EcoStruxure Machine Expert - Basic Example Guide

Importing Twido Drive Macros to M221 “xSample_ Twido_Macro_Drive_Conversion.smbe”

12/2018

EIO0000003373.00

www.schneider-electric.com

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

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

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, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

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

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

BEFORE YOU BEGIN

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

⚠ WARNING

UNGUARDED EQUIPMENT

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

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

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

START-UP AND TEST

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check be made and that enough time is allowed to perform complete and satisfactory testing.

⚠ WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

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

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

OPERATION AND ADJUSTMENTS

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer’s instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer’s instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

About the Book

At a Glance

Document Scope

This document describes how to convert a TwidoSuite application that contains drive macros to EcoStruxure Machine Expert - Basic. It also explains how to adapt this converted EcoStruxure Machine Expert - Basic project to various setups.

The examples described here are intended for learning purposes only. In general, they are intended to help you understand how to develop, test, commission, and integrate application logic and/or the device wiring of the equipment associated with your own design in your control systems. The examples are not intended to be used directly on products that are part of a machine or process.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Do not include any wiring information, programming or configuration logic, or parameter values from any of the examples in your machine or process without thoroughly testing your entire application. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

This document and its related EcoStruxure Machine Expert - Basic project file focus on specific instructions and function blocks provided with EcoStruxure Machine Expert - Basic, and on specific features available in EcoStruxure Machine Expert - Basic. They are intended to help you understand how to develop, test, commission, and integrate applicative software of your own design in your control systems.

The example is intended for new EcoStruxure Machine Expert - Basic users who already have some degree of expertise in the design and programming of control systems.

Validity Note

This document has been updated for the release of EcoStruxure^{TM} Machine Expert - Basic V1.0.

Related Documents

Title of Documentation	Reference Number
ATV61_communication_parameters_EN_V5.8_IE29	*1760661*
EcoStruxure Machine Expert - Basic Generic Functions - Library Guide	*EIO0000003289 (ENG)* *EIO0000003290 (FRE)* *EIO0000003291 (GER)* *EIO0000003292 (SPA)* *EIO0000003293 (ITA)* *EIO0000003294 (CHS)* *EIO0000003295 (POR)* *EIO0000003296 (TUR)*
EcoStruxure Machine Expert - Basic Example Guide - Altivar Control from Serial Modbus	*EIO0000003363*
EcoStruxure Machine Expert - Basic Example Guide - Importing Twido COMM Macros to M221	*EIO0000003372*

You can download these technical publications and other technical information from our website at https://www.schneider-electric.com/en/download

Product Related Information

⚠ 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, power outage and restart.
- 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.
- Observe all accident prevention regulations and local safety guidelines.[1]
- Each implementation of this equipment 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.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not include any wiring information, programming or configuration logic, or parameter values from any of the examples in your machine or process without thoroughly testing your entire application.

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

Chapter 1
Example Description

What Is in This Chapter?

This chapter contains the following topics:

Overview

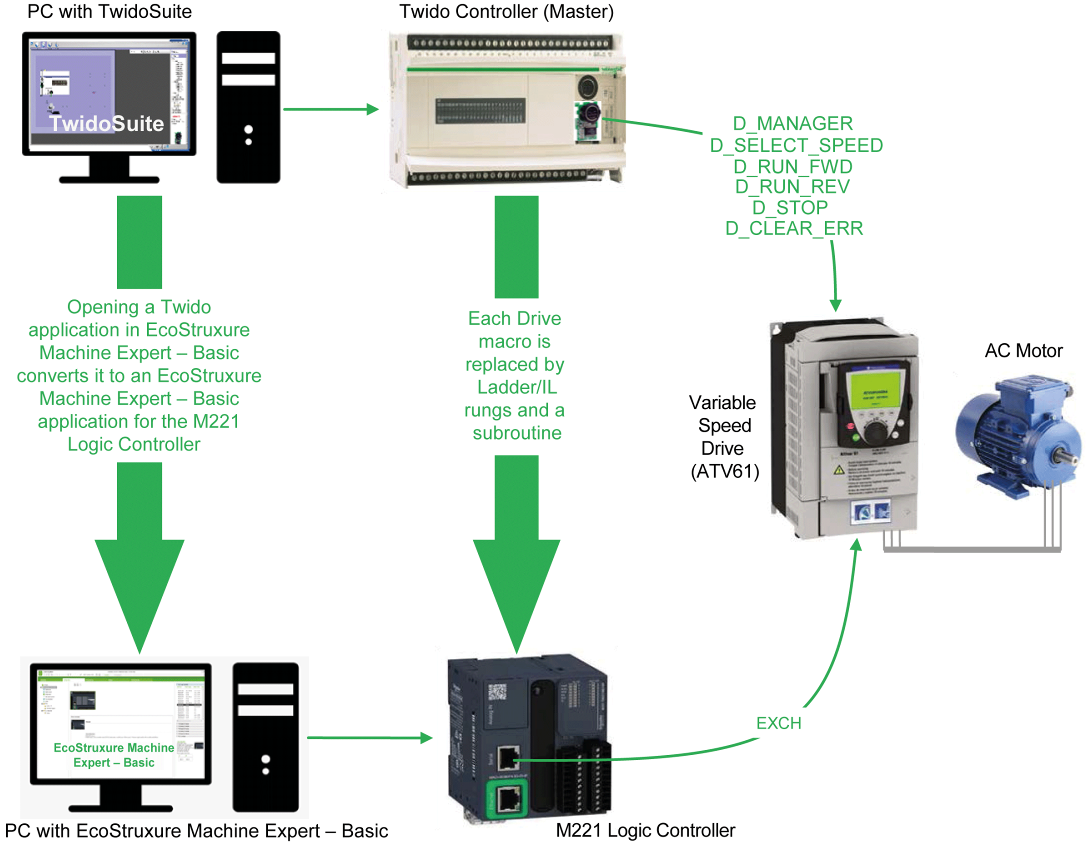
General

The Example guide and its corresponding project template, included with EcoStruxure Machine Expert - Basic, help you to understand the code generated after importing a Twido project containing Twido drive macros into EcoStruxure Machine Expert - Basic. This allows you to have an understanding of how variable speed drives work. If your configuration changes, it also shows how to modify the generated code to adapt it to a modified configuration.

This document describes:

- The conversion of a TwidoSuite project to an EcoStruxure Machine Expert - Basic project for the M221 Logic Controller.
- How to modify the resulting program to adapt it to a modified configuration.
- The operating principles of the drive macros.

The objective of the example is illustrated below:


PC with TwidoSuite
TwidoSuite
Twido Controller (Master)
D_MANAGER
D_SELECT_SPEED
D_RUN_FWD
D_RUN_REV
D_STOP
D_CLEAR_ERR
Opening a Twido application in EcoStruxure Machine Expert – Basic converts it to an EcoStruxure Machine Expert – Basic application for the M221 Logic Controller
Each Drive macro is replaced by Ladder/IL rungs and a subroutine
Variable Speed Drive (ATV61)
AC Motor
EXCH
EcoStruxure Machine Expert – Basic
PC with EcoStruxure Machine Expert – Basic
M221 Logic Controller

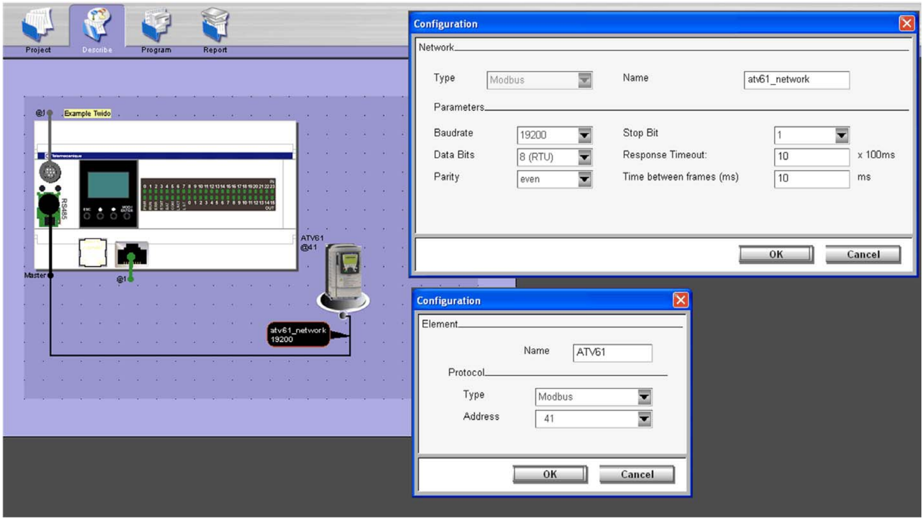
Automatic Conversion of Twido Drive Macros to EcoStruxure Machine Expert - Basic

Twido Project Example Using Drive Macros

This functioning Twido project implements drive macros. The TwidoSuite project file is available under the name `xSample_Twido_Macro_Drive_Conversion.xar`.

This project uses the Modbus protocol to communicate with a variable speed drive (an ATV 61 for this example) with Modbus slave address 41. The instance number of the macro is 3 and its starting address is `%MW80`.

This figure presents the communication parameters that are displayed in the **Describe** tab of TwidoSuite:



Configuration

Network

Type: Modbus
Name: atv61_network

Parameters

Baudrate: 19200
Data Bits: 8 (RTU)
Parity: even
Stop Bit: 1
Response Timeout: 10 x 100ms
Time between frames (ms): 10 ms

OK Cancel

Configuration

Element

Name: ATV61

Protocol

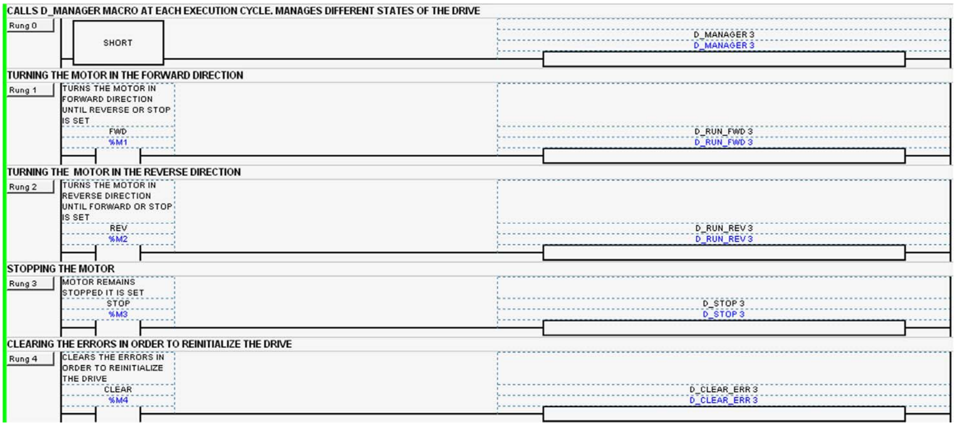
Type: Modbus
Address: 41

OK Cancel

The project focuses on how to use drive macros. It consists of rungs that call macros. Before being deployed in the field, it would also need to manage operating modes.

Below are some graphics from the **Program** editor of the Twido example.

Turning the motor in different directions, stopping, or reinitializing:



CALLS D_MANAGER MACRO AT EACH EXECUTION CYCLE. MANAGES DIFFERENT STATES OF THE DRIVE

Rung 0
SHORT
D_MANAGER 3
D_MANAGER 3

TURNING THE MOTOR IN THE FORWARD DIRECTION

Rung 1
TURNS THE MOTOR IN FORWARD DIRECTION UNTIL REVERSE OR STOP IS SET
FWD
%M1
D_RUN_FWD 3
D_RUN_FWD 3

TURNING THE MOTOR IN THE REVERSE DIRECTION

Rung 2
TURNS THE MOTOR IN REVERSE DIRECTION UNTIL FORWARD OR STOP IS SET
REV
%M2
D_RUN_REV 3
D_RUN_REV 3

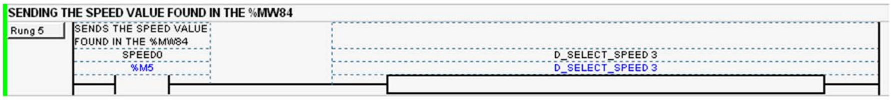
STOPPING THE MOTOR

Rung 3
MOTOR REMAINS STOPPED IT IS SET
STOP
%M3
D_STOP 3
D_STOP 3

CLEARING THE ERRORS IN ORDER TO REINITIALIZE THE DRIVE

Rung 4
CLEARS THE ERRORS IN ORDER TO REINITIALIZE THE DRIVE
CLEAR
%M4
D_CLEAR_ERR 3
D_CLEAR_ERR 3

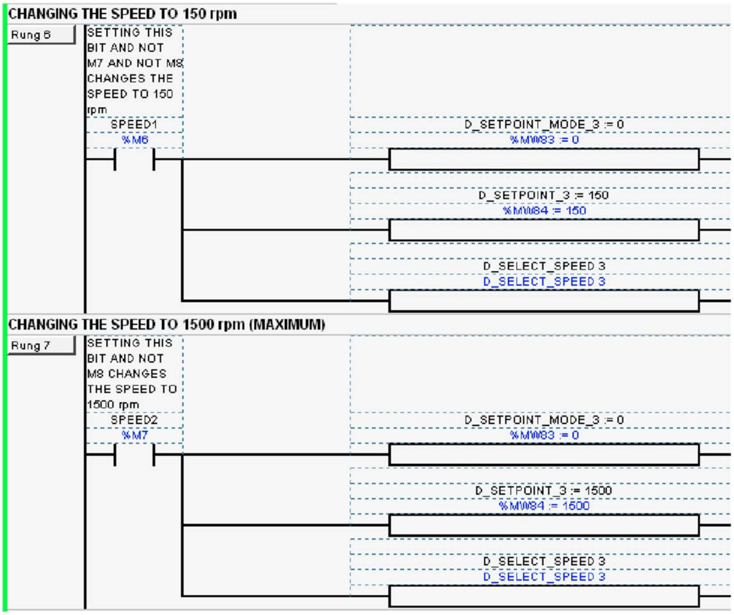
Sending the speed value found in `%MW84` (used with the animation table to set modifiable speeds):



SENDING THE SPEED VALUE FOUND IN THE %MW84

Rung 5
SENDS THE SPEED VALUE FOUND IN THE %MW84
SPEED0
%M5
D_SELECT_SPEED 3
D_SELECT_SPEED 3

Choosing different turning speeds:

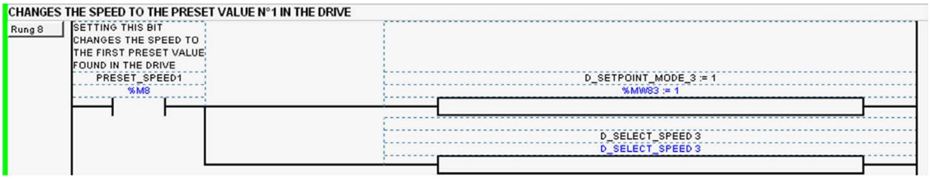


CHANGING THE SPEED TO 150 rpm

Rung 6
SETTING THIS BIT AND NOT M7 AND NOT M8 CHANGES THE SPEED TO 150 rpm
SPEED1
%M6
D_SETPOINT_MODE_3 := 0
%MW83 := 0
D_SETPOINT_3 := 150
%MW84 := 150
D_SELECT_SPEED 3
D_SELECT_SPEED 3

CHANGING THE SPEED TO 1500 rpm (MAXIMUM)

Rung 7
SETTING THIS BIT AND NOT M8 CHANGES THE SPEED TO 1500 rpm
SPEED2
%M7
D_SETPOINT_MODE_3 := 0
%MW83 := 0
D_SETPOINT_3 := 1500
%MW84 := 1500
D_SELECT_SPEED 3
D_SELECT_SPEED 3



CHANGES THE SPEED TO THE PRESET VALUE N°1 IN THE DRIVE

Rung 8

SETTING THIS BIT CHANGES THE SPEED TO THE FIRST PRESET VALUE FOUND IN THE DRIVE

PRESET_SPEED1
%M8

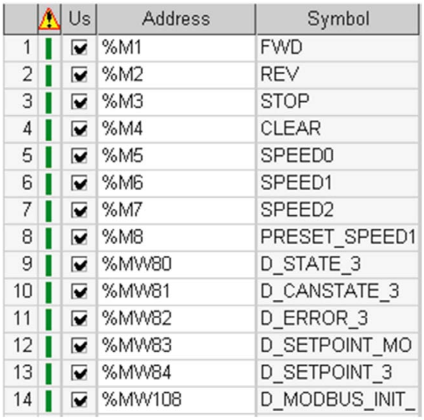
D_SETPOINT_MODE_3 := 1
%MW83 := 1

D_SELECT_SPEED 3
D_SELECT_SPEED 3

This project:

- Calls the `D_Manager` macro at each execution cycle to pass implicitly from one state to another.
- Turns the motor in the forward direction when `%M1` is set to TRUE.
- Turns the motor in the reverse direction when `%M2` is set to TRUE.
- Stops the motor when `%M3` is set to TRUE.
- Clears the detected errors when `%M4` is set to TRUE. This reinitializes the drive so that the other commands can be sent.
- Sends the speed value found at `%MW84` when `%M5` is set to TRUE.
- Sets the speed to 150 rpm when `%M6` is set to TRUE.
- Sets the speed to 1500 rpm when `%M7` is set to TRUE.
- Modifies the speed to the preset value 1 when `%M8` is set to TRUE.

An animation table is provided to manage the progress of the program:



	⚠	Us	Address	Symbol
1	🟩	☑	%M1	FWD
2	🟩	☑	%M2	REV
3	🟩	☑	%M3	STOP
4	🟩	☑	%M4	CLEAR
5	🟩	☑	%M5	SPEED0
6	🟩	☑	%M6	SPEED1
7	🟩	☑	%M7	SPEED2
8	🟩	☑	%M8	PRESET_SPEED1
9	🟩	☑	%MW80	D_STATE_3
10	🟩	☑	%MW81	D_CANSTATE_3
11	🟩	☑	%MW82	D_ERROR_3
12	🟩	☑	%MW83	D_SETPOINT_MO
13	🟩	☑	%MW84	D_SETPOINT_3
14	🟩	☑	%MW108	D_MODBUS_INIT_

Automatic Conversion of the Twido Project into EcoStruxure Machine Expert - Basic

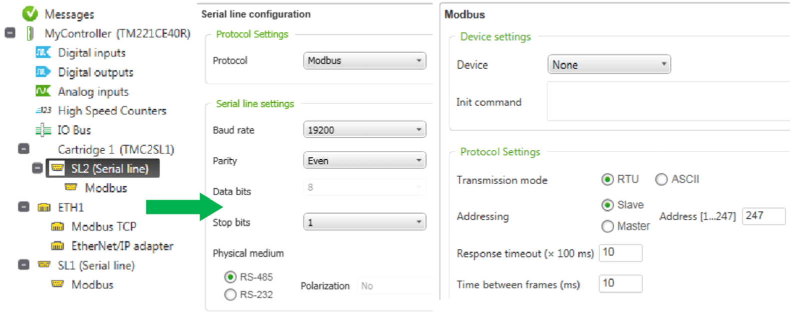
This TwidoSuite project is converted when opened in EcoStruxure Machine Expert - Basic. However, M221 Logic Controller serial line configuration is different from Twido controllers. For details, refer to Software Configuration *(see page 18)*.

Hardware Configuration

Verify that the hardware configuration displayed in the **Configuration** tab of EcoStruxure Machine Expert - Basic matches the physical configuration of your logic controller. In particular, verify the reference of the logic controller, and that all expansion module and cartridge references match those used in the physical configuration, and appear in the correct sequence. Adapt the EcoStruxure Machine Expert - Basic application as necessary.

Software Configuration

Verify that the serial line configuration of the M221 Logic Controller matches the previous settings in TwidoSuite. To do so, choose **MyController** → **Cartridge 1 (TMC2SL1)** → **SL2 (Serial line)** on the **Configuration** tab as shown below:



Serial line configuration

Protocol Settings

Protocol: Modbus

Serial line settings

Baud rate: 19200

Parity: Even

Data bits: 8

Stop bits: 1

Physical medium: ● RS-485 ○ RS-232

Polarization: No

Modbus

Device settings

Device: None

Init command:

Protocol Settings

Transmission mode: ● RTU ○ ASCII

Addressing: ● Slave ○ Master

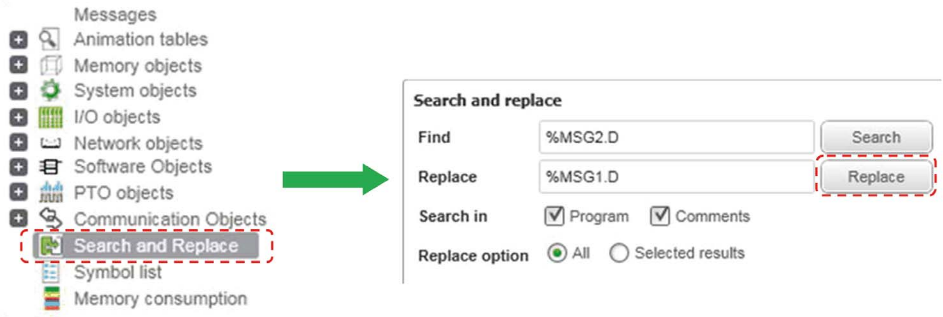
Address [1...247]: 247

Response timeout (× 100 ms): 10

Time between frames (ms): 10

NOTE: If you are not using the same serial line, for example, you have switched from SL1 to SL2 or vice versa, you must modify each `EXCH` instruction and `MSG` function block to use the other serial line.

To do so, you can use the **Search and Replace** tool under the **Tools** tab in EcoStruxure Machine Expert - Basic. This figure shows **Search and Replace** options:



Search and replace

Find: %MSG2.D

Replace: %MSG1.D

Search in: ☑ Program ☑ Comments

Replace option: ● All ○ Selected results

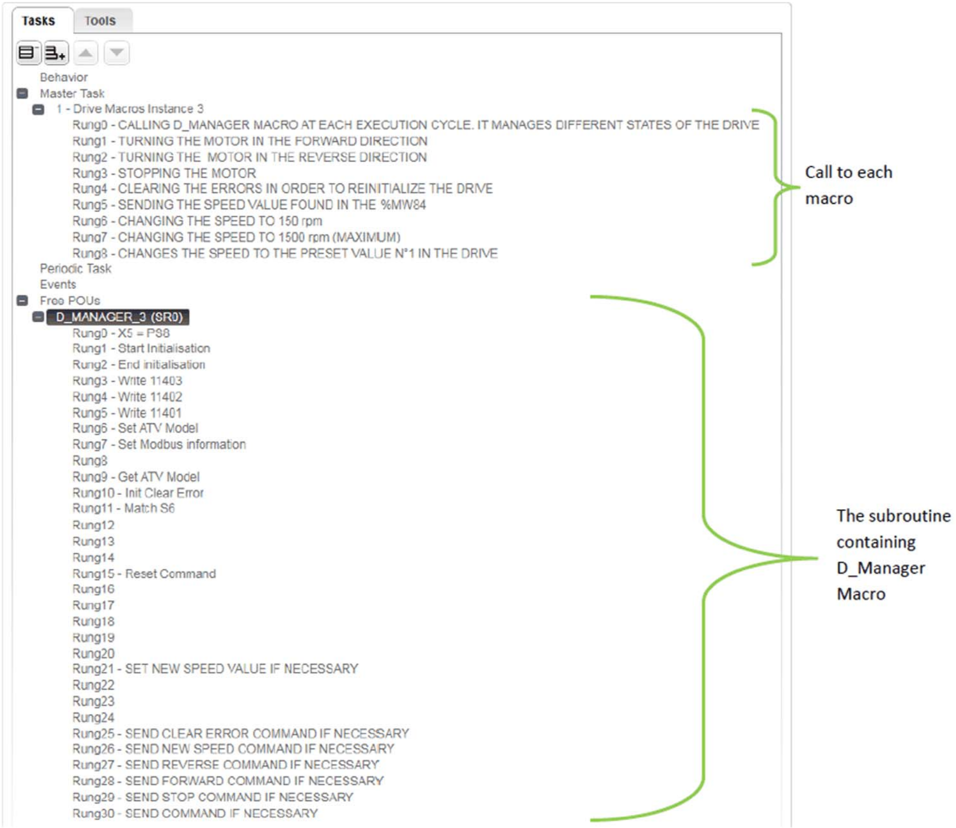
Opening the Template

A template is provided that corresponds to the conversion result of this Twido project. Open the template and use it to follow this document. To open the template, click ⚙ in the top left corner, click the **Templates** button, select `xSample_Twido_Macro_Drive_Conversion.smbe`, and click **Open Template**.

Conversion Result

The program in the template is functionally equivalent to the Twido program. However, as EcoStruxure Machine Expert - Basic does not support macros, each macro is replaced by a rung with equivalent instructions, and one subroutine (`SR0`).

The following graphics present the result of this conversion:

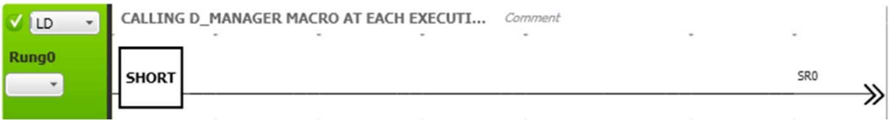


- Behavior
- Master Task
  - 1 - Drive Macros Instance 3
    - Rung0 - CALLING D_MANAGER MACRO AT EACH EXECUTION CYCLE. IT MANAGES DIFFERENT STATES OF THE DRIVE
    - Rung1 - TURNING THE MOTOR IN THE FORWARD DIRECTION
    - Rung2 - TURNING THE MOTOR IN THE REVERSE DIRECTION
    - Rung3 - STOPPING THE MOTOR
    - Rung4 - CLEARING THE ERRORS IN ORDER TO REINITIALIZE THE DRIVE
    - Rung5 - SENDING THE SPEED VALUE FOUND IN THE %MW84
    - Rung6 - CHANGING THE SPEED TO 150 rpm
    - Rung7 - CHANGING THE SPEED TO 1500 rpm (MAXIMUM)
    - Rung8 - CHANGES THE SPEED TO THE PRESET VALUE N°1 IN THE DRIVE
- Periodic Task
- Events
- Free POUs
  - D_MANAGER_3 (SR0)
    - Rung0 - X5 = P38
    - Rung1 - Start Initialisation
    - Rung2 - End initialisation
    - Rung3 - Write 11403
    - Rung4 - Write 11402
    - Rung5 - Write 11401
    - Rung6 - Set ATV Model
    - Rung7 - Set Modbus information
    - Rung8
    - Rung9 - Get ATV Model
    - Rung10 - Init Clear Error
    - Rung11 - Match S6
    - Rung12
    - Rung13
    - Rung14
    - Rung15 - Reset Command
    - Rung16
    - Rung17
    - Rung18
    - Rung19
    - Rung20
    - Rung21 - SET NEW SPEED VALUE IF NECESSARY
    - Rung22
    - Rung23
    - Rung24
    - Rung25 - SEND CLEAR ERROR COMMAND IF NECESSARY
    - Rung26 - SEND NEW SPEED COMMAND IF NECESSARY
    - Rung27 - SEND REVERSE COMMAND IF NECESSARY
    - Rung28 - SEND FORWARD COMMAND IF NECESSARY
    - Rung29 - SEND STOP COMMAND IF NECESSARY
    - Rung30 - SEND COMMAND IF NECESSARY

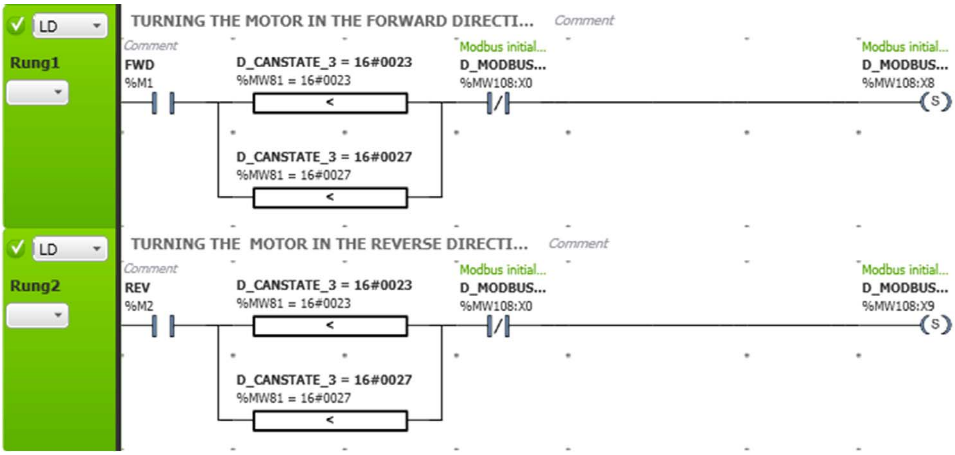
Call to each macro

The subroutine containing D_Manager Macro

This figure shows how the subroutine `D_Manager` macro equivalent is called:

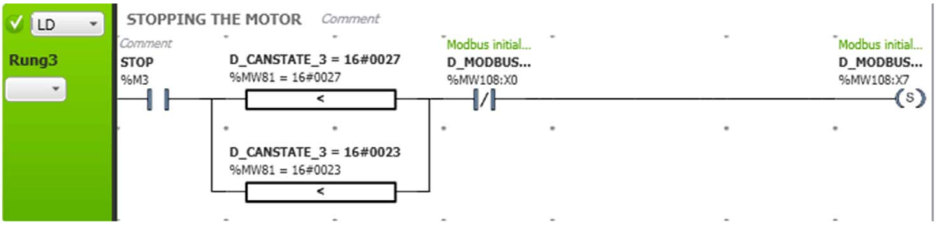


This figure shows turning the motor in forward and reverse directions. `D_Run_Fwd` and `D_Run_Rev` equivalents:



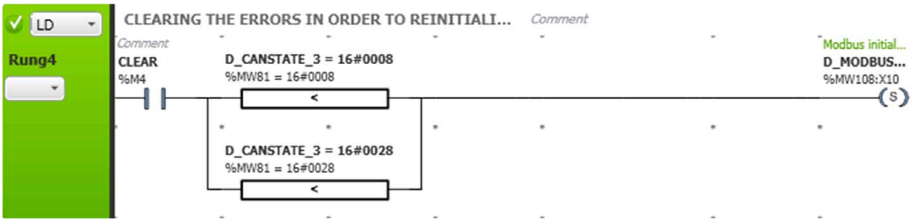
Setting to TRUE the bit 8 of `%MW108` turns the motor in the forward direction. Setting bit 9 to TRUE turns it in the reverse direction.

Stopping the motor. `D_Stop` equivalent:



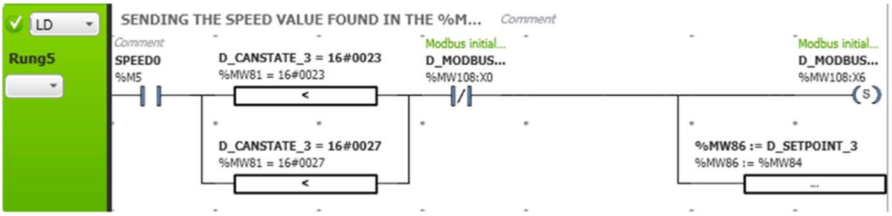
Setting to TRUE the bit 7 of `%MW108` stops the motor.

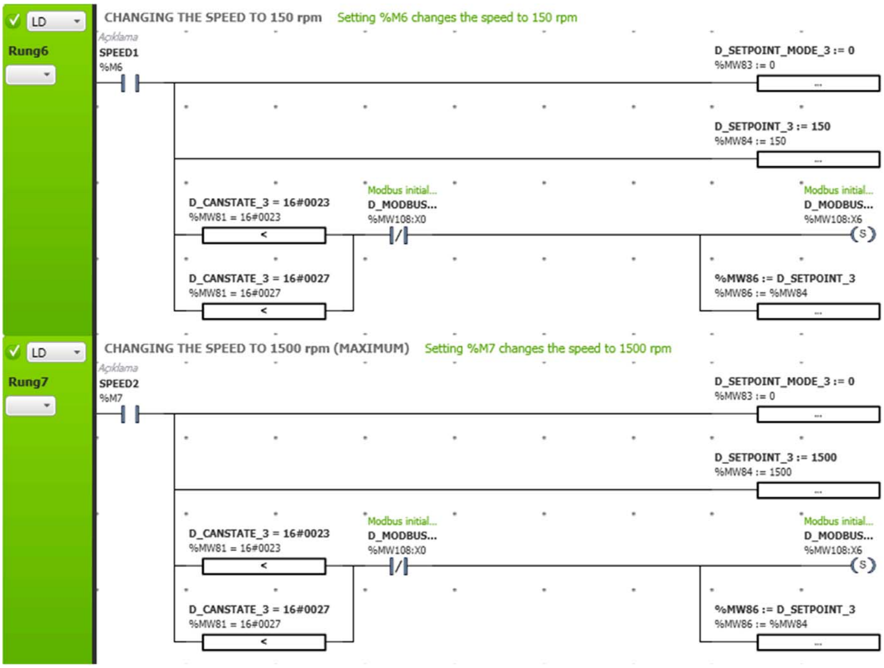
Clearing the detected errors so that the drive is reinitialized. `D_Clear_Errd` equivalent:



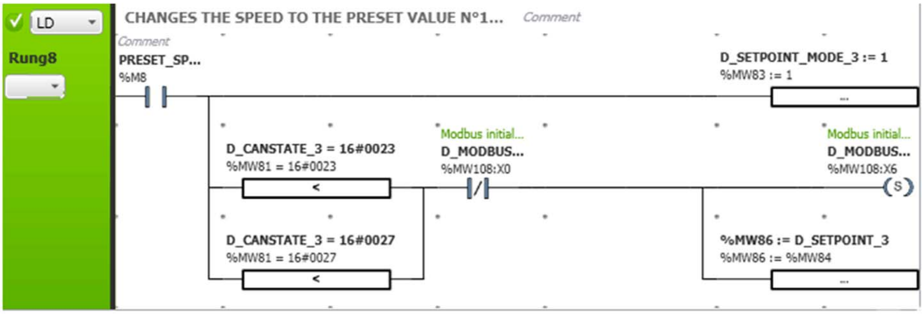
Setting to TRUE the bit 10 of `%MW108` clears the detected errors present in the ATV.

Modifying the speed. `D_Select_Speed` equivalent:





Modifying the speed to the preset value 1. `D_Select_Speed` equivalent:



Setting to TRUE the bit 6 of `%MW108` means that a new speed value needs to be sent, either as a preset speed or a modifiable speed.

The EcoStruxure Machine Expert - Basic project is functionally equivalent to the Twido project:

- Calls the subroutine `SR0` at each execution cycle to pass implicitly from one state to another.
- Turns the motor in the forward direction when `%M1` is set to TRUE.
- Turns the motor in the reverse direction when `%M2` is set to TRUE.
- Stops the motor when `%M3` is set to TRUE.
- Clears detected errors when `%M4` is set to TRUE. This reinitializes the drive so that the other commands can be re-sent.
- Sends the speed value found at `%MW84` when `%M5` is set to TRUE.
- Sets the speed to 150 rpm when `%M6` is set to TRUE.
- Sets the speed to 1500 rpm when `%M7` is set to TRUE.
- Modifies the speed to the preset value 1 when `%M8` is set to TRUE.

The subroutine `SR0` corresponds to the `D_Manager` macro in the TwidoSuite program. All other macros are converted to a rung that sets one bit to TRUE in the word `Modbus_init` (`%MW108`). This word is then used inside the subroutine to determine to which state the drive must go. The details on this subroutine can be found in Operating Principles of Drive Macros *(see page 29)*.

Modifying the Converted Project

Introduction

Although the conversion works for the drive macros as they appeared in the original Twido project, if you modify your configuration you need to modify certain parameter values in your application.

NOTE: Each rung, with the exception of the shared subroutine, is independent from the others. So if you modify one of them, the others are not affected. For more information on how to modify settings, refer to Operating Principles of Drive Macros *(see page 29)*.

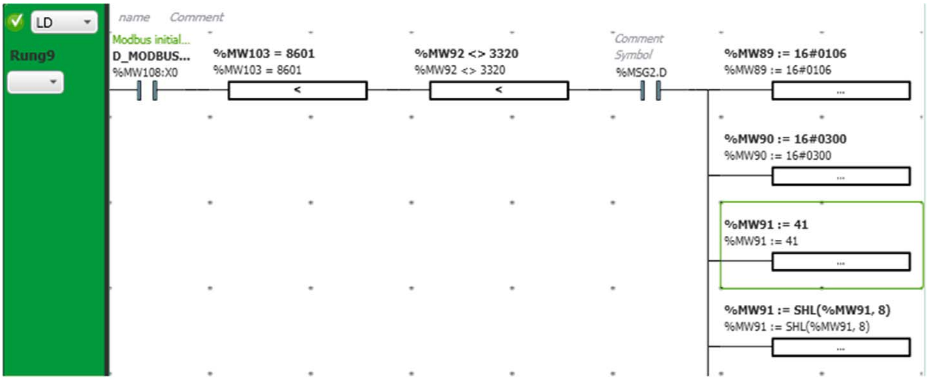
Modifying the Modbus Slave Address

If you change your drive, or need to assign a new Modbus slave address to it, you must change the slave address value in all `EXCH` instructions. `EXCH` instructions are used for Modbus communications. For more details, refer to *EcoStruxure Machine Expert - Basic Example Guide - Importing Twido COMM Macros (Sample_Twido_Macro_COMM_Conversion.smbe)*.

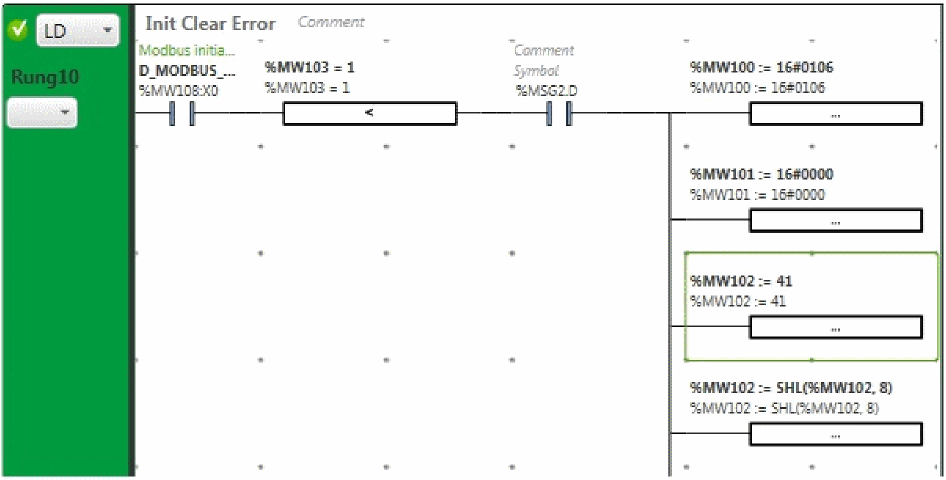
One way to modify the slave address is to search for the addresses `%MW91` and `%MW102` (macro start address `%MW80 + 11` and `%MW80 + 22`, respectively). These addresses are only used in the subroutine.

The following graphics show where the Modbus slave address is specified. After modifying this value in all locations, you will be able to communicate with the new Modbus slave address. The address modification needs to be done in rungs 9, 10, 21, and 30 of the subroutine.

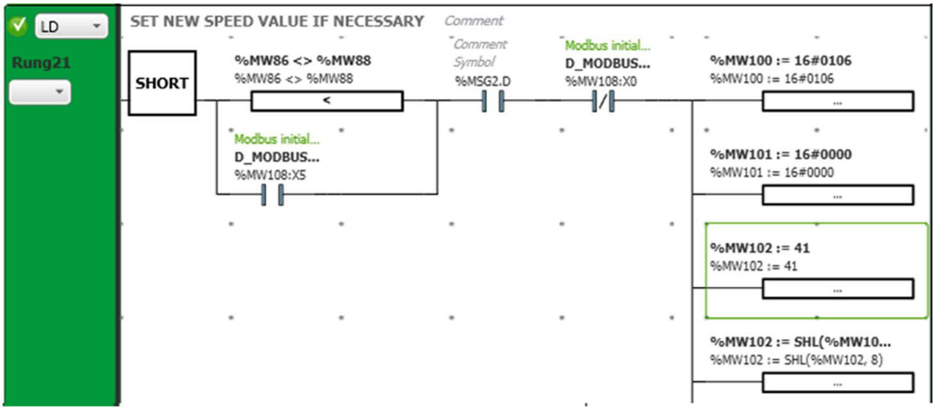
Slave address in Rung 9:



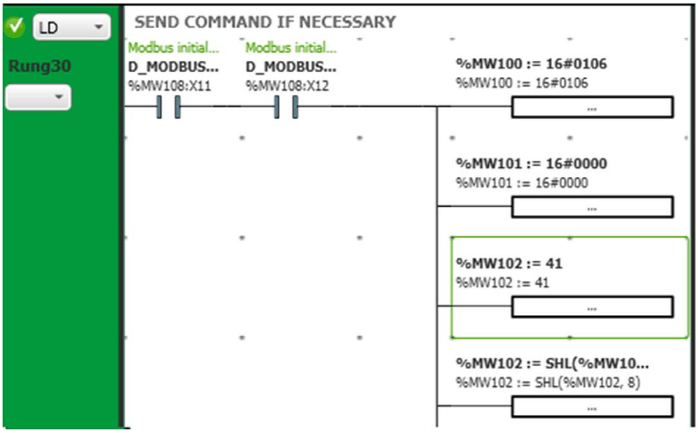
Slave address in Rung 10:



Slave address in Rung 21:



Slave address in Rung 30:



Replace the value 41 with the new number at these locations to modify the slave address.

Modifying Speed Values and Modes

Modifying the speed, turning direction, and so on, is the same as in Twido. In each rung, set a bit to TRUE in the memory word `%MW108` (29th word of the macro), depending on the type of macro.

Bit details of memory word `%MW108`:

Bit	Name/Use
0	Refer to Operating Principles of Drive Macros *(see page 29)*.
1	
2	
3	
4	
5	
6	Select Speed
7	Stop
8	Run Forward
9	Run Reverse
10	Clear Error

Bit	Name/Use
11	Refer to Operating Principles of Drive Macros *(see page 29)*.
12	
13	
14	
15	

The speed values, as in Twido, are modified in memory word `%MW84` (5th word of the macro) and the speed modes are modified in memory word `%MW83`.

To re-use the macros for other purposes in your application, you can copy and paste a rung from the first POU and modify the speed values, or specify different contact addresses.

Appendices

Appendix A
Appendices

Operating Principles of Drive Macros

Introduction

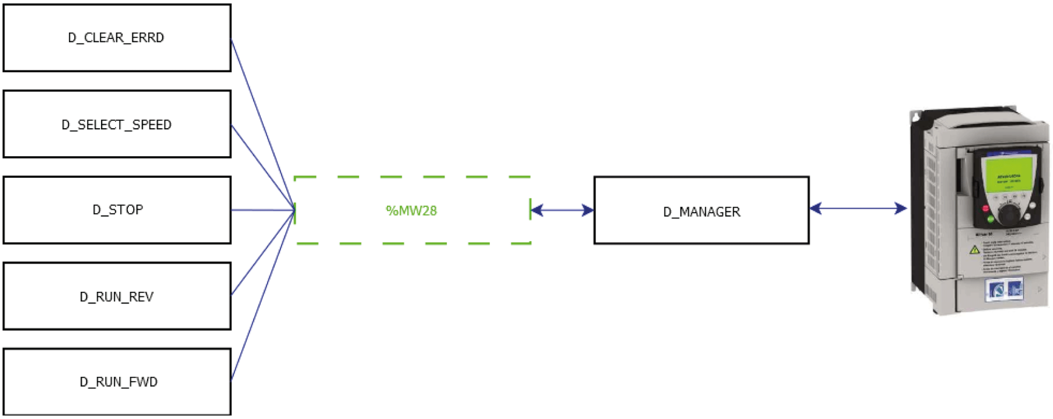
This section presents advanced adaptations beyond converting and using Twido drive macros in EcoStruxure Machine Expert - Basic.

Each TwidoSuite macro is converted into rungs in EcoStruxure Machine Expert - Basic. In this section, you will obtain a better understanding of the operating principles of the drive macros.

From this point onwards, the memory words are referenced by their relative position within the macro. This means that `%MW80` mentioned previously is now `%MW0`, `%MW91` is now `%MW11`, and so on. This representation is used to allow the same references to be used across multiple instances of macros.

Twido Drive Macros Interaction

In Twido drive macros, communication is done through the `D_Manager` macro (the subroutine), while the other macros modify a specific memory word (`%MW28`), which serves as a messenger. An illustration of interaction between drive macros is given below:



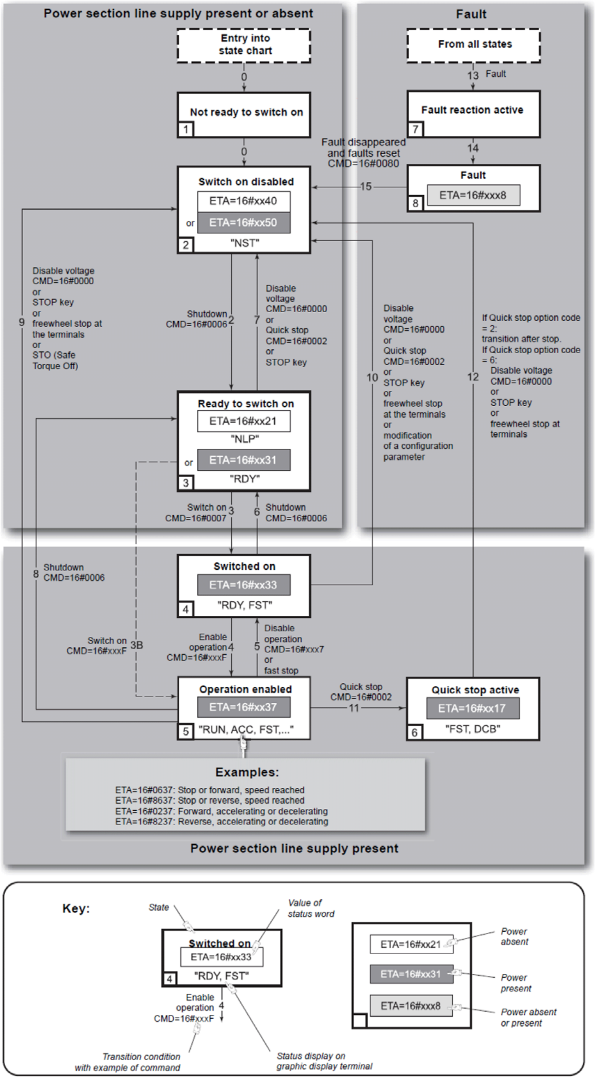
Additionally, the `D_Select_Speed` macro modifies the values in `Select_Mode` (`%MW3`) and `Select_Point` (`%MW4`), which are subsequently taken into account by the `D_Manager` macro.

Altivar Operating Principles

This document focuses on the role of the certain memory words of the ATV 61 and their values. It does not describe in detail the electrical operating principles, nor its communication with the motor.

Altivar drives have a common structure of memory words which can be shared using the same principles.

An Altivar drive has different operating states. The following state chart shows the different states of the Altivar drive and the conditions for passing from one state to another:



Power section line supply present or absent
Entry into state chart
0
Not ready to switch on
1
0
Switch on disabled
ETA=16#xx40
or
ETA=16#xx50
2
"NST"
Fault
From all states
13 Fault
Fault reaction active
7
14
Fault
ETA=16#xxx8
8
Fault disappeared and faults reset CMD=16#0080
15
9
Disable voltage CMD=16#0000 or STOP key or freewheel stop at the terminals or STO (Safe Torque Off)
Shutdown CMD=16#0006
2
7
Disable voltage CMD=16#0000 or Quick stop CMD=16#0002 or STOP key
10
Disable voltage CMD=16#0000 or Quick stop CMD=16#0002 or STOP key or freewheel stop at the terminals or modification of a configuration parameter
12
If Quick stop option code = 2: transition after stop. If Quick stop option code = 6: Disable voltage CMD=16#0000 or STOP key or freewheel stop at terminals
Ready to switch on
ETA=16#xx21
"NLP"
or
ETA=16#xx31
3
"RDY"
Switch on CMD=16#0007
3
6
Shutdown CMD=16#0006
8
Shutdown CMD=16#0006
Switched on
ETA=16#xx33
4
"RDY, FST"
Switch on CMD=16#xxxF
3B
Enable operation CMD=16#xxxF
4
5
Disable operation CMD=16#xxx7 or fast stop
Operation enabled
ETA=16#xx37
5
"RUN, ACC, FST,..."
Quick stop CMD=16#0002
11
Quick stop active
ETA=16#xx17
6
"FST, DCB"
Examples:
ETA=16#0637: Stop or forward, speed reached
ETA=16#8637: Stop or reverse, speed reached
ETA=16#0237: Forward, accelerating or decelerating
ETA=16#8237: Reverse, accelerating or decelerating
Power section line supply present
Key:
State
Value of status word
Switched on
ETA=16#xx33
4
"RDY, FST"
Enable operation CMD=16#xxxF
4
Transition condition with example of command
Status display on graphic display terminal
ETA=16#xx21
Power absent
ETA=16#xx31
Power present
ETA=16#xxx8
Power absent or present

Passing from one state to another is done using the command word (CMD). Each state has a mask used for reading its status (ETA).

A simple description of this machine is:

- When the motor is running normally, the ATV is in state **Operation enabled** (5).
  To switch to this state, the command word 000F hex must be sent when the ATV is in state 4.
- When the ATV is shut down (not braking), it is in state **Ready to switch on** (3).
  To switch to this state, the command word 0006 hex must be sent when the ATV is in either state 5 or state 2. The motor stops due its own friction.
- When the ATV is switched on, it is in state **Switched on** (4).
  To switch to this state, the command word 0007 hex must be sent when the ATV is in either state 3 or state 5.
- When in state 5, the command word 0002 hex activates the quick stop causing the ATV to go to the **Quick stop active** state (6). It stops the motor faster than a normal stop, but it requires the motor to be reinitialized and the drive must pass through states 3 and 4 to start again.
- From all states, a detected error forces the ATV to go to state **Fault** (8).
  The ATV must then pass to state 2 with the command word 0080 hex.

By using the `EXCH` instruction or the `Read Var` and `Write Var` function blocks, it is possible to manage an ATV, as shown in the template `xSample_ATV_Modbus_SL_M221`.

Refer to Altivar Parameters *(see page 32)* where are described the memory words used by the `D_Manager` macro for reading and writing.

It is also necessary to know the role of memory words used within the macro. A drive macro uses 30 memory words (`%MW0` to `%MW29`). Seven memory words (`D_STATE`, `D_CANSTATE`, `D_ERROR`, `D_SETPOINT_MODE`, `D_SETPOINT`, `D_MODBUS_INIT_PHASE`) are named explicitly and described in the TwidoSuite online help. The others are reserved for the `D_Manager` macro and used internally by this macro. The memory word `%MW28` is used as a messenger between the other macros and also between the rungs within the `D_Manager` macro itself.

In summary, a drive macro works with:

- The parameters used by the ATV *(see page 32)*
- The 30 memory words used by the macro *(see page 33)*
- The 16 bits of the memory word `%MW28` *(see page 38)*

Altivar Parameters

The parameters of the ATV 61 are detailed in the document ATV61_communication_parameters_EN_V5.8_IE29 *(see page 10)*. However, this table summarizes these parameters and also which ones the `D_Manager` macro reads or writes to:

Logic Address	Description
8601	This is the Command word (CMD), used to pass from one state to another. The values it takes are the following: ● 0000 hex to disable voltage ● 0002 hex to quick stop ● 0006 hex to stop ● 0007 hex to switch on ● 000F hex to enable operation ● 0080 hex to clear detected errors
8602	This is the commanded speed value. The value written in `D_Setpoint` (`%MW4`) is written to this logic address. The value can be converted to a frequency using the following formula: *w* = 2 x *f* x 60 / *n* Where *w* is the speed in rpm, *f* is the frequency and *n* is the number of poles. With a 4-pole motor, the formula becomes: *f* = *w* / 30 By reading the address 8604 (output velocity), you can verify whether the drive has reached the speed value you requested.
8603	This is the aforementioned ETA value, that is, the present state of the ATV. It is used with a mask and the masked value is stored somewhere else. Depending on the state of the ATV, the `D_Manager` macro sends the appropriate CMD words. Refer to ATV61_communication_parameters_EN_V5.8_IE29 *(see page 10)* to understand the meaning of the value found in this word.
8606	Contains the ERRD value. It shows the detected error and it is cleared by sending the command 0080 hex.
8413	This logic address selects the type of the first channel. In the template, it contains the value 164, which means Modbus.
11401, 11402 and 11403	These are the preset speed values. They are initialized with 173, 174, and 175 respectively. If no preset values are set, using modifiable preset speeds is not possible.
3320	This is the part number of the ATV. Its value can be used to detect whether it is an ATV 31, ATV 61, or ATV 71.

Macro Memory Words

This table describes the memory words used in the macro:

Memory Word Address	Name/Use
%MW0	State
%MW1	Can State
%MW2	Error
%MW3	Set Point Mode
%MW4	Set Point
%MW5	Command
%MW6	Set Point Backup
%MW7	Command Backup
%MW8	Set Point Backup
%MW9	First Exchange Table Reading 4 Words
%MW10	
%MW11	
%MW12	
%MW13	
%MW14	
%MW15	
%MW16	
%MW17	
%MW18	
%MW19	
%MW20	Second Exchange Table Writing 1 Word
%MW21	
%MW22	
%MW23	
%MW24	
%MW25	
%MW26	
%MW27	
%MW28	Messenger
%MW29	–

For more information, refer to the reference documents on Modbus communication, and the EXCH instructions *(see EcoStruxure Machine Expert - Basic, Generic Functions Library Guide)*.

First Exchange Table

This table is used to read 4 words from the drive. It uses 11 words and has the following format:

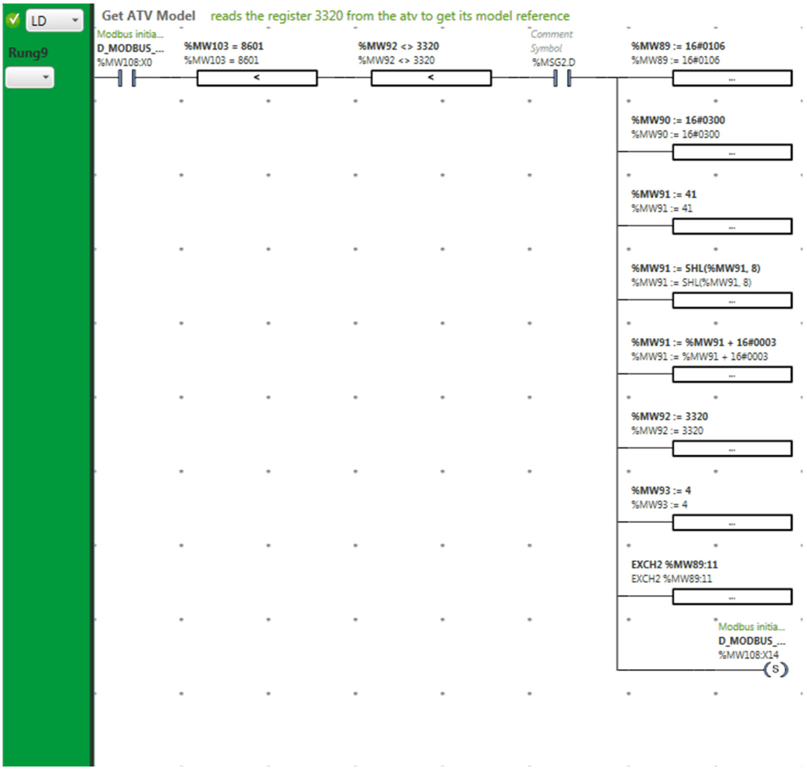
	Memory Word Address	Memory Word Name	Value (if possible)	
Control Table	%MW9	Control Table 1 : Header	16#0106	Filled by the Master Manually
	%MW10	Control Table 2 : Offset	16#0300	
Modbus Request	%MW11	*1st Byte* *2nd Byte* Request 1: Slave Address Function Code	*1st Byte* *2nd Byte* Address 03	
	%MW12	Request 2 : Starting Address		
	%MW13	Request 3 : Quantity of Words	4	
Modbus Response	%MW14	*1st Byte* *2nd Byte* Response 1: Slave Address Function Code	*1st Byte* *2nd Byte* Address 03	Filled by the Slave (Automatic)
	%MW15	Response 2 : Quantity of Words	4	
	%MW16	Response 3 : Value 1		
	%MW17	Response 4 : Value 2		
	%MW18	Response 5 : Value 3		
	%MW19	Response 6 : Value 4		

An exchange table is called by using the `EXCH` instruction within an operation block:

```
EXCH1 %MW89:11
```

where 11 is the total number of words in the exchange table.

You can see the configuration of this table in rung 9 and its use in rungs 9, 13, and 26.



Get ATV Model reads the register 3320 from the atv to get its model reference

```
Modbus initia...
D_MODBUS_...
%MW108:X0

%MW103 = 8601
%MW103 = 8601
<

%MW92 <> 3320
%MW92 <> 3320
<

Comment
Symbol
%MSG2.D

%MW89 := 16#0106
%MW89 := 16#0106
...

%MW90 := 16#0300
%MW90 := 16#0300
...

%MW91 := 41
%MW91 := 41
...

%MW91 := SHL(%MW91, 8)
%MW91 := SHL(%MW91, 8)
...

%MW91 := %MW91 + 16#0003
%MW91 := %MW91 + 16#0003
...

%MW92 := 3320
%MW92 := 3320
...

%MW93 := 4
%MW93 := 4
...

EXCH2 %MW89:11
EXCH2 %MW89:11
...

Modbus initia...
D_MODBUS_...
%MW108:X14
(S)
```

Second Exchange Table

This table is used to write one word to the drive, and is used in the macro.

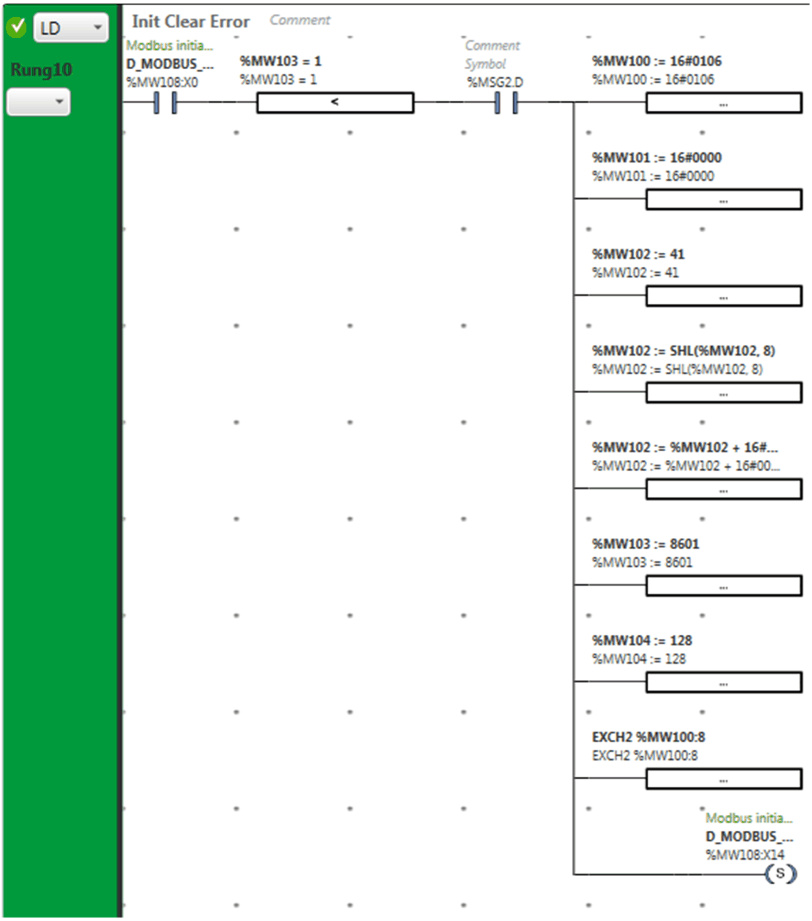
	Memory Word Address	Memory Word Name	Value (if possible)	
Control Table	%MW20	Control Table 1: Header	16#0106	Filled by the Master (Manually)
	%MW21	Control Table 2: Offset	16#0007	
Modbus Request	%MW22	*1st Byte* *2nd Byte* Request 1: Slave Address Function Code	*1st Byte* *2nd Byte* Address 06	
	%MW23	Request 2: Starting Address		
	%MW24	Request 3: Value		
Modbus Response	%MW25	*1st Byte* *2nd Byte* Reponse 1: Slave Address Function Code	*1st Byte* *2nd Byte* Address 06	Filled by the Slave (Automatic)
	%MW26	Response 2: Echo Starting Address		
	%MW27	Response 3: Echo Value		

An exchange table is called by using the `EXCH` instruction within an operation block:

```
EXCH1 %MW100:8
```

where 8 corresponds to the total number of words in the exchange table.

The configuration is in rungs 10, 21, and 30 and it is used in rungs 3, 4, 5, 7, 8, 10, 21, and 30.



Modbus initia...
D_MODBUS_...
%MW108:X0

%MW103 = 1
%MW103 = 1

<

Comment
Symbol
%MSG2.D

%MW100 := 16#0106
%MW100 := 16#0106

%MW101 := 16#0000
%MW101 := 16#0000

%MW102 := 41
%MW102 := 41

%MW102 := SHL(%MW102, 8)
%MW102 := SHL(%MW102, 8)

%MW102 := %MW102 + 16#...
%MW102 := %MW102 + 16#00...

%MW103 := 8601
%MW103 := 8601

%MW104 := 128
%MW104 := 128

EXCH2 %MW100:8
EXCH2 %MW100:8

Modbus initia...
D_MODBUS_...
%MW108:X14

Memory Word %MW28

The following table shows the purpose of each bit of the memory word %MW28:

Bit	Name
0	Error
1	%S6 of logic controller
2	–
3	ATV reference (0 for ATV61/71)
4	–
5	Order to send Speed Value
6	Select Speed
7	Stop
8	Run Forward
9	Run Reverse
10	Clear Error
11	Order to send a command
12	–
13	–
14	Frame is sent
15	–