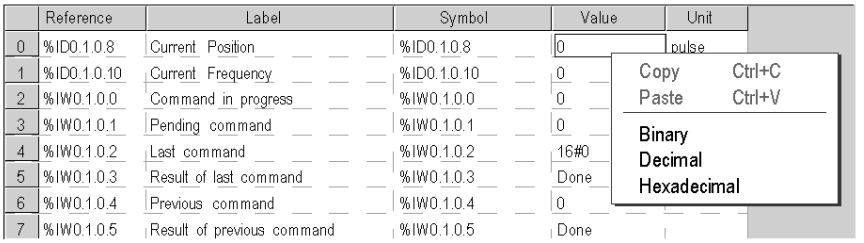
Overview

This is a description af the parameters found on the debugging screen on Unity Pro.

Possible Actions

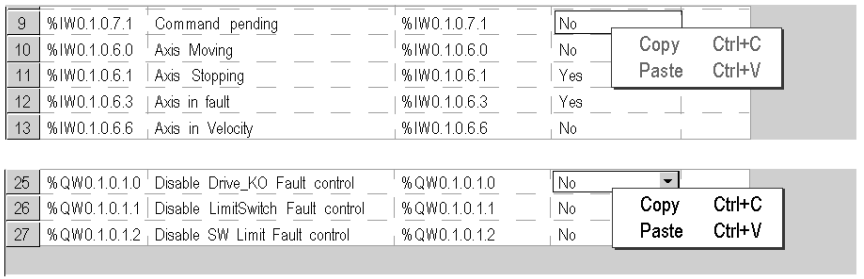
Different actions are possible with language interface objects

Numeric



	Reference	Label	Symbol	Value	Unit
0	%ID0.1.0.8	Current Position	%ID0.1.0.8	0	pulse
1	%ID0.1.0.10	Current Frequency	%ID0.1.0.10	0	
2	%IW0.1.0.0	Command in progress	%IW0.1.0.0	0	
3	%IW0.1.0.1	Pending command	%IW0.1.0.1	0	
4	%IW0.1.0.2	Last command	%IW0.1.0.2	16#0	
5	%IW0.1.0.3	Result of last command	%IW0.1.0.3	Done	
6	%IW0.1.0.4	Previous command	%IW0.1.0.4	0	
7	%IW0.1.0.5	Result of previous command	%IW0.1.0.5	Done	

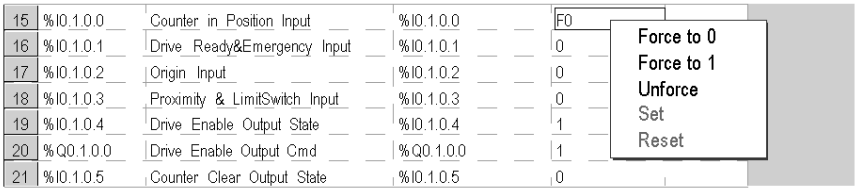
Binary



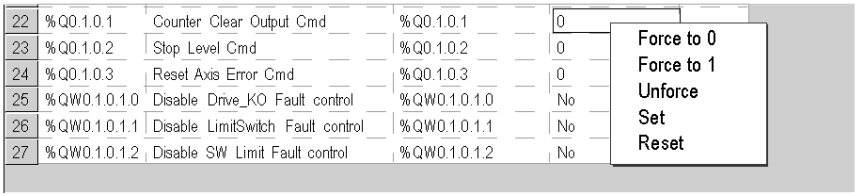
9	%IW0.1.0.7.1	Command pending	%IW0.1.0.7.1	No
10	%IW0.1.0.6.0	Axis Moving	%IW0.1.0.6.0	No
11	%IW0.1.0.6.1	Axis Stopping	%IW0.1.0.6.1	Yes
12	%IW0.1.0.6.3	Axis in fault	%IW0.1.0.6.3	Yes
13	%IW0.1.0.6.6	Axis in Velocity	%IW0.1.0.6.6	No

25	%QW0.1.0.1.0	Disable Drive_KO Fault control	%QW0.1.0.1.0	No
26	%QW0.1.0.1.1	Disable LimitSwitch Fault control	%QW0.1.0.1.1	No
27	%QW0.1.0.1.2	Disable SW Limit Fault control	%QW0.1.0.1.2	No

IOIM Binary



15	%I0.1.0.0	Counter in Position Input	%I0.1.0.0	F0
16	%I0.1.0.1	Drive Ready&Emergency Input	%I0.1.0.1	0
17	%I0.1.0.2	Origin Input	%I0.1.0.2	0
18	%I0.1.0.3	Proximity & LimitSwitch Input	%I0.1.0.3	0
19	%I0.1.0.4	Drive Enable Output State	%I0.1.0.4	1
20	%Q0.1.0.0	Drive Enable Output Cmd	%Q0.1.0.0	1
21	%I0.1.0.5	Counter Clear Output State	%I0.1.0.5	0



22	%Q0.1.0.1	Counter Clear Output Cmd	%Q0.1.0.1	0
23	%Q0.1.0.2	Stop Level Cmd	%Q0.1.0.2	0
24	%Q0.1.0.3	Reset Axis Error Cmd	%Q0.1.0.3	0
25	%QW0.1.0.1.0	Disable Drive_KO Fault control	%QW0.1.0.1.0	No
26	%QW0.1.0.1.1	Disable LimitSwitch Fault control	%QW0.1.0.1.1	No
27	%QW0.1.0.1.2	Disable SW Limit Fault control	%QW0.1.0.1.2	No

Value Table

This table describes all the debugging elements with their default value.

Label	Address in configuration	Type	Internal values	Default value
Current Position	%IDr.m.c.8	Num	Signed	0
Current Frequency	%IDr.m.c.10	Num	Signed	0
Command in progress	%IWr.m.c.0	Num	Unsigned	0
Pending command	%IWr.m.c.1	Num	Unsigned	0
Last command	%IWr.m.c.2	Num	Unsigned	0
Result of last command	%IWr.m.c.3	List	• Done • Error • Aborted • N/A	N/A
Previous command	%IWr.m.c.4	Num		0
Result of previous command	%IWr.m.c.5	List	• Done • Error • Aborted • N/A	N/A
Command busy	%IWr.m.c.7.0	Binary	Yes(0)/No(1)	No
Command pending	%IWr.m.c.7.1	Binary	Yes(0)/No(1)	No
Axis Moving	%IWr.m.c.6.0	Binary	Yes(1)/No(0)	No
Axis Stopping	%IWr.m.c.6.1	Binary	Yes(1)/No(0)	No
Axis in fault	%IWr.m.c.6.3	Binary	Yes(1)/No(0)	No
Axis in Velocity	%IWr.m.c.6.6	Binary	Yes(1)/No(0)	No
Axis referenced	%IWr.m.c.6.7	Binary	Yes(1)/No(0)	No
Drive Ready&Emergency Input	%Ir.m.c.0	Binary	0/1	0
Counter in Position Input	%Ir.m.c.1	Binary	0/1	0
Origin Input	%Ir.m.c.2	Binary	0/1	0
Proximity & LimitSwitch Input	%Ir.m.c.3	Binary	0/1	0
Drive Enable Output State	%Ir.m.c.4	Binary	0/1	0
Drive Enable Output Cmd	%Qr.m.c.0	Binary	0/1	0
Counter Clear Output State	%Ir.m.c.5	Binary	0/1	0
Counter Clear Output Cmd	%Qr.m.c.1	Binary	0/1	0
Stop Level Cmd	%Qr.m.c.2	Binary	0/1	0
Reset Axis Error Cmd	%Qr.m.c.3	Binary	0/1	0
Disable Drive KO Fault	%QWr.m.c.1.0	Binary	Yes(1)/No(0)	No

Label	Address in configuration	Type	Internal values	Default value
Disable LimitSwitch Fault	%QWr.m.c.1.1	Binary	Yes(1)/No(0)	No
Disable SW Limit Fault	%QWr.m.c.1.2	Binary	Yes(1)/No(0)	No

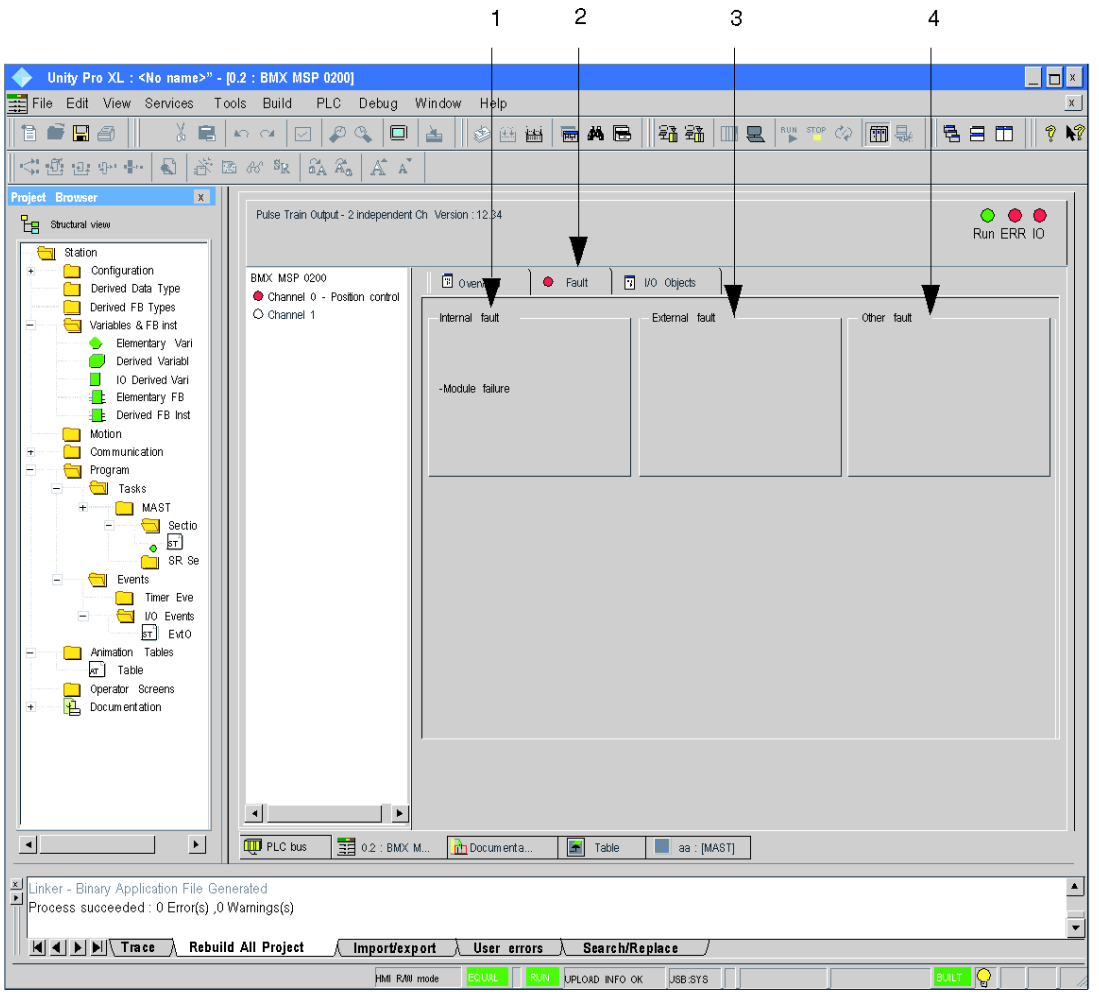
Diagnostic Screen for the BMX MSP 0200 PTO module

At a Glance

This section presents the diagnostic screen for the BMX MSP 0200 PTO module. A module's diagnostic screen may only be accessed in online mode unlike other modules for M340, the PTO module diagnostic screen is accessible even if CH_ERROR = 0.

Illustration

The figure below presents the Diagnostic Screen for the BMX MSP 0200 PTO module in position control mode.


1
2
3
4
Unity Pro XL : <No name>" - [0.2 : BMX MSP 0200]
File Edit View Services Tools Build PLC Debug Window Help
Project Browser
Structural view
Station
Configuration
Derived Data Type
Derived FB Types
Variables & FB inst
Elementary Vari
Derived Variabl
IO Derived Vari
Elementary FB
Derived FB Inst
Motion
Communication
Program
Tasks
MAST
Sectio
SR Se
Events
Timer Eve
I/O Events
EvtIO
Animation Tables
Table
Operator Screens
Documentation
Pulse Train Output - 2 independent Ch Version : 12.34
Run ERR IO
BMX MSP 0200
Channel 0 - Position control
Channel 1
Overv
Fault
I/O Objects
Internal fault
-Module failure
External fault
Other fault
PLC bus
0.2 : BMX M...
Documenta...
Table
aa : [MAST]
Linker - Binary Application File Generated
Process succeeded : 0 Error(s) ,0 Warnings(s)
Trace
Rebuild All Project
Import/export
User errors
Search/Replace
HMI R/W mode
EQUAL
RUN
UPLOAD INFO OK
USB:SYS
BUILT

Description of the Screen

The following table presents the various parts of the diagnostic screen.

Number	Element	Function
1	Internal faults field	This field displays the module's active internal detected errors.
2	Tab	The tab in the foreground indicates the current mode. The current mode is therefore the detected errors display mode in this example.
3	External faults field	This field displays the module's active external errors.
4	Other faults field	This field displays the module's active detected errors, other than internal and external detected errors.

Diagnostic Parameters Description

BMX MSP 0200 Diagnostics

This table describes the list of errors the diagnostic screen will display.

Object	Type	Symbol	Detail
%MWr.m.c.2		**CH_FLT**	**Standard channel detected errors**
x0	External	EXT_FLT_PWS	External power supply fault
x1	External	EXT_FLT_OUTPUTS	External fault on outputs (short-circuit, overload)
x2	Unused		
x3	Unused		
x4	Internal	INTERNAL_FLT	Inoperative channel or Module missing
x5	Other	CONF_FLT	Hardware or software configuration fault.
x6	Other	COM_FLT	Error communication with PLC
x7	Other	APPLI_FLT	Application error
%MWr.m.c.3		**CMD_FLT**	**Command Faults**
x0	Other	OVERRUN_CMD	Overrun condition while sending command
x1	Other	AXIS_IN_FLT	Invalid command due to axis in ErrorStop state
x2	Other	CMD_CODE_INV	Invalid command code
x3	Other	CMD_SEQ_INV	Invalid sequence of commands
x4	Other	BUFFER_FULL	Command rejected due to buffer full (Idle=FreeCmdBuf=0)
x5	Other	AXIS_NOT_REFERENCED	Positioning command rejected due to axis not referenced
x6	Other	TGT_POS_INV	Invalid target position
x7	Other	TGT_VEL_INV	Invalid target velocity
x8	Other	BUFFER_MODE_INV	Invalid buffer mode
%MWr.m.c.4		**ADJUST_FLT**	**Adjustment Parameter Faults**
x0	Other	OVERRUN_ADJUST	Overrun condition during adjustment instruction
x1	Other	SW_HIGH_LIMIT_INV	Invalid SW high limit
x2	Other	SW_LOW_LIMIT_INV	Invalid SW low limit
x3	Other	ACC_RATE_INV	Invalid acceleration rate
x4	Other	DEC_RATE_INV	Invalid deceleration rate
x5	Other	EMER_DEC_RATE_INV	Invalid emergency deceleration rate
x6	Other	START_FREQ_INV	Invalid start frequency
x7	Other	STOP_FREQ_INV	Invalid stop frequency
x8	Other	HOMING_VELO_INV	Invalid homing frequency

Object	Type	Symbol	Detail
%MWr.m.c.5		**AXIS_ERROR**	**Axis Errors**
x0	External	DRIVE_KO	Drive_Ready&Emergency input is off
x1	External	LIMIT_FLT	Limit have been exceeded (LimitSwitch input)
x2	External	SW_HIGH_LIMIT_FLT	High software limit reached
x3	External	SW_LOW_LIMIT_FLT	Low software limit reached
x4	External	HOMING_FLT	Error during homing
x5	Unused		
x6	Unused		
x7	Unused		

Management of Detected Errors

Overview

Four kinds of detected errors can be encountered by the BMX MSP 0200 module and reported in the status objects (%MWr.m.c.2 to %MWr.m.c.5): Standard errors, Command errors, Adjustment parameter errors, Axis errors.

Standard Channel Faults

These are reported through %MWr.m.c.2 object (Standard Channel Error) and induce a channel error, reported in %Ir.m.c.ERR.

Detected errors described by bits 4 to 7 (internal, configuration, communication and application errors) have the same meaning as for all other modules from M340 range.

External Power Supply Fault (%MWr.m.c.2.0) reports a supply error if this report is enabled by configuration (i.e. if Power Supply Fault - %KWr.m.c.1.8 – is set to General I/O Fault).

⚠ CAUTION

IRREVERSIBLE DAMAGE TO PTO MODULE

Do not reverse the connection of the external power supply.

Follow the wiring *(see page 31)*, mounting and installation *(see page 19)* instructions.

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

If enabled by configuration (i.e. if Output Fault - %KWr.m.c.1.9 – is set to General I/O Fault), external detected error on outputs (%MWr.m.c.2.1) are reported for: *(see page 44)*

- a short-circuit,
- an overload,
- loss of power supply if Power Supply Fault is localy configured

Detected Command Errors

These occur when a command is rejected by the module or when sending the command is unsuccessful.

Detected errors are reported into %MWr.m.c.1.1 CMD_ERR object.

A detected command error generates the following behavior:

- The axis is put in error stop state (reported through AXIS_STS – %IWr.m.c.6 – object with bits 1 (STOPPING) and 3 (AXIS_FLT) set to 1).
- The detail of the detected error is described in %MWr.m.c.3 (Command Fault object).
- Any command in progress or in buffer will be aborted in error.
- If a Frequency Generator profile was currently being output, the axis will be stopped immediately. Otherwise, the axis will be stopped smoothly using the emergency deceleration rate.

No other commands are accepted before the axis is stopped and the detected axis error is reset (through Reset_Axis_Error – %Qr.m.c.3 – object).

⚠ WARNING

UNCONTROLLED RESTART

If Reset_Axis_Error (%Qr.m.c.3) is set to 1, the module will accept commands from the application again, which can generate a motion.

Install audible and visual alarm on your application.

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

Detected Adjustment Parameter Errors

These occur when adjustment parameters are rejected or when sending the parameters is unsuccessful. *(see page 124)*

Detected errors are reported into %MWr.m.c.1.2 ADJUST_ERR object.

A detected adjustment parameter error does not put the axis in ErrorStop state, and does not have an impact on the channel behavior.

The channel will continue running with previous parameters as though no parameters had been sent.

Detected Axis Errors

There are 4 different kinds of detected axis errors.

Drive_KO or Emergency

If monitoring is enabled (Implicit object %QWr.m.c.1.0 (Disable Axis Faults / Drive_Ready&Emergency) is set to 0), and if Drive_Enable physical output has been active for more than 100ms, this error will be detected as soon as the Drive_Ready&Emergency physical input falls to low state.

This detected error induces the following behavior:

- The axis is put in error stop state (reported through AXIS_STS – %IWr.m.c.6 – object with bits 1 (STOPPING) and 3 (AXIS_FLT) set to 1).
- The detail of the detected error is described in %MWr.m.c.5 Axis Errors object (bit 0: DRIVE_KO).
- The axis is unreferenced (%IWr.m.c.6.7 reset to 0).
- Any command in progress or in buffer will be aborted in error and no further command can be sent.
- If any profile was currently being output, the axis will be stopped immediately.

There is no deceleration phase using emergency deceleration rate here. Such a condition is a mechanical axis or an external emergency, both of which require an immediate stop of the mechanical axis.

When the condition is corrected (or monitoring is disabled), reset the detected axis error (through Reset_Axis_Error – %Qr.m.c.3 – object) in order to send a new command.

⚠ WARNING

UNCONTROLLED RESTART

If Reset_Axis_Error (%Qr.m.c.3) is set to 1, the module will accept commands from the application again, which can generate a motion.

Install audible and visual alarm on your application.

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

Limit crossing

If monitoring is enabled (Implicit object %QWr.m.c.1.1 (DISABLE_LIMIT_FLT) is set to 0), this error is detected when Proximity&LimitSwitch physical input rises

This detected error induces the following behavior:

- The axis is put in error stop state (reported through AXIS_STS – %IWr.m.c.6 – object with bits 1 (STOPPING) and 3 (AXIS_FLT) set to 1).
- The detail of the detected error is described in %MWr.m.c.5 Axis Errors object (bit 1: LIMIT_FLT).
- No impact on the value of %IWr.m.c.6.7 (Axis referenced)
- Any command in progress or in buffer will be aborted in error.
- If a Frequency Generator profile was currently being output, the axis will be stopped immediately. Otherwise, the axis will be stopped smoothly using the emergency deceleration rate.

Only the following commands can be accepted :

- Frequency Generator or Move Velocity commands in the opposite direction of the previous command. As soon as the axis is back in the valid area the Proximity&LimitSwith input is set to low and the axis must be stopped. The detected axis error remains (STOPPING and AXIS_FLT bits of AXIS_STS object and LIMIT_FLT bit of AXIS_ERROR object remain set to 1).
- Short Cam with Positive Limit and Short Cam with Negative Limit, when these commands are used, the detected error will be cleared.

The detected axis error needs to be reset (through %Qr.m.c.3 object) before being able to send other new commands.

⚠ WARNING

UNCONTROLLED RESTART

If Reset_Axis_Error (%Qr.m.c.3) is set to 1, the module will accept commands from the application again, which can generate a motion.

Install audible and visual alarm on your application.

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

Important: As both PTO channel and drive have a limit switch input, it is not recommended to use the same cabling for both of them. Otherwise, an out-of-limit condition on the drive would induce a DRIVE_KO detected error on the PTO channel simultaneously with the Limit Fault. It would not be possible then to have the same behaviour as described previously for Limit Crossing (velocity/homing commands would be rejected).

SW limit reached

If monitoring enabled (Implicit object %QWr.m.c.1.2 (DISABLE_SW_LIMIT_FLT) is set to 0) this internally managed detected error occurs when the current position as seen by the channel (%IDr.m.c.8) reaches one of the two SW limit values.

This detected error induces the following behavior:

- The axis is put in error stop state (reported through AXIS_STS – %IWr.m.c.6 – object with bits 1 (STOPPING) and 3 (AXIS_FLT) set to 1).
- The detail of the detected error is described in %MWr.m.c.5 Axis Errors object (bit 2: SW_HIGH_LIMIT_FLT or bit 3: SW_LOW_LIMIT_FLT).
- No impact on the value of %IWr.m.c.6.7 (Axis referenced)
- Any command in progress or in buffer will be aborted in error.
- If a Frequency Generator profile was currently being output, the axis will be stopped immediately. Otherwise the axis will be stopped smoothly using the emergency deceleration rate.

In this state, the following commands are accepted: Frequency Generator or Move Velocity in the opposite direction of the previous command (in order for the axis to return to the valid area) are accepted.

As soon as the axis is back and stopped in the valid range of position values the SW limit error disappears, but the axis error remains (STOPPING and AXIS_FLT bits of AXIS_STS object and SW_HIGH/LOW_LIMIT_FLT bit of AXIS_ERROR object stay high).

The detected axis error needs to be reset (through %Qr.m.c.3 object) before being able to send other new commands.

⚠ WARNING

UNCONTROLLED RESTART

If Reset_Axis_Error (%Qr.m.c.3) is set to 1, the module will accept commands from the application again, which can generate a motion.

Install audible and visual alarm on your application.

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

Overflow of the position value

This detected error is a specific case of SW limit error and happens when the position value goes beyond the minimum or maximum possible pulse number (-2,147,483,648 or 2,147,483,647).

This will cause a change of sign of the position, whose value is no more significant.

If SW Limit monitoring is enabled, an error will be detected and the following behavior will be induced:

- The axis is put in error stop state (reported through AXIS_STS – %IWr.m.c.6 – object with bits 1 (STOPPING) and 3 (AXIS_FLT) set to 1).
- The detail of the detected error is described in %MWr.m.c.5 Axis Errors object (bit 2: SW_HIGH_LIMIT_FLT or bit 3: SW_LOW_LIMIT_FLT).
- The axis is unreferenced (%IWr.m.c.6.7 is reset to 0).
- Any command in progress or in buffer will be aborted in error.
- If a Frequency Generator profile was currently being output, the axis will be stopped immediately. Otherwise the axis will be stopped smoothly using the emergency deceleration rate

The axis error needs to be reseted (through %Qr.m.c.3 object) before being able to send other new commands but the axis remains unreferenced

⚠ WARNING

UNCONTROLLED RESTART

If Reset_Axis_Error (%Qr.m.c.3) is set to 1, the module will accept commands from the application again, which can generate a motion.

Install audible and visual alarm on your application.

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

NOTE: If the axis is referenced and the SW limit monitoring is disabled, if the maximum or minimum position value is reached in continuous command, no specific processing will occur. The position will change sign and continue evolving.

Homing faults

These occur during the execution of a homing command.

There are two possible cases:

- Homing Time-out detected error: when Counter_in_Position input is used (set by configuration), a detected homing function error is reported if Counter_in_Position remains low after a certain duration (time out value to be configured in setting parameters).
- Homing-mode specific detected errors: unauthorized start from cam, wrong direction.
  For the details of these conditions, please check the description of each homing mode *(see page 185)*

This detected error induces the following behavior:

- The axis is put in error stop state (reported through AXIS_STS – %IWr.m.c.6 – object with bits 1 (STOPPING) and 3 (AXIS_FLT) set to 1).
- The detail of the detected error is described in %MWr.m.c.5 Axis Errors object (bit 4: HOMING_FLT).
- The current homing command is aborted in error.
- The axis is unreferenced (%IWr.m.c.6.7 set to 0).

The detected axis error needs to be reset (through %Qr.m.c.3 object) before being able to send other new commands.

⚠ WARNING

UNCONTROLLED RESTART

If Reset_Axis_Error (%Qr.m.c.3) is set to 1, the module will accept commands from the application again, which can generate a motion.

Install audible and visual alarm on your application.

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

The Language Objects of the PTO Function

14

Subject of this Chapter

This chapter describes the language objects associated to the BMX MSP 0200 module tasks as well as the different ways of using them.

What's in this Chapter?

This chapter contains the following topics:

Introducing Language Objects for Application-Specific PTO

General

The BMX MSP 0200 PTO module has only one associated IODDT. It is predefined and contains language objects for inputs/outputs belonging to the channel of an application-specific module.

The IODDT associated with the module is T_PTO_BMX.

NOTE: IODDT variables can be created in two different ways:

- Using the **I/O objects** tab. *(see Unity Pro, Operating Modes,)*
- Using the Data Editor *(see Unity Pro, Operating Modes,)*.

Language Object Types

The IODDT contains a set of language objects allowing its operation to be controlled and checked.

There are two types of language objects:

- **Implicit Exchange Objects:** these objects are automatically exchanged on each cycle revolution of the task associated with the module.
- **Explicit Exchange Objects:** these objects are exchanged on the application's request, using explicit exchange instructions.

Implicit exchanges concern the inputs/outputs of the module (measurement results, information and commands). These exchanges enable the debugging of the module.

Explicit exchanges enable the module to be set, diagnosed or order the output a specific profile.

Position Control IODDT Object

At a glance

This section globaly presents the position control IODDT languages and objects.

T_PTO_BMX

Input/output table linked to T_PTO_BMX IODDT object

	Symbol	Address	Type	Description
IMP	**CH_ERROR**	**%I.r.m.c.ERR**	**BOOL**	**Channel error**
IMP	DRIVE_READY_EMERGENCY	%Ir.m.c.0	EBOOL	State of Physical input Drive_Ready_Emergency
IMP	C_IN_POS	%Ir.m.c.1	EBOOL	Counter in Position
IMP	ORIGIN	%Ir.m.c.2	EBOOL	Origin Physical Input State
IMP	PROXIMITY_LIMIT	%Ir.m.c.3	EBOOL	Proximity&LimitSwitch Physical Input State
IMP	DRIVE_ENABLE_ECHO	%Ir.m.c.4	EBOOL	State of Drive Enable Level output
IMP	COUNTER_CLEAR_ECHO	%Ir.m.c.5	EBOOL	State of Counter Clear output
IMP	ACT_CMD_NB	%IWr.m.c.0	INT	Number of the command in progress
IMP	BUF_CMD_NB	%IWr.m.c.1	INT	Number of the command in buffer
IMP	LAST_CMD_NB	%IWr.m.c.2	INT	Number of last command executed
IMP	LAST_RESULT	%IWr.m.c.3	INT	Status of last command executed
IMP	PREV_CMD_NB	%IWr.m.c.4	INT	History: Number of the command executed previously
IMP	PREV_RESULT	%IWr.m.c.5	INT	History: Status of the command executed previously
IMP	**AXIS_STS**	**%IWr.m.c.6**	**INT**	**Axis Status**
IMP	AXIS_MOVING	%IWr.m.c.6.0	BOOL	The axis is moving
IMP	AXIS_STOPPING	%IWr.m.c.6.1	BOOL	The axis is stopping
IMP	AXIS_FLT	%IWr.m.c.6.3	BOOL	Axis in ErrorStop state
IMP	IN_VELOCITY	%IWr.m.c.6.6		This axis is running at target frequency (for continuous profiles)
IMP	REFERENCED	%IWr.m.c.6.7	BOOL	The axis is referenced
IMP	**CMD_MGT**	**%IWr.m.c.7**	**INT**	**Command Management**
IMP	IDLE	%IWr.m.c.7.0	BOOL	No command is being executed
IMP	FREE_CMD_BUF	%IWr.m.c.7.1	BOOL	No command is pending
IMP	CURRENT_POSITION	%IDr.m.c.8	DINT	Current Position (in Pulses)
IMP	CURRENT_FREQUENCY	%IDr.m.c.10	DINT	Current Frequency (in Hz)

	Symbol	Address	Type	Description
IMP	DRIVE_ENABLE_LEVEL	%Qr.m.c.0	EBOOL	Force Drive Enable Level output to Highstate
IMP	COUNTER_CLEAR	%Qr.m.c.1	EBOOL	Force Counter Clear output to Highstate
IMP	STOP_LEVEL	%Qr.m.c.2	EBOOL	Stop the axis
IMP	RESET_AXIS_ERROR	%Qr.m.c.3	EBOOL	Reset axis error
IMP	**EVT_SOURCES_ENABLING**	**%QWr.m.c.0**	**INT**	**Field of Enable Event bits**
IMP	EVT_POSITION_REACHED	%QWr.m.c.0.0	BOOL	Call event when target position is reached
IMP	EVT_REFERENCING_DONE	%QWr.m.c.0.1	BOOL	Call event when axis referencing is done
IMP	AXIS_FAULT_DISABLING	%QWr.m.c.1	INT	Disable Axis Fault Detection bits
IMP	DISABLE_DRIVE_KO_FLT	%QWr.m.c.1.0	BOOL	Disable default report when Drive_Ready input is low
IMP	DISABLE_LIMIT_FLT	%QWr.m.c.1.1	BOOL	Disable default report when a limit is crossed
IMP	DISABLE_SW_LIMIT_FLT	%QWr.m.c.1.2	BOOL	Disable default report when SW limits are reached
SYS	**EXCH_STS**	**%MWr.m.c.0**	**INT**	**Exchange status**
SYS	STS_IN_PROGR	%MWr.m.c.0.0	BOOL	Status parameter read in progress
SYS	CMD_IN_PROGR	%MWr.m.c.0.1	BOOL	Command parameter write in progress
SYS	ADJ_IN_PROGR	%MWr.m.c.0.2	BOOL	Adjust parameter exchange in progress
SYS	RECONF_IN_PROGR	%MWr.m.c.0.15	BOOL	Reconfiguration in progress
SYS	**EXCH_RPT**	**%MWr.m.c.1**	**INT**	**Channel report**
SYS	STS_ERR	%MWr.m.c.1.0	BOOL	Error while reading channel status
SYS	CMD_ERR	%MWr.m.c.1.1	BOOL	Error while sending a command on the channel
SYS	ADJ_ERR	%MWr.m.c.1.2	BOOL	Error while adjusting the channel
SYS	RECONF_ERR	%MWr.m.c.1.15	BOOL	Error while reconfiguring the channel
STS	**CH_FLT**	**%MWr.m.c.2**	**INT**	**Channel faults**
STS	EXT_FLT_PWS	%MWr.m.c.2.0	BOOL	External Power Supply Fault
STS	EXT_FLT_OUTPUTS	%MWr.m.c.2.1	BOOL	External fault on the outputs
STS	INTERNAL_FLT	%MWr.m.c.2.4	BOOL	Internal fault: Channel inoperative
STS	CONF_FLT	%MWr.m.c.2.5	BOOL	Hardware or software configuration status
STS	COM_FLT	%MWr.m.c.2.6	BOOL	Bus Communication fault
STS	APPLI_FLT	%MWr.m.c.2.7	BOOL	Application fault
STS	**CMD_FLT**	**%MWr.m.c.3**	**INT**	**Command Faults**
STS	OVERRUN_CMD	%MWr.m.c.3.0	BOOL	Overrun condition while sending command

	Symbol	Address	Type	Description
STS	AXIS_IN_FLT	%MWr.m.c.3.1	BOOL	Invalid command due to axis in ErrorStop state
STS	CMD_CODE_INV	%MWr.m.c.3.2	BOOL	Invalid command code
STS	CMD_SEQ_INV	%MWr.m.c.3.3	BOOL	Invalid sequence of commands
STS	BUFFER_FULL	%MWr.m.c.3.4	BOOL	Command rejected due to buffer full (Idle=FreeCmdBuf=0)
STS	AXIS_NOT_REFERENCED	%MWr.m.c.3.5	BOOL	Positioning command rejected due to axis not referenced
STS	TGT_POS_INV	%MWr.m.c.3.6	BOOL	Invalid target position
STS	TGT_VEL_INV	%MWr.m.c.3.7	BOOL	Invalid target velocity
STS	BUFFER_MODE_INV	%MWr.m.c.3.8	BOOL	Invalid buffer mode
STS	**ADJUST_FLT**	**%MWr.m.c.4**	**INT**	**Adjustment Parameter Faults**
STS	OVERRUN_ADJUST	%MWr.m.c.4.0	BOOL	Overrun fault during adjustment instruction
STS	SW_HIGH_LIMIT_INV	%MWr.m.c.4.1	BOOL	Invalid SW high limit
STS	SW_LOW_LIMIT_INV	%MWr.m.c.4.2	BOOL	Invalid SW low limit
STS	ACC_RATE_INV	%MWr.m.c.4.3	BOOL	Invalid acceleration rate
STS	DEC_RATE_INV	%MWr.m.c.4.4	BOOL	Invalid deceleration rate
STS	EMER_DEC_RATE_INV	%MWr.m.c.4.5	BOOL	Invalid emergency deceleration rate
STS	START_FREQ_INV	%MWr.m.c.4.6	BOOL	Invalid start frequency
STS	STOP_FREQ_INV	%MWr.m.c.4.7	BOOL	Invalid stop frequency
STS	HOMING_VELO_INV	%MWr.m.c.4.8	BOOL	Invalid homing velocity
STS	**AXIS_ERROR**	**%MWr.m.c.5**	**INT**	**Axis Errors**
STS	DRIVE_KO	%MWr.m.c.5.0	BOOL	Drive Ready input is off
STS	LIMIT_FLT	%MWr.m.c.5.1	BOOL	Limit crossing has been detected
STS	SW_HIGH_LIMIT_FLT	%MWr.m.c.5.2	BOOL	Software high limit has been reached
STS	SW_LOW_LIMIT_FLT	%MWr.m.c.5.3	BOOL	Software low limit has been reached
STS	HOMING_FLT	%MWr.m.c.5.4	BOOL	Error during homing
CMD	CMD_CODE	%MWr.m.c.6	INT	Command Code
CMD	BUFFER_MODE	%MWr.m.c.7	INT	Buffer Mode for Positioning Commands
CMD	TGT_POSITION	%MDr.m.c.8	DINT	Target/Reference Position
CMD	TGT_VELOCITY	%MDr.m.c.10	DINT	Target Velocity
CMD	CMD_SENT_NB	%MWr.m.c.13	INT	Number of last command sent (Read only)
PRM	SW_HIGH_LIMIT	%MDr.m.c.14	DINT	Software High Limit
PRM	SW_LOW_LIMIT	%MDr.m.c.16	DINT	Software Low Limit
PRM	START_FREQ	%MWr.m.c.18	UINT	Start Frequency

	Symbol	Address	Type	Description
PRM	STOP_FREQ	%MWr.m.c.19	UINT	Stop Frequency
PRM	ACC_RATE	%MWr.m.c.20	UINT	Acceleration Rate
PRM	DEC_RATE	%MWr.m.c.21	UINT	Deceleration Rate
PRM	EMERGENCY_DEC_RATE	%MWr.m.c.22	UINT	Emergency Deceleration Rate
PRM	HOMING_VELOCITY	%MWr.m.c.23	UINT	Homing Velocity
PRM	HOMING_TIMEOUT_VALUE	%MWr.m.c.24	UINT	Homing Time Out Value
PRM	HYSTERESIS	%MWr.m.c.25	UINT	Hysteresis Value for A/B phases output mode

Explicit Exchange Language Objects Associated with the Application-Specific Function

Introduction

Explicit exchanges are performed when requested using these instructions:

- READ_STS *(see Unity Pro, I/O Management, Block Library)* (read status words)
- WRITE_CMD *(see Unity Pro, I/O Management, Block Library)* (write command words)
- WRITE_PARAM *(see Unity Pro, I/O Management, Block Library)* (write adjustment parameters)
- READ_PARAM *(see Unity Pro, I/O Management, Block Library)* (read adjustment parameters)
- SAVE_PARAM *(see Unity Pro, I/O Management, Block Library)* (save adjustment parameters)
- RESTORE_PARAM *(see Unity Pro, I/O Management, Block Library)* (restore adjustment parameters)

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

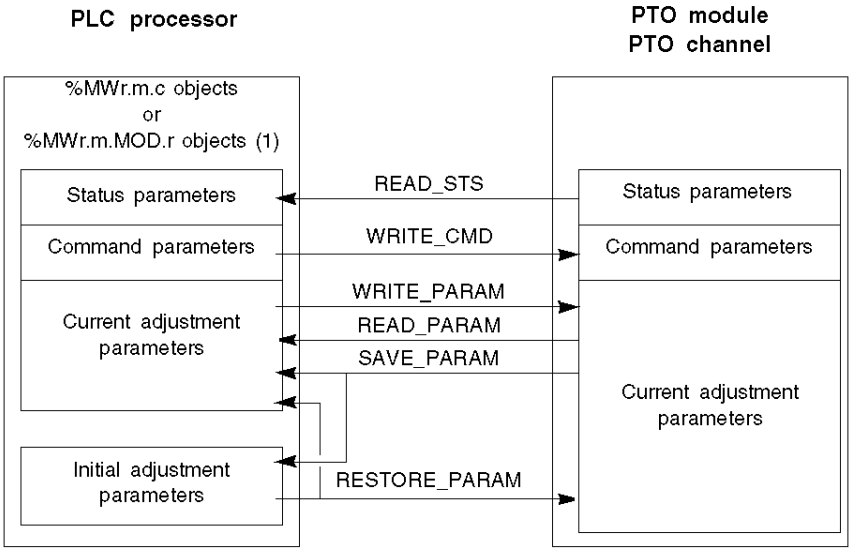
NOTE:

These objects can:

- provide information about the module (for example, type of channel fault)
- have command control of the module (for example, switch command)
- define the module's operating modes (save and restore adjustment parameters in the process of application)

General Principle for Using Explicit Instructions

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



(1) Only with READ_STS and WRITE_CMD instructions.

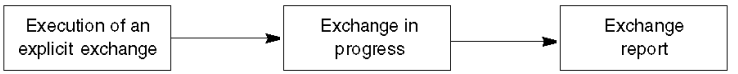
Managing Exchanges

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

To do this, two types of information are available:

- information concerning the exchange in progress *(see Unity Pro, I/O Management, Block Library)*
- the exchange report *(see Unity Pro, I/O Management, Block Library)*

The following diagram describes the management principle for an exchange.



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

Explicit System Objects %MWSys

Explicit System Objects %MWSys

Explicit System Objects %MWSys

Object	Type	Symbol	Detail
%MWr.m.c.0	INT	EXCH_STS	Implicit exchange execution indicators
x0	bit	STS_IN_PROGR	= 1 exchange in progress for READ_STS
x1	bit	CMD_IN_PROGR	= 1 exchange in progress for WRITE_CMD and PTO EFs
x2	bit	ADJUST_IN_PROGR	= 1 exchange in progress for adjustment parameters (via WRITE_PARAM, READ_PARAM, SAVE_PARAM, RESTORE_PARAM)
x15	bit	RECONF_IN_PROGR	= 1 indicates a reconfiguration on channel c of the module from the console (modification of the configuration parameters + cold start-up of the channel)
%MWr.m.c.1	INT	EXCH_RPT	Exchange report INT, updating at the end of exchange, 0 = correct exchange, 1 = incorrect exchange
x0	bit	STS_ERR	= 1 Fault when reading channel status INTs
x1	bit	CMD_ERR	= 1 Fault when exchanging WRITE_CMD or PTO EFs
x2	bit	ADJUST_ERR	= 1 Fault when exchanging adjustment parameters
x15	bit	RECONF_ERR	= 1 Fault when reconfiguring the channel

Explicit Status Parameters %MWStat

Object	Type	Symbol	Detail
%MWr.m.c.2		**CH_FLT**	**Standard channel errors**
x0	External	EXT_FLT_PWS	External power supply fault
x1	External	EXT_FLT_OUTPUTS	External fault on outputs (short-circuit, overload)
x2	Unused		
x3	Unused		
x4	Internal	INTERNAL_FLT	Inoperative channel or Module missing
x5	Other	CONF_FLT	Hardware or software configuration fault.
x6	Other	COM_FLT	Error communication with PLC
x7	Other	APPLI_FLT	Application error
%MWr.m.c.3		**CMD_FLT**	**Command Faults**
x0	Other	OVERRUN_CMD	Overrun condition while sending command
x1	Other	AXIS_IN_FLT	Invalid command due to axis in ErrorStop state
x2	Other	CMD_CODE_INV	Invalid command code
x3	Other	CMD_SEQ_INV	Invalid sequence of commands
x4	Other	BUFFER_FULL	Command rejected due to buffer full (Idle=FreeCmdBuf=0)
x5	Other	AXIS_NOT_REFERENCED	Positioning command rejected due to axis not referenced
x6	Other	TGT_POS_INV	Invalid target position
x7	Other	TGT_VEL_INV	Invalid target velocity
x8	Other	BUFFER_MODE_INV	Invalid buffer mode
%MWr.m.c.4		**ADJUST_FLT**	**Adjustment Parameter Faults**
x0	Other	OVERRUN_ADJUST	Overrun condition during adjustment instruction
x1	Other	SW_HIGH_LIMIT_INV	Invalid SW high limit
x2	Other	SW_LOW_LIMIT_INV	Invalid SW low limit
x3	Other	ACC_RATE_INV	Invalid acceleration rate
x4	Other	DEC_RATE_INV	Invalid deceleration rate
x5	Other	EMER_DEC_RATE_INV	Invalid emergency deceleration rate
x6	Other	START_FREQ_INV	Invalid start frequency
x7	Other	STOP_FREQ_INV	Invalid stop frequency
x8	Other	HOMING_VELO_INV	Invalid homing frequency

Object	Type	Symbol	Detail
%MWr.m.c.5		**AXIS_ERROR**	**Axis Errors**
x0	External	DRIVE_KO	Drive_Ready&Emergency input is off
x1	External	LIMIT_FLT	Limit crossing have been detected (LimitSwitch input)
x2	External	SW_HIGH_LIMIT_FLT	High software limit reached
x3	External	SW_LOW_LIMIT_FLT	Low software limit reached
x4	External	HOMING_FLT	Error during homing
x5	Unused		
x6	Unused		
x7	Unused		

Explicit Command Parameters %MWCmd

Explicit Command Parameters %MWCmd

Explicit Command Parameters %MWCmd

Object	Type	Symbol	Detail
%MWr.m.c.6	INT		
byte 0	Byte	CMD_Code	**1.** Frequency Generator **2.** Velocity Profile **3.** Absolute Positioning **4.** Relative Positioning **5.** Homing **6.** Set Position
byte 1	Byte	Unused	
%MWr.m.c.7	INT		
byte 0	Byte	Buffer_Mode	For Absolute and Relative Positioning commands: 0: Abort 1: Buffered 2: BlendingPrevious
byte 1	Byte	Unused	
%MDr.m.c.8	DINT	TGT_Position	For Absolute and Relative Positioning commands: Target Position / Distance (in pulses) For Homing and Set Position commands: Position value to set when reference signal is detected
%MDr.m.c.10	DINT	TGT_Velocity	Target velocity (in Hz)
%MWr.m.c.12			Reserved
%MWr.m.c.13	INT		
byte 0	Byte	CMD_SENT_NB	Sent command number (Read only)
byte 1	Byte		

Explicit Adjustment Parameters %MWAdjust

Explicit Adjustment Parameters %MWAdjust

Explicit Adjustment Parameters %MWAdjust

Object	Type	Symbol	Detail
%MDr.m.c.14	DINT	SW_High_Limit	Software Pulse Number High Limit Value from -2,147,483,647 to 2,147,483,647 Default: 2,147,483,647
%MDr.m.c.16	DINT	SW_Low_Limit	Software Pulse Number Low LimitValue from -2,147,483,648 to 2,147,483,646 Default: -2,147,483,648
%MWr.m.c.18	UINT	Start_Freq	0: No use of start frequency parameter (Default) Otherwise: value in Hz from 1 to 65,535
%MWr.m.c.19	UINT	Stop_Freq	No use of stop frequency parameter (Default) Otherwise: value in Hz from 1 to 65,535
%MWr.m.c.20	UINT	Acc_Rate	For all profiles except Frequency Generator Value from 10 to 32,500 Default: 100
%MWr.m.c.21	UINT	Dec_Rate	For all profiles except Frequency Generator Value from 10 to 32,500 Default: 100
%MWr.m.c.22	UINT	Emergency_Dec_Rate	Deceleration rate used in case of emergency stop (limits crossed, errors) Value from 10 to 32,500 Default: 100
%MWr.m.c.23	UINT	Homing_Velocity	For Homing Command: Value in Hz from 1 to 65,535 Default: 1
%MWr.m.c.24	UINT	Homing Time Out Value	For Homing Command: Only used when Homing I/O Settings parameter is set to 2. Value in ms from 0 to 65,535 Default: 65,535
%MWr.m.c.25	INT	Hysteresis (slack)	When Output mode is A/B Phases (reversed or not): Defines the numerical hysteresis to apply on PTO outputs in case of change of direction Value in pulses from 0 to 255 Default: 0
%MWr.m.c.26	INT	Reserved	Reserved

Implicit Exchange Language Objects Associated with the Application-Specific Function

At a Glance

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

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

Reminders

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

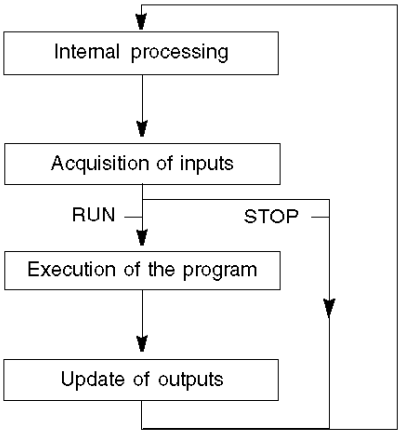
The outputs (%Q and %QW) are updated at the end of the task, only when the PLC is in RUN mode.

NOTE: When the task occurs in STOP mode, either of the following are possible, depending on the configuration selected:

- outputs are set to fallback position (fallback mode)
- outputs are maintained at their last value (maintain mode)

Figure

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



Implicit Status Objects %I, %IW

Implicit Status Objects %I, %IW

Implicit Status Objects %I, %IW

Object	Type	Symbol	Detail
%Ir.m.c.0	EBOOL	Drive_Ready&Emergency	Image of the corresponding physical input
%Ir.m.c.1	EBOOL	Counter_in_Position	Image of the corresponding physical input
%Ir.m.c.2	EBOOL	Origin	Image of the corresponding physical input
%Ir.m.c.3	EBOOL	Proximity&LimitSwitch	Image of the corresponding physical input
%Ir.m.c.4	EBOOL	Drive_Enable Level Output	State of the Drive_Enable Output
%Ir.m.c.5	EBOOL	Counter_Clear Output	State of the Counter_Clear Output
%IWr.m.c.0	INT		Current command
byte 0	Byte	Act_Cmd_Nb	Internal Command Number for the command being processed Value 0: means no command
byte 1	Byte		Unused
%IWr.m.c.1	INT		Next command
byte 0	Byte	Buf_Cmd_Nb	Internal Command Number for the command in buffer Value 0: means no command
byte 1	Byte		Unused
%IWr.m.c.2	INT		Last command executed
byte 0	Byte	Last_Cmd_Nb	Internal Command Number Value 0: means no command
byte 1	Byte		Unused
%IWr.m.c.3	INT		Status of last command executed
byte 0	Byte	Last_Result	Possible values: 0 = Done 1 = Aborted 2 = Error FF: Nothing
byte 1	Byte		Unused
%IWr.m.c.4	INT		History: Command executed previously
byte 0	Byte	Prev_Cmd_Nb	Internal Command Number Value 0: means no command
byte 1	Byte		Unused
%IWr.m.c.5	INT		History: Status of command executed previously

Object	Type	Symbol	Detail
byte 0	Byte	Prev_Result	Possible values: 0 = Done 1 = Aborted 2 = Error FF: Nothing (after Stop or ResetError)
byte 1	Byte		Unused
%IWr.m.c.6	INT	AXIS_STS	Status of the axis
byte 0	Byte		
x0	bool	AXIS_MOVING	The axis is moving
x1	bool	AXIS_STOPPING	The axis is in stopping state
x2	bool		Unused
x3	bool	AXIS_FLT	Axis in fault: Details on status in %MWStat
x4	bool		Unused
x5	bool		Unused
x6	bool	IN_VELOCITY	The axis is running at target frequency (for continuous profiles)
x7	bool	REFERENCED	
%IWr.m.c.7	INT	CMD_MGT	Specific objects for command management
byte 0	Byte		
x0	bool	Idle	0 = The channel is busy processing a command. 1 = No command is being processed by the channel (a new command can be sent)
x1	bool	FreeCmdBuf	0 = A command is waiting to be executed. 1 = No command has been buffered (a new command can be sent).
%IDr.m.c.8	DINT	Position	Current Position (in pulses)
%IDr.m.c.10	DINT	Frequency	Current Frequency (in Hz)

Implicit Event Data %IW

Implicit Event Data %IW

Implicit Event Data %IW

Object	Type	Symbol	Detail
%IWr.m.c.12	INT	EVT_Souce_Enabling	One bit per source
x0	bit	EVT_Position_Reached	Position reached
x1	bit	EVT_Referencing_Done	Referencing done
%IWr.m.c.13	INT	Unused	
%IDr.m.c.14	DINT	Current_ Position	Current Position (in pulses)

Implicit Command Objects %Q, %QW

Implicit Command Objects %Q, %QW

Implicit Command Objects %Q, %QW

Object	Type	Symbol	Detail
%Qr.m.c.0	EBOOL	Drive_Enable_Level	Value to send to the physical Enable_Drive output 0 = Disable (Default) 1 = Enable
%Qr.m.c.1	EBOOL	Counter_Clear	Value to send to the physical Clear_Counter output When active, command to clear the drive internal error counter, if option enabled by configuration (in Homing I/O Settings)
%Qr.m.c.2	EBOOL	Stop_Level	Command to stop the axis when high
%Qr.m.c.3	EBOOL	Reset_Axis_Error	When high, command to reset all axis errors: transition from ErrorStop to StandStill state.
%QWr.m.c.0	INT	EVT_Souce_Enabling	One bit per source 0 = Disable (Default) 1 = Enable
x0	bit	EVT_Position_Reached	Position reached
x1	bit	EVT_Referencing_Done	Referencing done
%QWr.m.c.1	INT	Disable Axis Faults	One bit per fault source
x0	bit	Drive_Ready&Emergency	0 = An error is reported when the Drive_Ready&Emergency input goes low and Drive_Enable physical output is active. (Default) 1 = Drive_Ready&Emergency input monitoring is disabled.
x1	bit	LimitSwitch	0 = An error is reported when the Proximity&LimitSwitch input goes high. (Default) 1 = Proximity&LimitSwitch input monitoring is disabled.
x2	bit	SW Limits	0 = Enable software limits control (Default) 1 = Disable software limits control

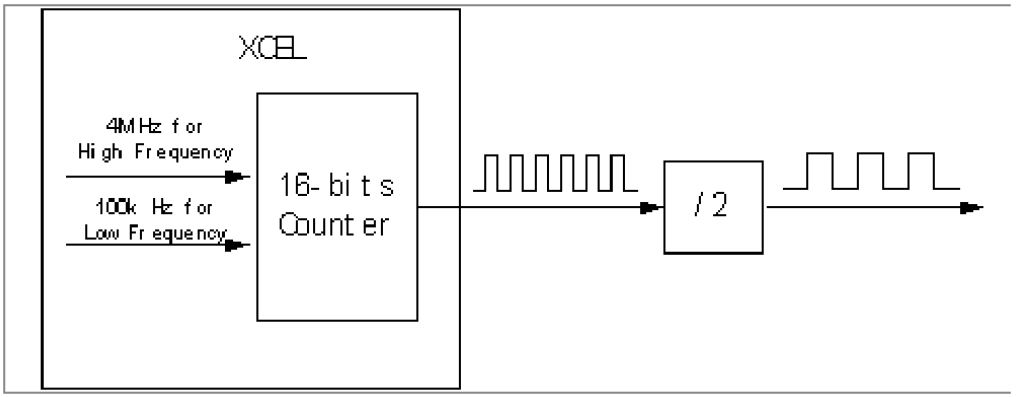
Limitations and Performances

15

Key Performances

Pulse Generator

This function unit generates a Pulse Output as follows:



The internal counter uses 4 MHz as the Clock Source for high-frequency output from 100 Hz to 400 kHz.

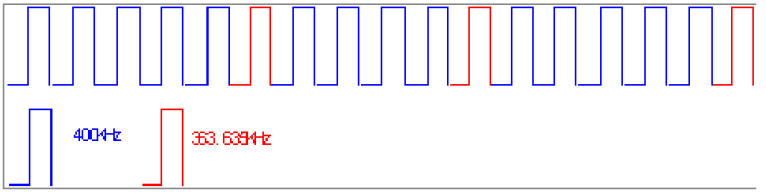
The internal counter uses 100 kHz as the Clock Source for low-frequency output from 2 Hz to 100 Hz. (The output here refers to the one before external frequency-dividing circuit)

In high-frequency case, the output obtained directly from the internal counter has the frequency as 4M / Modulo (Modulo is an integral value, which is put into the counter to divide the Clock Source). We can see that a 4 MHz Clock Source is not sufficient to generate all the frequencies in the range from 100 Hz to 400 kHz with a 0.5% accuracy. For some frequencies, a specific algorithm is used to correct the output. This algorithm makes the output pulse vary between the Clock Source divided by Modulo and divided by Modulo+1. Aan appropriate variation ratio is implemented to make sure that the average frequency reaches a 0.5% accuracy.

For example, if the desired output frequency is 393 kHz:

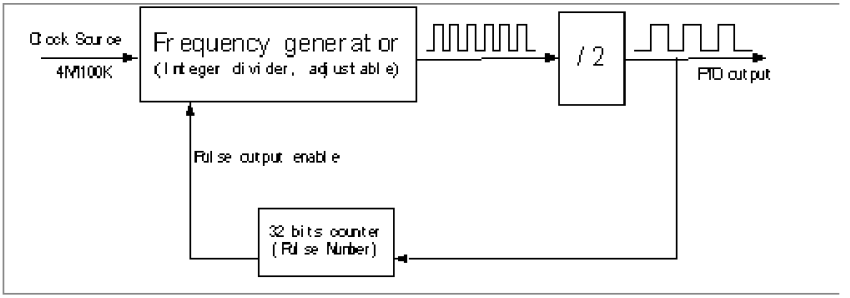
The Modulo in this case is 10, the real output pulse will vary between 400 kHz and 363.6363 kHz, and the ratio is between 4:1 and 5:1.

The real output picture is as follows:



Pulse Number

Pulse Generator Loop (2 ms):



There is a 32-bit counter in every PTO channel to count the pulse output number in order to ensure that there is no error on the pulse number.

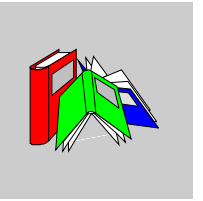
Commands Processing

Only one command can be sent and processed at each PLC task cycle.

In case of a sequence of commands:

- If BufferMode is Aborted, the response time will be related to the PLC task cycle. That is to say that the current command will not be stopped, and the new command will not be started before the next cycle.
- If BufferMode is Buffered or BlendingPrevious, the response time is independent from the PLC task cycle (considering that the command was sent at least one cycle before the current command is completed).

Glossary



0-9

%I

According to the IEC standard, %I indicates a language object of type discrete IN.

%IW

According to the IEC standard, %IW indicates a language object of type analog IN.

%KW

According to the IEC standard, %KW indicates a language object of type constant word.

%M

According to the IEC standard, %M indicates a language object of type memory bit.

%MW

According to the IEC standard, %MW indicates a language object of type memory word.

%Q

According to the IEC standard, %Q indicates a language object of type discrete OUT.

%QW

According to the IEC standard, %QW indicates a language object of type analog OUT.

A

A/B Phases

Output mode in which both output signals (For example: phase A and phase B) are a pulse train signal at the same frequency (target frequency) and for which the direction is given by the phase difference between A and B.

Acceleration

The rate at which something increases its velocity. Acceleration is usually measured in units of velocity change for each unit of time (inches/second (velocity) per second (time)) and in this example is given either in ms or Hz/2ms.

Accuracy

The relative status of something compared to its absolute or perfect value. In motion control this will most often be a position description.

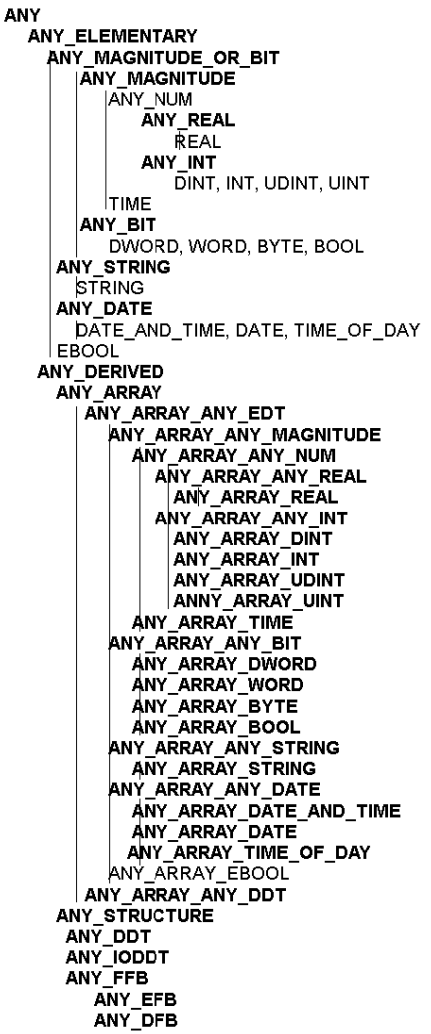
A command may be sent to move 4.0"(101.6 mm): the accuracy of the system will be defined by how close to the absolute value of 4.0"(101.6 mm) the system can complete the movement. Accuracy may be defined as a one time incident or the average over a number of cycles or motions.

Positioning accuracy will normally be defined in terms of deviation (+/- from theoretical) or limits of acceptable variation from a theoretical value. For example 3.8"-4.2" (96.52 mm - 106.68 mm) could define acceptable limits of variation around a theoretical point of 4.0" (101.6 mm)

ANY

There is a hierarchy among the various data types. In the DFBs, it is sometimes possible to declare variables that can contain several types of values. In that case we use `ANY_xxx` types.

The figure below describes this hierarchical structure:



ARRAY

An `ARRAY` is a table containing elements of a single type.

The syntax is as follows: `ARRAY [<limits>] OF <Type>`

Example:

ARRAY [1..2] OF BOOL is a one-dimensional table with two elements of type BOOL.

ARRAY [1..10, 1..20] OF INT is a two-dimensional table with 10x20 elements of type INT.

Axis

An axis is a mechanical part driven by an electric motor. It serves to guide rotation or translation.

B

BlendingPrevious

Buffer byte value for which one positioning command follows another one. The next command starts as soon as the previous one reaches its Target_Position and will begin at the previous Target_Velocity.

BOOL

BOOL is the abbreviation for the Boolean type. This is the basic data type in computing. A BOOL variable can have either of the following two values: 0 (FALSE) or 1 (TRUE).

A bit extracted from a word is of type BOOL, for example: %MW10.4.

Buffer

The buffer is an input (a byte) that defines how two conscutive commands will be treated regarding Absolute and Relative Positioning commands . There are 3 possible values: Abort, value = 0, the second command cancels the one running and starts immediatly; Buffered, value = 1, the second command starts once the previous one is finalized (axis is stopped); BlendingPrevious, value = 2, explanation in the BlendingPrevious glossary entry.

BYTE

When 8 Bits are grouped together, they are called a BYTE. You can enter a BYTE either in binary mode or in base 8.

The BYTE type is encoded in an 8 bit format which, in hexadecimal format, ranges from 16#00 to 16#FF.

C

Counter_in_Position

The Counter_in_Position input (sometimes called Position_Completed) corresponds to an output of the drive indicating that the drive's internal position error counter is empty. This input can be used for homing processes to ensure a synchronization between the PTO channel's position counter and the drive.

Current Position

The position of an axis relative to the requested position. This may be the position at the end of the move or the position at any time during the move.

CW / cCW

Clock Wise / Counter Clock Wise: Output mode in which each output signal (i.e. CW signal and CCW signal) is alternatively the pulse train signal according to the direction.

D

DDT

DDT is the abbreviation of Derived Data Type.

A derived data type is a set of elements with the same type (`ARRAY`) or with different types (structure).

Deceleration

The rate at which something decreases its velocity. Deceleration is usually measured in units of velocity change for each unit of time (Inches/second (velocity) per second (time)) and in this example is given either in ms or Hz/2ms.

DFB

DFB is the abbreviation of Derived Function Block.

DFB types are function blocks that can be defined in ST, IL, LD or FBD language.

Using these DFB types in an application makes it possible to:

- simplify the design and entry of the program;
- make the program easier to read;
- make it easier to debug;
- reduce the amount of code generated.

DINT

DINT is the abbreviation of Double INTeger (encoded in 32 bits).

The upper/lower limits are as follows: -(2 to the power of 31) to (2 to the power of 31) - 1.

Example:

-2147483648, 2147483647, 16#FFFFFFFF.

Drive

An electronic device that translates a motion controller command into a electrical current that controls a motor.

E

EBOOL

EBOOL is the abbreviation of Extended BOOLean. An EBOOL type has a value (0 (FALSE) or 1 (TRUE), but also rising or falling edges and forcing functions.

An EBOOL variable occupies one byte in memory.

The byte contains the following information:

- one bit for the value;
- one bit for the history (whenever the object changes state, the value is copied to the history bit);
- one bit for forcing (equal to 0 if the object is not forced, or 1 if the bit is forced).

The default value of each bit is 0 (FALSE).

EF

EF is the abbreviation of Elementary Function.

This is a block used in a program which performs a predefined logical function.

A function does not have any information on the internal state. Several calls to the same function using the same input parameters will return the same output values. Information on the graphic form of the function call can be found in the "functional block (instance)". Unlike a call to a function block, function calls include only an output which is not named and whose name is identical to that of the function. In FBD, each call is indicated by a unique number via the graphic block. This number is managed automatically and cannot be modified.

Other functions using the SDKC can be developed with the development kit.

Elementary function

See EF.

EN

EN stands for **EN**able; it is an optional block input. When the EN input is enabled, an ENO output is set automatically.

If EN = 0, the block is not enabled; its internal program is not executed, and ENO is set to 0.

If EN = 1, the block's internal program is run and ENO is set to 1. If an error occurs, ENO is set to 0.

If the EN input is not connected, it is set automatically to 1.

ENO

ENO stands for **E**rror **NO**tification; this is the output associated with the optional input EN.

If ENO is set to 0 (because EN = 0 or in case of an execution error):
- the status of the function block outputs remains the same as they were during the previous successful scanning cycle.
- the output(s) of the function, as well as the procedures, are set to "0".

Event

Task performed with priority over all other tasks, to reduce the response time of the application to certain events.

F

FBD

FBD is the abbreviation of Function Block Diagram.

FBD is a graphical programming language that works like a flowchart. By adding simple logical blocks (AND, OR, etc.), each function or function block in the program is represented in this graphical format. For each block, the inputs are on the left and the outputs on the right. Block outputs can be linked to inputs of other blocks in order to create complex expressions.

FFB

Collective term for EF (elementary function), EFB (elementary function block) and DFB (derived function block).

Function

See EF.

Function Block Diagram

See FBD.

H

Home Position

A reference position for all absolute positioning movements. Usually defined by a home limit switch and/or encoder marker. Normally set by a homing command and retained as long as control system is operational.

Homing

Locating a unique reference position for axis calibration.

I

IL

IL is the abbreviation of Instruction List.

This language is a series of basic instructions.

It is very close to assembly language used to program processors.

Each instruction is made up of an instruction code and an operand.

INT

`INT` is the abbreviation of single INTeger (encoded in 16 bits).

The upper/lower limits are as follows: -(2 to the power of 15) to (2 to the power of 15) - 1.

Example:

`-32768, 32767, 2#1111110001001001, 16#9FA4.`

IODDT

IODDT is the abbreviation of Input/Output Derived Data Type.

The term IODDT indicates a structured data type representing a module or a channel of a PLC module. Each expert module has its own IODDTs.

L

LD

LD is the abbreviation of Ladder Diagram.

LD is a programming language that represents instructions to be executed as graphical diagrams very similar to electrical diagrams (contacts, coils, etc.).

Limit Switch

The Proximity&LimitSwich input is used to signal that the axis has reached a limit of the valid area (either on the positive or the negative side), except in case set homing type is short cam with marker.

Long Cam Negative

Homing procedure that enables to reference the axis by searching for a negative limit switch-type sensor.

Long Cam Positive

Homing procedure that enables to reference the axis by searching for a positive limit switch-type sensor.

Lxm

Abbreviation for Lexium, a Schneider Electric drive brand.

M

Motion

The act of changing position, the PTO module has 2 different motion types:

1. Continuous: the drive does a persistent movement which is stopped only by activating the STOP command.
2. Discrete: the drive describes a movement cycle with a start and an end.

MSP

Motion Single axis controller PTO.

O

Open Loop/Close Loop

Open loop control refers to a motion control system with no external sensors to provide position or velocity correction signals.

A closed loop control is a motion control system that has position and velocity feedback to generate a correction signal by comparing its position and velocity to desired parameters. Feedback devices are typically encoders, resolvers, LVTDs and/or tachometers.

Origin

The origin input is used for all types of homing commands to signal that the axis has reached the reference point.

Overcurrent

Any current in excess of the rated current of the drive to maintain or move to a new position at a given velocity and acceleration or deceleration rate.

P

PLCopen

PLCopen is a Vendor- and product-independent worldwide association on a standard regarding: programming. Effectively this standardization is done by defining libraries of reusable components. In this way the programming is less hardware dependent, the reusability of the application software increased, the cost involved in training and support reduced, and the application becomes scalable.

Position Loop

Portion of the command signals that generates the position information based on position feedback.

Positioning

Specifying a move by giving a target position, a velocity and an acceleration and deceleration. The target position can be an absolute position, or a relative position from the current position.

PowerSuite

PowerSuite is a Schneider Electric software which allows configuration of the Schneider Electric drives (Lexium, ATV, TeSys, ATS)

Procedure

Procedures are technically functional views. The only difference with elementary functions is the fact that procedures can include more than one output and that they handle the `VAR_IN_OUT` data type. In appearance, procedures are no different from elementary functions.

Procedures are an extension to the IEC 61131-3 standard.

Profile

Graphical representation of movement. This can be position vs. time, velocity vs.time or torque vs. time.

Proximity

The Proximity&LimitSwitch input is used as Proximity signal during homing command when set homing type is short cam with marker. The signal represents a proximity area around the reference point. The accurate position of the reference point is given by the zero marker signal.

PTO

Pulse Train Output

Pulse + Direction

Output mode in which the first output signal (CW, i.e. Pulse) is the pulse train signal, while the second output signal (CCW, i.e. Direction) gives the direction.

R

Referencing

Procedure to set the feedback device relative to a specific reference point.

RS422

Standard interface multi-port serial communication port.

S

Shielded Cable

A cable that has a metallic sleeve wrapped around all of the conductors that comprise its center. The metal sleeve is then grounded to eliminate the effects of electrical noise on the signals being carried by the cable.

Short Cam

Homing procedure that enables referencing of the axis by searching for an absolute positioned external physical switch (reference on the negative side of the absolute switch/short cam).

Short Cam with Marker

Homing procedure that enables referencing the axis by searching for Zero pulse (also called Marker or reference pulse) in encoder within a proximity area delimited by an absolute switch (short cam).

Short Cam with Negative Limit

Homing procedure that enables referencing of the axis by searching for an absolute positioned external physical switch (reference on the negative side of the absolute switch/short cam) within an area delimited on the negative side by a limit switch.

Short Cam with Positive Limit

Homing procedure that enables referencing the axis by searching for an absolute positioned external physical switch (reference on the negative side of the absolute switch/short cam) within an area delimited on the positive side by a limit switch.

Slack Correction

Slack correction is used to define the number of ouput pulses to ignore after every change of direction.

ST

ST is the abbreviation of Structured Text.

The structured literal language is a developed language similar to computer programming languages. It can be used to organize a series of instructions.

SW Limit

Software limits (High and Low) that define the field in which the application can run. These limits are always included in the axis physical limits.

T

TIME

The TIME type expresses a time in milliseconds. Encoded in 32 bits, this type can be used to obtain times from 0 to 2^{32}-1 milliseconds.

The TIME type has the following units: days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms). A literal value of type TIME is represented by a combination of the preceding types prefixed with T#, t#, TIME# or time#.

Examples: T#25h15m, t#14,7S, TIME#5d10h23m45s3ms

Twisted Pair

Two wires twisted together for the purpose of eliminating the effect of electrical noise.

U

UDINT

UDINT is the abbreviation of Unsigned Double INTeger (encoded in 32 bits). The upper/lower limits are as follows: 0 to (2 to the power of 32) - 1.

Example:

0,4294967295,2#11111111111111111111111111111111,8#37777777777, 16#FFFFFFFF.

UINT

UINT is the abbreviation of the Unsigned INTeger format (encoded in 16 bits). The upper/lower limits are as follows: 0 to (2 to the power of 16) - 1.

Example:

0, 65535, 2#1111111111111111, 8#177777, 16#FFFF.

USIC

The USIC (Universal Signal Interface Converter) is an interface adapter that is used as a universal adapter for a pulse/direction interface to a master controller (for example/ PLC).

V

Variable

Memory entity of type BOOL, WORD, DWORD, etc., whose contents can be modified by the program currently running.

Velocity

The speed at which a motor or mechanical system runs.

W

WORD

The type WORD is encoded in a 16 bit format and is used to perform processing on a series of bits.

This table shows the upper/lower limits of each of the bases that can be used:

Base	Lower limit	Upper limit
Hexadecimal	16#0	16#FFFF
Octal	8#0	8#177777
Binary	2#0	2#1111111111111111

Examples of representation

Data	Representation in one of the bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Write_cmd

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

Index

H

I

L

M

O

P

S