ENGINEERING
TOMORROW

Programming Guide

Function Blocks with Schneider Electric Unity Pro

drives.danfoss.com | VLT®

Contents

1 Introduction

1.1 Disclaimer

The software is provided "as is", without warranty of any kind, expressed or implied, including, but not limited to, the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or any legal entity part of Danfoss group be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the software, or the use, or other dealings in the software.

1.2 Purpose of the User Guide

The function blocks show examples on how it is possible to integrate Danfoss VLT® drives in a Schneider Electric Unity Pro V13 system. The function blocks are not protected and can be altered to serve the specific needs for the application. Danfoss takes no responsibility to losses due to code faults in these function blocks or wrong use.

This manual provides:

- Step-by-step approach on how to integrate Danfoss VLT® drives into a Schneider Electric Unity Pro V13 system.
- Procedure on using the library to communicate with Danfoss VLT® drives in Schneider Electric Unity Pro V13 system, including examples.

Function blocks for Schneider Electric Unity Pro supporting Danfoss VLT® drives.

Danfoss VLT® drives with the following options:

- VLT® Modbus TCP MCA 122

The manual is intended for use by qualified personnel.

1.3 Abbreviations

Abbreviation	Description
FB	Function block
RPM	Revolution per minute
STW	Status word
CTW	Control word
Amp	Ampere
FC	Frequency converter
DDT	Derived data types
PDO	Process data object
SDO	Service data object
DUT	Device under test
LIB	Library
MBTCP	Modbus TCP

1.4 What are Function Blocks?

Function blocks are predefined programs or functions contained within a single program element that can be used in the PLC program.

1.4.1 Advantages of Using Function Blocks

- Basic skeleton:
  - FB provides the basic infrastructure towards the user.
  - Frees up time to focus on complex and application-specific implementation of the external device.
  - Reuse of an FB several times in a program without rewriting the FB.
  - Easy to use - knowledge of the internal operation of the drive or complex algorithms is not required.
- Pretested function:
  - The FB is pretested for working and functionality.
- Extensibility:
  - FBs can be extended in future by Danfoss. It is possible to incorporate the FBs with minimal modification in the existing program.

1.5 Overview of the Danfoss Library (VLT_MBTCP_LIB_V1_00)

The *VLT_MBTCP_LIB_V1_00* library is a collection of predefined function blocks provided by Danfoss. Use these blocks as an aid to simplify programs, containing standard functionality for programming Modbus TCP, Schneider Electric Unity Pro systems, and Danfoss drives.

The library contains the following FBs:

- **Basic operation function block (VLT_MBTCP_FC_BASIC)**: Dedicated to handling the basic operation of the drive and connected motor operations.
- **Parameter access function block (VLT_MBTCP_FC_PARAM_ACCESS)**: Dedicated to parameter read/write through an acyclic channel.
- **Diagnostics function block (VLT_MBTCP_FC_DIAGNOSTICS)**: Dedicated to read the diagnostic information.
- **Flexible control function block (VLT_MBTCP_FC_FLEXIBLE_CTRL)**: Dedicated to handle flexible control operation of drive and connected motor operations.

The function blocks do not support the following features:

- DS402 CANopen Profile.
- PROFIdrive profile.

1.6 Basic Operation Function Block (VLT_MBTCP_FC_BASIC)

The function block provides the following functionalities:

- **Control and monitoring**: monitor the drive and control the command or setpoint from the controller to/from the drive.
- **Reverse**: forward or reverse the direction of the motor.
- **Speed regulation**: allows the speed reference of the drive.
- **Failure management**: the *FAULT* output pin is set to *TRUE* if there is a drive fault. This drive fault must be reset by the input pin *RESET* to close the fault. The fault only disappears if the actual root cause of the fault has disappeared.

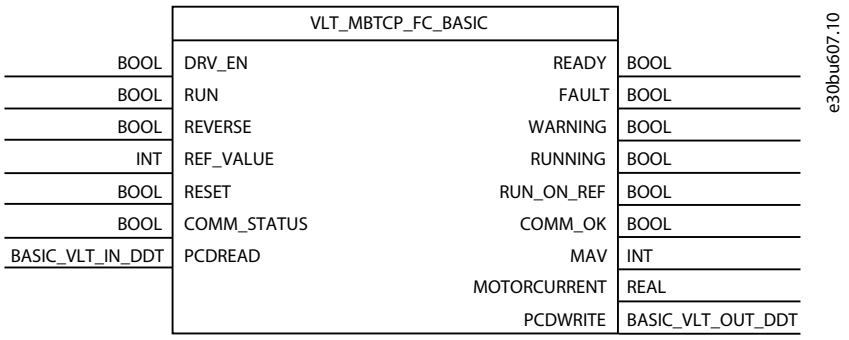


Illustration 1: Basic Operation Function Block Layout

Table 1: Input Parameter

Parameter	Type	Description	
DRV_EN	BOOL	0=the drive is not ready, 1=enabling the drive to ready.	
RUN	BOOL	0= the drive is in stop mode, 1=start command to the drive.	
REVERSE	BOOL	0=the drive runs in forward direction, 1=the drive runs in reverse directon.	
REF_VALUE	INT	The reference value requests the drive to run at reference speed and only accepts positive values. The range value 0–10000 equals to 0.00–100.00%. Enter the reference value without decimal. For example: To run the drive at 56.75%, enter *REF_VALUE* as 5675 to achieve the motor speed.	
RESET	BOOL	1=resets the device failure and resets the *FAULT* output to 0.	
COMM_STATUS	BOOL	Connect the *Healthy_Bit* variable of the slave device, and hold the current Modbus TCP state and link status of the Modbus TCP slave device.	
PCDREAD	BASIC_VLT_IN_DDT	Process data sent from the drive contains information about the current state of the drive.	
	Parameter	**Type**	**Description**
	Free0	**Array [0...7] of byte**	Array of PCDREAD
	Free0 [0]	BYTE	Status word low byte.
	Free0 [1]	BYTE	Status word high byte.
	Free0 [2]	BYTE	MAV low byte.
	Free0 [3]	BYTE	MAV high byte.
	Free0 [4]	BYTE	Motor current low byte 2.
	Free0 [5]	BYTE	Motor current high byte 3.
	Free0 [6]	BYTE	Motor current low byte 0.
	Free0 [7]	BYTE	Motor current high byte 1.

Table 2: Output Parameter

Parameter	Type	Description
READY	BOOL	1=the drive is ready for operation.

Parameter	Type	Description	
FAULT	BOOL	1=a detected failure in the control block. To reset the *FAULT* output pin, activate the *RESET* input pin.	
WARNING	BOOL	0=no warning, 1=the drive is in warning state.	
RUNNING	BOOL	1=the drive is running and has an output frequency (MAV>0).	
RUN_ON_REF	BOOL	0=the actual motor speed reaches the preset speed reference.	
COMM_OK	BOOL	1=the communication between the device and PLC is OK.	
MAV	INT	Main actual value in %, expressed in integer value 0–10000. For example: the MAV value is 9949 means that the drive is running at 99.49%.	
MOTORCUR-RENT	REAL	Motor current in Amps.	
PCDWRITE	BASIC_VLT_OUT_DDT	Process data sent from the PLC to the drive.	
	Parameter	**Type**	**Description**
	Free0	**Array [0...3] of byte**	**Array of PCDWRITE**
	Free0 [0]	BYTE	Control word low byte.
	Free0 [1]	BYTE	Control word high byte.
	Free0 [2]	BYTE	Reference low byte.
	Free0 [3]	BYTE	Reference high byte.

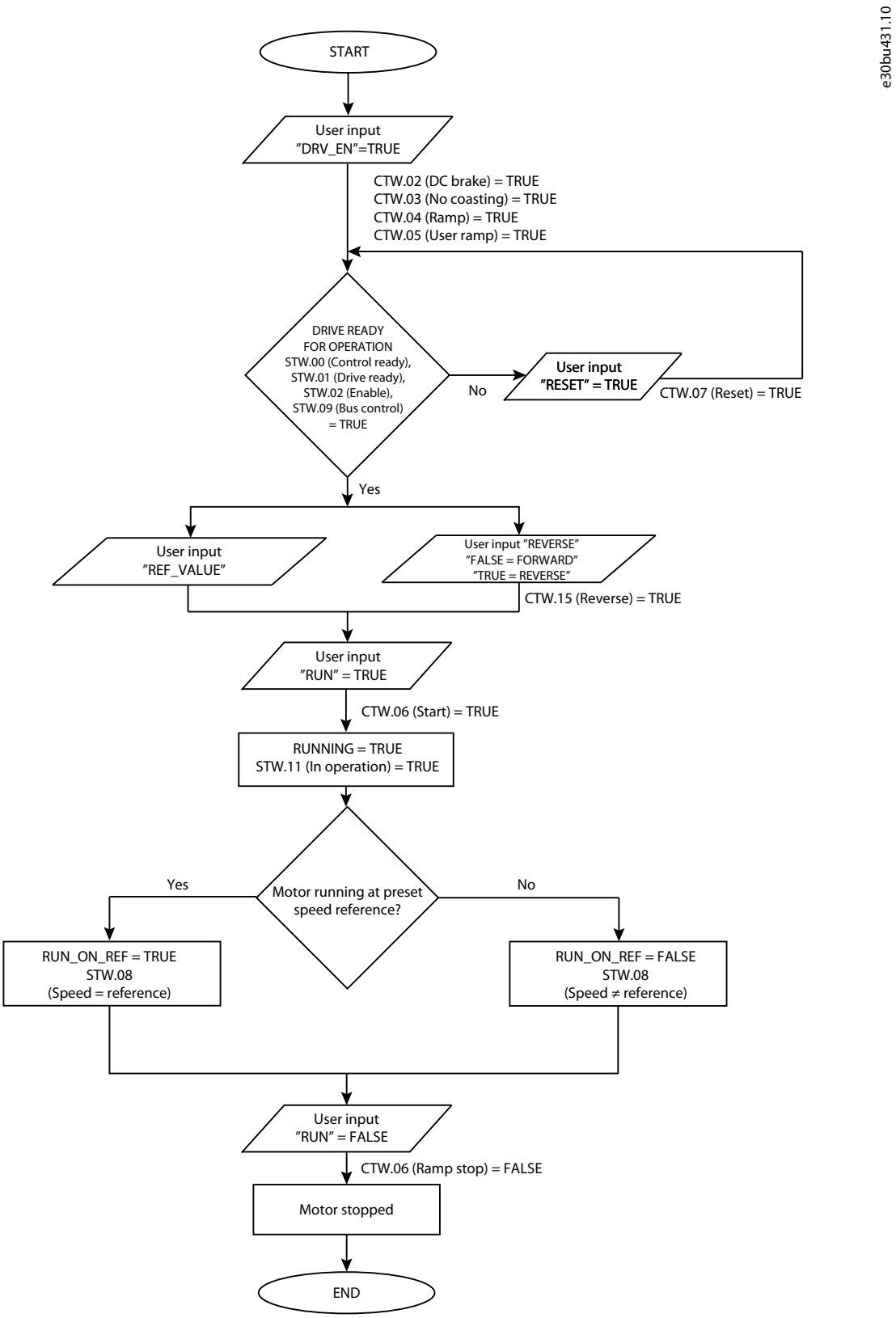


Illustration 2: Flow Chart for Basic Operation Of Drive Control

1.7 Parameter Access Function Block (VLT_MBTCP_FC_PARAM_ACCESS)

The function block provides functionality to read and write VLT® FC series parameters access through SDO service.

- Carry out a read or write operation (only one operation at a time).
- Read string type parameters.
- Write string type parameters.
- Read non-string type parameters.
- Write non-string type parameters.
- Show valid abort code on invalid read-write operations.
- Configurable timeout.

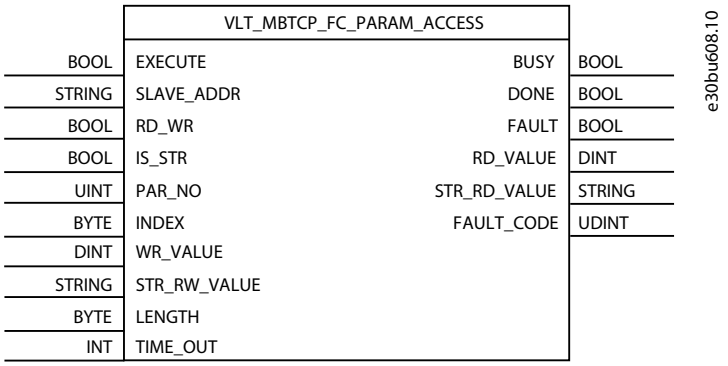


Illustration 3: Parameter Access Function Block Layout

The following elementary function blocks are embedded inside *VLT_MBTCP_FC_PARAM_ACCESS* from Schneider Electric Unity Pro *Communication* Libraries.

- READ_VAR: a function block which allows to read the data from Modbus TCP slave through explicit message.
- WRITE_VAR: a function block which allows to write the data on parameter of Modbus TCP slave through explicit message download.

Table 3: Input Parameter

Parameter	Type	Description
EXECUTE	BOOL	1=the rising edge of this signal starts the requested operation, 0=triggers an ACK of the end-of-operation notification and the client.
SLAVE_ADDR	STRING	Modbus TCP address.
RD_WR	BOOL	0= parameter read operation, 1=parameter write operation.
IS_STR	BOOL	0=to perform the operation based on *RD_WR* input for non-string data type (UDINT, DINT, INT, and so on) parameter. 1=to perform the operation based on *RD_WR* input for string data type parameter.
PAR_NO	UINT	Enter the drive parameter number in the integer format. (Example: *parameter 3-41 Ramp 1 Ramp Up Time*).
INDEX	BYTE	Enter the index number (Example: *parameter 310 [2] Preset Reference*) 3–10 is the parameter number and 2 is the index number.
WR_VALUE	DINT	Write integer data value to the drive.
STR_WR_VALUE	STRING	String parameters are written to the drive (VLT® drives supports maximum 25 characters only).
LENGTH	BYTE	Length of data can be read or write.

Parameter	Type	Description
TIME_OUT	INT	Timeout determines the maximum waiting time for the response (Timeout is a multiple of 100 ms which means if enter 30, the timeout is 30*100 ms =3 s).

Table 4: Output Parameter

Parameter	Type	Description
BUSY	BOOL	0=execution is not started, or execution is completed, 1=execution is processing.
DONE	BOOL	0=value exchange is not completed, 1=value exchange is completed.
FAULT	BOOL	0=value exchange is completed, 1=value exchange is unsuccessful.
RD_VALUE	DINT	Integer data value read from the drive.
STR_RD_VALUE	STRING	String parameter can be read. (VLT® drives support maximum 25 characters only).
FAULT_CODE	UINT	Indicates the fault code associated with the most recently executed command.

1.8 Diagnostics Function Block (VLT_MBTCP_FC_DIAGNOSTICS)

The function block provides functionality to read VLT® FC series diagnostics information through SDO service.

If the drive *parameter 8-07 Diagnosis Trigger* is set to:

- **Disable**: do not send extended diagnosis/emergency data even if they appear in the drive.
- **Trigger on alarms**: send extended diagnosis/emergency data when 1 or more alarms appear in *parameter 16-90 Alarm Word*.
- **Trigger alarm/warn.**: send extended diagnosis/emergency data if 1 or more alarms or warnings appear in *parameter 16-90 Alarm Word* and *parameter 16-92 Warning Word*.
- EXEMPTION! Warning word 3 and alarm word 3 are not reflected on fieldbus.

The function block supports the following functionalities:

- Show alarm word in 32-bit format (*parameter 16-90 Alarm Word*).
- Show warning word in 32-bit format (*parameter 16-92 Warning Word*).

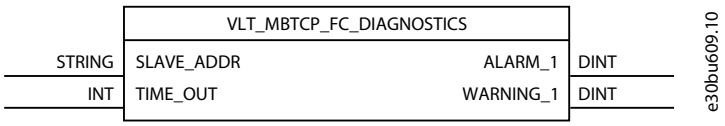


Illustration 4: Diagnostics Function Block Layout

The following elementary function blocks are embedded inside *VLT_MBTCP_FC_DIAGNOSTICS* function block from Schneider Electric Unity Pro *Communication* Libraries.

- READ_VAR: a function block which allows to read the diagnostics information (alarm word & warning word) from Modbus TCP slave through explicit message access.

Table 5: Input Parameter

Parameter	Type	Description
SLAVE_ADDR	STRING	Modbus TCP address.

Parameter	Type	Description
TIME_OUT	INT	Timeout determines the maximum waiting time for the response (Timeout is a multiple of 100 ms which means if enter 30, the timeout is 30*100 ms =3 s).

Table 6: Output Parameter

Parameter	Type	Description
ALARM_1	DINT	The alarms read from the drive *parameter 16-90 Alarm Word*.
WARNING_1	DINT	The warnings read from the drive *parameter 16-92 Warning Word*.

1.9 Flexible Control Function Block (VLT_MBTCP_FC_FLEXIBLE_CTRL)

This function block provides the functionality to read and write cyclic process data (*parameter 12-21 Process Data Config Write* and *parameter 12-22 Process Data Config Read*).

The function block supports the following functionalities:

- Read process data from the drive in user accessible format. That is, PCD Read [0] (STW expressed in data bits), PCD Read [1] (main actual value), and PCD Read [2] to PCD Read [9] depends on the drive configuration.
- Write process data to the drive in user accessible format. That is, PCD Write [0] (CTW expressed in data bits), PCD Write [1] (reference value), and PCD Write [2] to PCD Write [9] depends on the drive configuration.

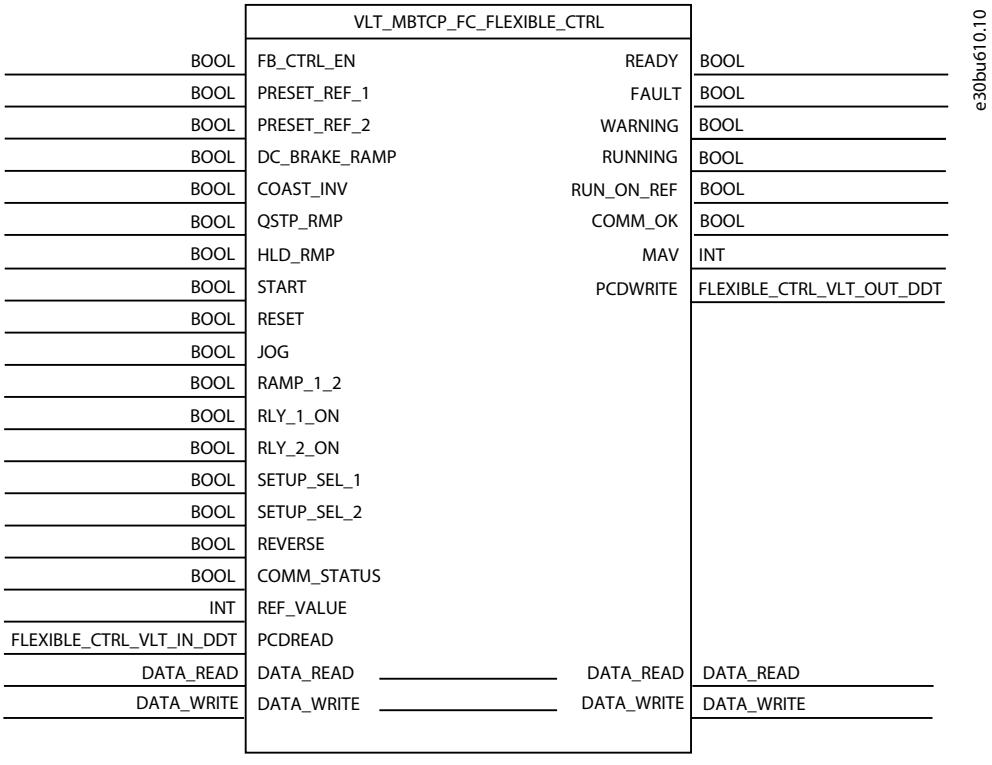


Illustration 5: Flexible Control Function Block Layout

Table 7: Input Parameter

Parameter	Type	Description
FB_CTRL_EN	BOOL	1=enabling control from function block. CTW.10=*TRUE*.
PRESET_REF_1	BOOL	Bit 0, 0=reference value, 1=selection lsb.
PRESET_REF_2	BOOL	Bit 1, 0=reference value, 1=selection msb.
DC_BRAKE_RAMP	BOOL	Bit 2, 0=DC brake, 1=ramp.
COAST_INV	BOOL	Bit 3, 0= coasting, 1=no coasting.
QSTP_RMP	BOOL	Bit 4, 0=quick stop, 1=ramp.
HLD_RMP	BOOL	Bit 5, 0=hold output frequency, 1=use ramp.
START	BOOL	Bit 6, 0=ramp stop, 1=start.
RESET	BOOL	Bit 7, 0=no function, 1=reset.
JOG	BOOL	Bit 8, 0= no function, 1=jog.
RAMP_1_2	BOOL	Bit 9, 0= ramp 1, 1=ramp 2.
RLY_1_ON	BOOL	Bit 11, 0= relay 01 deactived, 1=relay 01 active.
RLY_2_ON	BOOL	Bit 12, 0= relay 02 deactived, 1=relay 02 active.
SETUP_SEL_1	BOOL	Bit 13, 0=parameter set-up, 1=selection lsb. *Multi Setup* in *parameter 0-10 Active Set-up* must be selected.
SETUP_SEL_2	BOOL	Bit 14, 0=parameter set-up, 1=selection msb. *Multi Setup* in *parameter 0-10 Active Set-up* must be selected.
REVERSE	BOOL	Bit 15, 0=no fuction, 1=reverse.
COMM_STATUS	BOOL	Connect the *Healthy_Bit* variable of the slave device, hold the current Modbus TCP state, and link status of the Modbus TCP slave device.
REF_VALUE	INT	Main reference value (speed) in raw -16384 to + 16384.

Parameter	Type	Description	
PCDREAD	FLEXIBLE_CTRL_VLT_IN_DDT	Input data read structure. Holds the data structure obtained from the process data (PCDREAD) and expressed in user accessible data format.	
	Parameter	**Type**	**Description**
	Free0	**Array [0...19] of byte**	**Array of PCDREAD**
	Free0 [0]	BYTE	Status word low byte.
	Free0 [1]	BYTE	Status word high byte.
	Free0 [2]	BYTE	MAV low byte.
	Free0 [3]	BYTE	MAV high byte.
	Free0 [4]	BYTE	*Parameter 12-22 [2] low byte.*
	Free0 [5]	BYTE	*Parameter 12-22 [2] high byte.*
	Free0 [6]	BYTE	*Parameter 12-22 [3] low byte.*
	Free0 [7]	BYTE	*Parameter 12-22 [3] high byte.*
	Free0 [8]	BYTE	*Parameter 12-22 [4] low byte.*
	Free0 [9]	BYTE	*Parameter 12-22 [4] high byte.*
	Free0 [10]	BYTE	*Parameter 12-22 [5] low byte.*
	Free0 [11]	BYTE	*Parameter 12-22 [5] high byte.*
	Free0 [12]	BYTE	*Parameter 12-22 [6] low byte.*
	Free0 [13]	BYTE	*Parameter 12-22 [6] high byte.*
	Free0 [14]	BYTE	*Parameter 12-22 [7] low byte.*
	Free0 [15]	BYTE	*Parameter 12-22 [7] high byte.*
	Free0 [16]	BYTE	*Parameter 12-22 [8] low byte.*
	Free0 [17]	BYTE	*Parameter 12-22 [8] high byte.*
	Free0 [18]	BYTE	*Parameter 12-22 [9] low byte.*
	Free0 [19]	BYTE	*Parameter 12-22 [9] high byte.*

Table 8: Output Parameter

Parameter	Type	Description
READY	BOOL	1=the drive is ready for operation. Status word STW.01 gives information about the drive ready status.
FAULT	BOOL	1=a detected failure in the control block. To reset the *ALARM* output pin, activate the *RESET* input pin. Drive failure (STW.03 drive trips, STW.04 shows error but not tripped, STW.06 drive is tripped and locked).
WARNING	BOOL	1=warning has been activated for the drive. Warning is high when STW.07 is high.
RUNNING	BOOL	1=the drive is running and has an output frequency (MAV>0). Status word STW. 11 gives information about the motor running status.
RUN_ON_REF	BOOL	1=the drive is running and has an output frequency (speed=reference). Status word STW.08 gives information about the motor running status.

Parameter	Type	Description	
COMM_OK	BOOL	1=the communication between the device and PLC is OK.	
MAV	INT	Measured actual value (speed) in raw value (-16384) to (+16384).	
PCDWRITE	FLEXIBLE_CTRL_VLT_OUT_DDT	Input data read structure. Holds the data structure obtained from the process data (PCDREAD) and expressed in user accessible data format.	
	Parameter	**Type**	**Description**
	Free0	**Array [0...19] of byte**	**Array of PCDWRITE**
	Free0 [0]	BYTE	Control word low byte.
	Free0 [1]	BYTE	Control word high byte.
	Free0 [2]	BYTE	Reference low byte.
	Free0 [3]	BYTE	Reference high byte.
	Free0 [4]	BYTE	*Parameter 12-21 [2] low byte.*
	Free0 [5]	BYTE	*Parameter 12-21 [2] high byte.*
	Free0 [6]	BYTE	*Parameter 12-21 [3] low byte.*
	Free0 [7]	BYTE	*Parameter 12-21 [3] high byte.*
	Free0 [8]	BYTE	*Parameter 12-21 [4] low byte.*
	Free0 [9]	BYTE	*Parameter 12-21 [4] high byte.*
	Free0 [10]	BYTE	*Parameter 12-21 [5] low byte.*
	Free0 [11]	BYTE	*Parameter 12-21 [5] high byte.*
	Free0 [12]	BYTE	*Parameter 12-21 [6] low byte.*
	Free0 [13]	BYTE	*Parameter 12-21 [6] high byte.*
	Free0 [14]	BYTE	*Parameter 12-21 [7] low byte.*
	Free0 [15]	BYTE	*Parameter 12-21 [7] high byte.*
	Free0 [16]	BYTE	*Parameter 12-21 [8] low byte.*
	Free0 [17]	BYTE	*Parameter 12-21 [8] high byte.*
	Free0 [18]	BYTE	*Parameter 12-21 [9] low byte.*
	Free0 [19]	BYTE	*Parameter 12-21 [9] high byte.*

Table 9: Input/output Parameter

Parameter	Type	Description	
DATA_READ	DATA_READ	Input data read structure. Holds the data structure obtained from the process data (PCDREAD) and expressed in user accessible data format.	
	Parameter	**Type**	**Description**
	STW	**Structure**	**Status word**
	BIT_00_CTRL_RDY	BOOL	0=control not ready, 1=control ready.
	BIT_01_DRV_RDY	BOOL	0=drive, 1=drive ready.
	BIT_02_COST_RDY	BOOL	0=coasting, 1=enable.
	BIT_03_TRIP	BOOL	0=no error, 1=trip.
	BIT_04_ERROR	BOOL	0=no error, 1=error (no trip).
	BIT_05_RESERVED	BOOL	0=reserved, 1=no action.
	BIT_06_TRIPLOCK	BOOL	0=no error, 1=trip lock.
	BIT_07_WARINING	BOOL	0=no warning, 1=warning.
	BIT_08_RUNNING_REF	BOOL	0=speed<>reference, 1=speed=reference.
	BIT_09_CTRL_SOURCE	BOOL	0=local operation, 1=bus control.
	BIT_10_F_LIMIT_OK	BOOL	0=out of frequency limit, 1=frequency limit ok.
	BIT_11_IN_OPR	BOOL	0=no operation, 1=in operation.
	BIT_12_IN_AUTOSTART	BOOL	0=drive OK, 1=stopped, auto start.
	BIT_13_VOLT_MAX	BOOL	0=voltage OK, 1=voltage exceeded.
	BIT_14_TORQ_MAX	BOOL	0=torque OK, 1=torque exceeded.
	BIT_15_TIME_EXD	BOOL	0=timer OK, 1=timer exceeded.
	MAV	INT	Measured actual value (speed) in raw value 0 to (+16384).
	PCD_02	INT	Depends on the drive configuration.
	PCD_03	INT	Depends on the drive configuration.
	PCD_04	INT	Depends on the drive configuration.
	PCD_05	INT	Depends on the drive configuration.
	PCD_06	INT	Depends on the drive configuration.
	PCD_07	INT	Depends on the drive configuration.
	PCD_08	INT	Depends on the drive configuration.
	PCD_09	INT	Depends on the drive configuration.

Parameter	Type	Description	
DATA_WRITE	DATA_WRITE	Holds the data structure in user accessible data format to send the process data (PCDWRITE) to the drive.	
	Parameter	**Type**	**Description**
	CTW	**Structure**	**Control word**
	BIT_00_PRESET_REF_1	BOOL	0=reference value, 1=selection lsb.
	BIT_01_PRESET_REF_2	BOOL	0=reference value, 1=selection msb.
	BIT_02_DC_BRAKE_RAMP	BOOL	0=DC brake, 1=ramp.
	BIT_03_COAST_INV	BOOL	0=coasting, 1=no coasting.
	BIT_04_QSTP_RMP	BOOL	0=quick stop, 1=ramp.
	BIT_05_HLD_RMP	BOOL	0=hold output frequency, 1=use ramp.
	BIT_06_START	BOOL	0=ramp stop, 1=start.
	BIT_07_RESET	BOOL	0=no function, 1=reset.
	BIT_08_JOG	BOOL	0=no function, 1=jog.
	BIT_09_RAMP_1_2	BOOL	0=ramp 1, 1=ramp 2.
	BIT_10_USE_CTRL_WORD	BOOL	0=local control, 1=fieldbus control.
	BIT_11_RLY_1_ON	BOOL	0=relay 01 deactivated, 1=relay 01 active.
	BIT_12_RLY_2_ON	BOOL	0=relay 02 deactivated, 1=relay 02 active.
	BIT_13_SETUP_SEL_1	BOOL	0=parameter set-up 1, 1=selection msb.
	BIT_14_SETUP_SEL_2	BOOL	0=parameter set-up 2, 1=selection msb.
	BIT_15_REVERSE	BOOL	0=no reverse, 1=reverse.
	REF_VALUE	INT	Main reference speed is a raw value (-16384) to (+16384).
	PCD_02	INT	Depends on the drive configuration.
	PCD_03	INT	Depends on the drive configuration.
	PCD_04	INT	Depends on the drive configuration.
	PCD_05	INT	Depends on the drive configuration.
	PCD_06	INT	Depends on the drive configuration.
	PCD_07	INT	Depends on the drive configuration.
	PCD_08	INT	Depends on the drive configuration.
	PCD_09	INT	Depends on the drive configuration.

2 Using FBs in Schneider Electric Unity Pro

2.1 Importing Danfoss Library into a Project

Procedure

1. Download the VLT_MBTCP_LIB_V1_00.zip file from Danfoss website.
2. Unzip the VLT_MBTCP_LIB_V1_00.zip file.

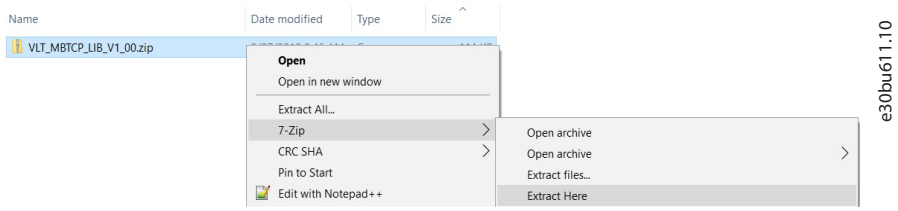


Illustration 6: Extract VLT_MBTCP_LIB_V1_00 Library

3. From the Microsoft Windows start menu, select *Types Library Update* in the Schneider Electric Folder.

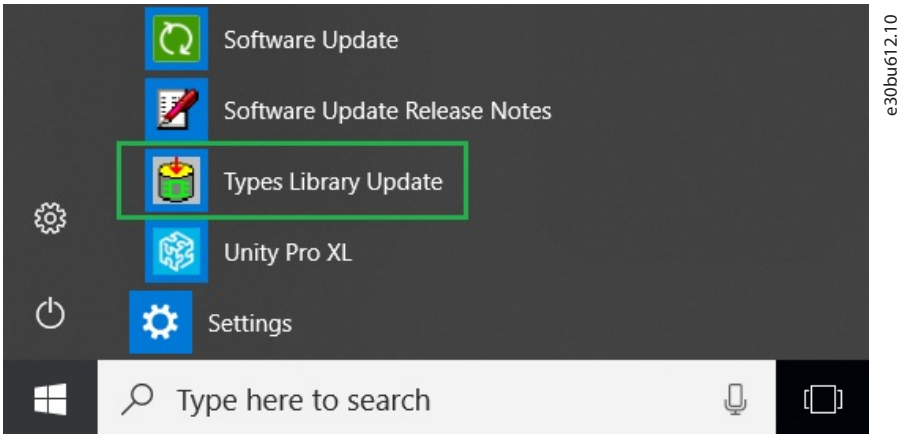


Illustration 7: Types Library Update

4. Select the highlighted button.

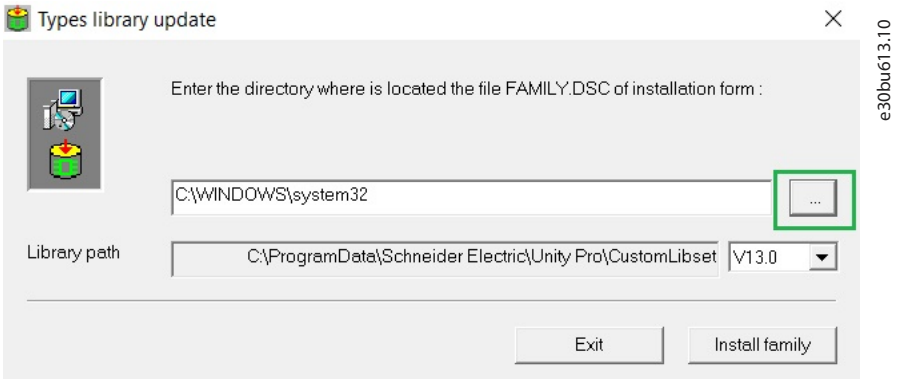


Illustration 8: Library Update

5. Navigate to the unzipped *VLT_MBTCP_LIB_V1_00* library folder location, select *FAMILY.DSC* file.

6. Press *Open*.

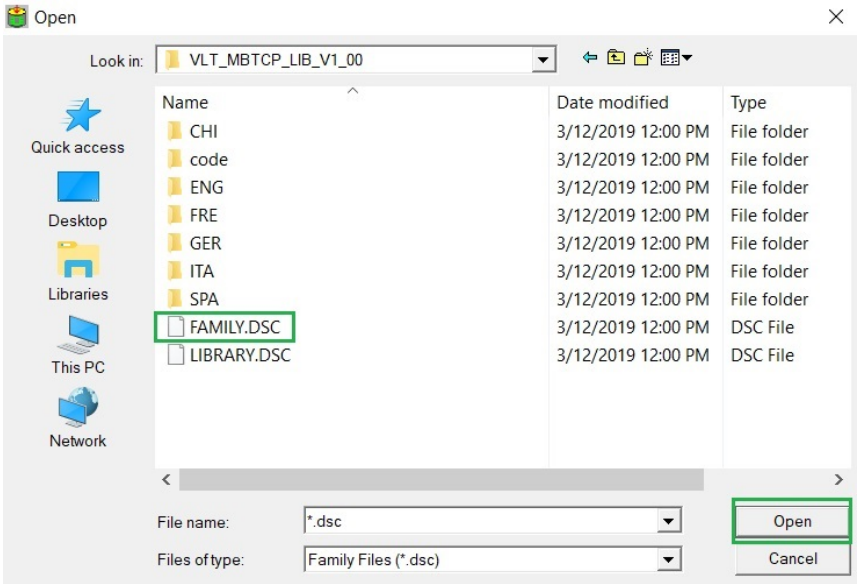


Illustration 9: Select Family.DCS File

7. Click *Install family* to install the *VLT_MBTCP_LIB_V1_00* library.

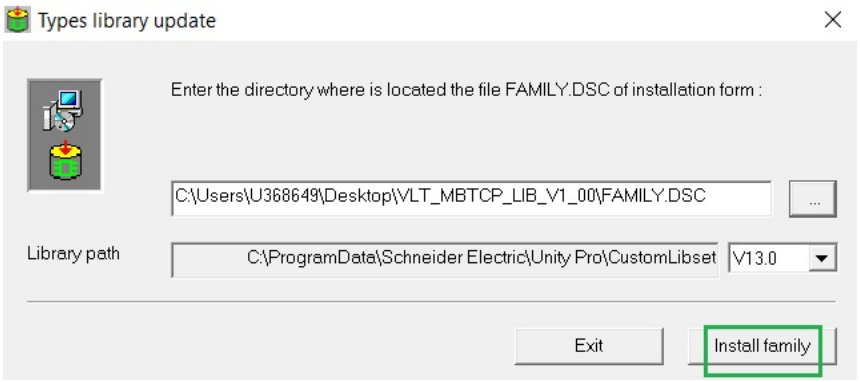


Illustration 10: Install Family

8. After successful installation, press *OK* in *The installation has succeeded* message.

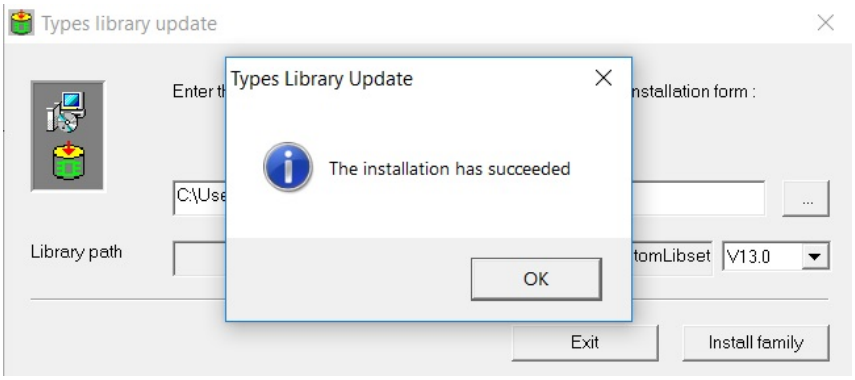


Illustration 11: Successful Installation

9. Open Unity Pro.

10. Select *Types Library Manager* from *Tools* menu bar.

11. Verify whether *VLT_MBTCP_LIB_V1_11* library is installed as shown in following illustration.

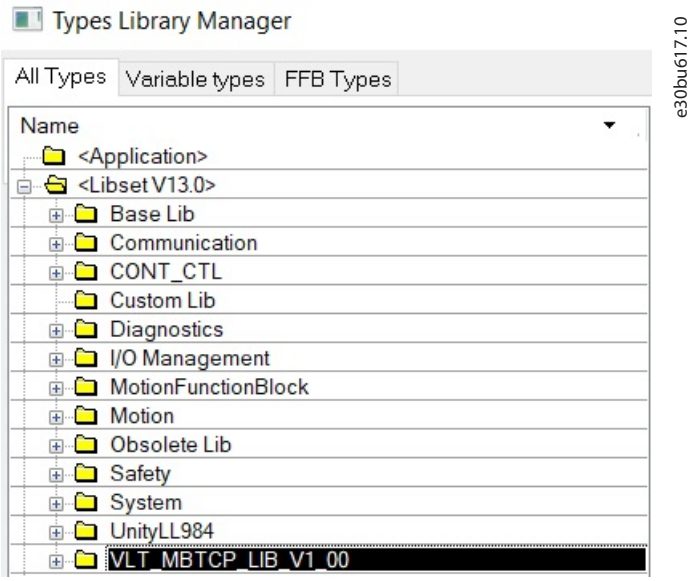


Illustration 12: Types Library Manager with VLT_MBTCP_LIB_V1_00 Library

2.2 Creating a Project

Procedure

1. Open Schneider Electric Unity Pro software.
2. From the menu bar, select *File⇒New*....

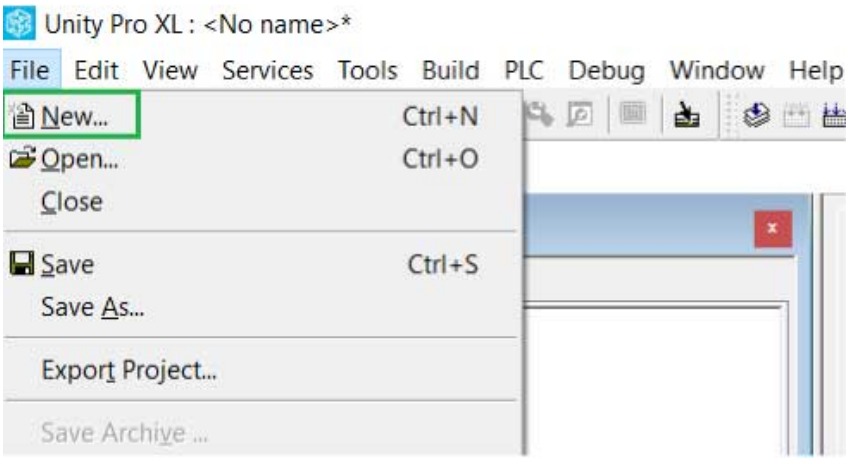


Illustration 13: Main Section Creation

→ *New Project* dialog box appears.

3. Select the used M340 Ethernet PLC, and click *OK*.

NOTICE

Configuration should be made according to the actual used PLC.

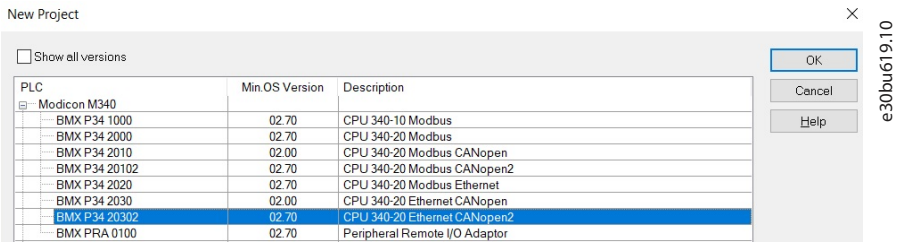
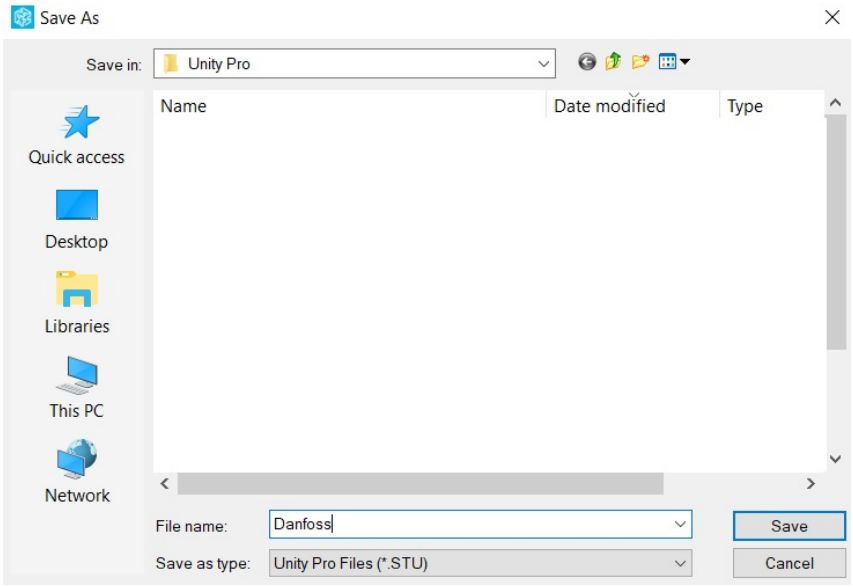


Illustration 14: CPU Selection

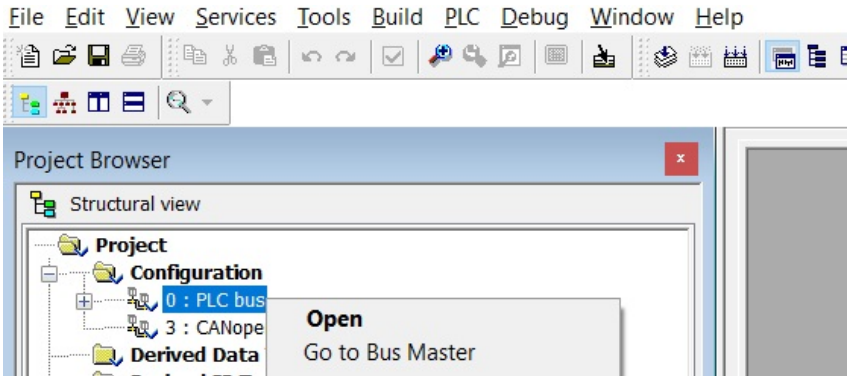
4. From the menu bar, select *File⇒Save As*.
5. In *Save As* pop-up window, enter the file name as *Danfoss*.
6. Press *Save*.



e30bu620.10

Illustration 15: Save As the Project

7. Navigate to *Project Browser⇒Project⇒Configuration*, right-click *PLC bus*, and select *Open*.



e30bu621.10

Illustration 16: PLC Bus Configuration

8. In *PLC BUS* slot, select according to the actual hardware configuration.
9. Right-click and select *New Device*….

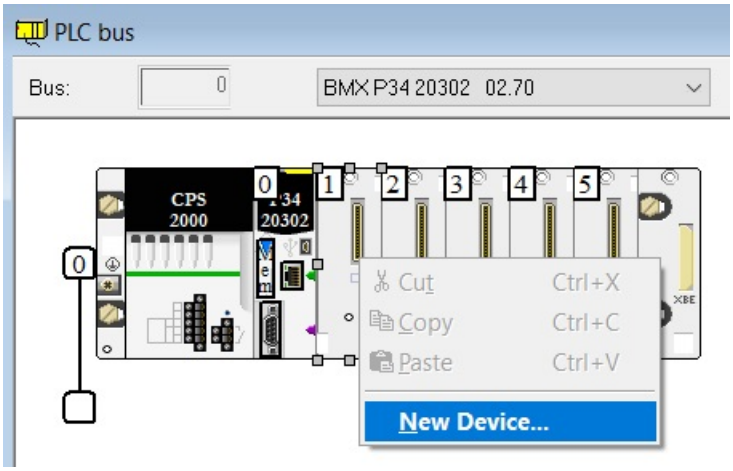


Illustration 17: PLC Bus New Device Configuration

10. From *New Device* tab⇒*Communication*, select the NOC communication module *BMX NOC 0401.2*, and click *OK*.

NOTICE

The hardware configuration has to match the actual used PLC.

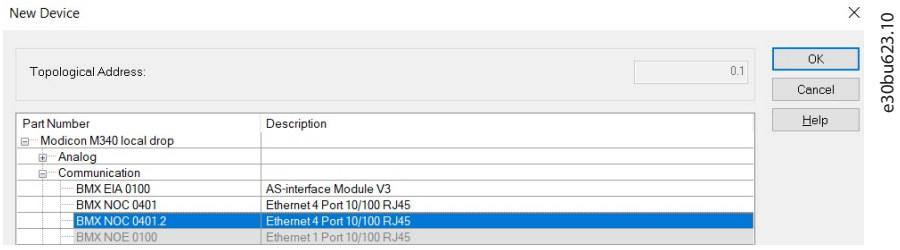


Illustration 18: NOC Module Selection

11. In *Properties of device* pop-up window, use the default DTM name and click *OK*.

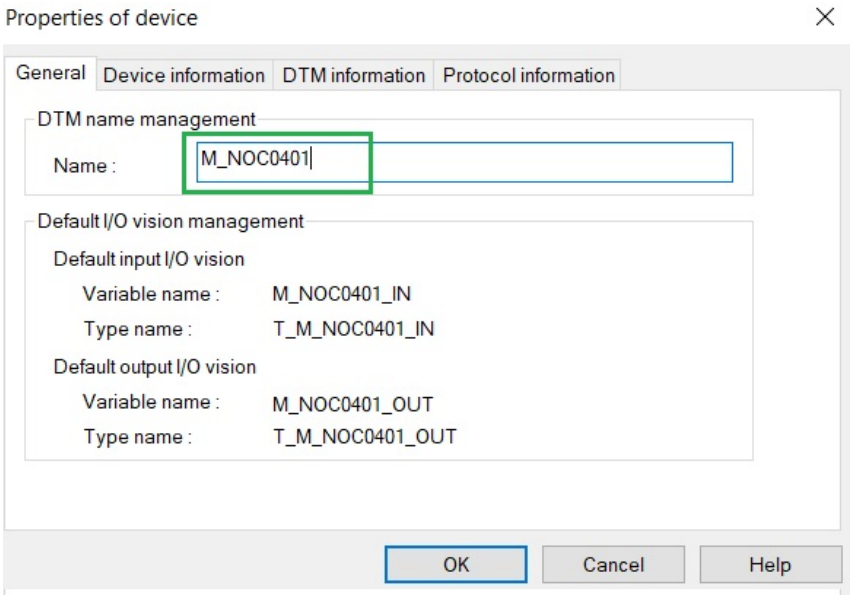


Illustration 19: NOC DTM Name Creation

12. From the menu bar, select *Build⇒Rebuild All Project*.

13. Navigate to *Project Browser⇒Variables & FB Instances⇒Derived Variables*, variables are generated automatically as shown in the following illustration.

- NOC DTM input variable⇒M_NOC0401_IN
- NOC DTM output variable⇒M_NOC0401_OUT

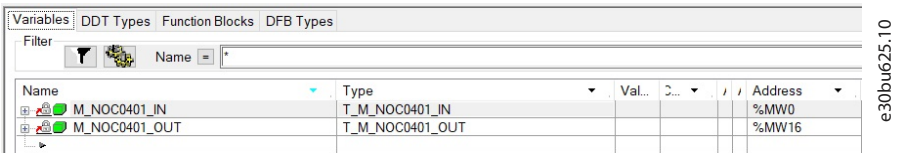


Illustration 20: NOC DTM Input & Output Variables

14. Navigate to *Project Browser⇒Project⇒Configuration⇒PLC bus⇒Rack 0⇒BMX NOC 0401.2*.

15. Select the NOC module, right-click, and select *Open*.

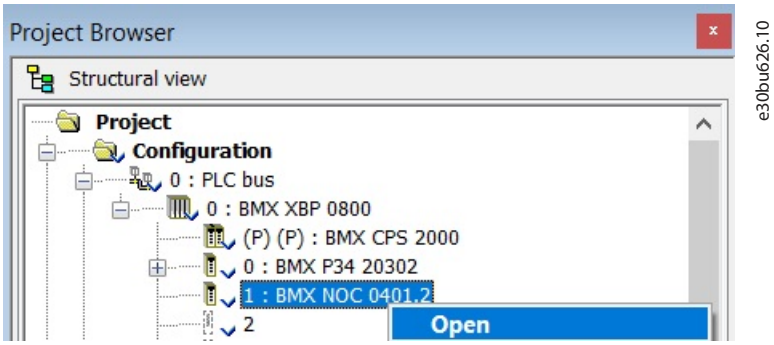


Illustration 21: NOC Module Selection

16. Configure the NOC module memory size as shown in the following illustration.

- Input: %MW, index: 0, max size: 100.
- Output: %MW, index: 100, max size: 100.

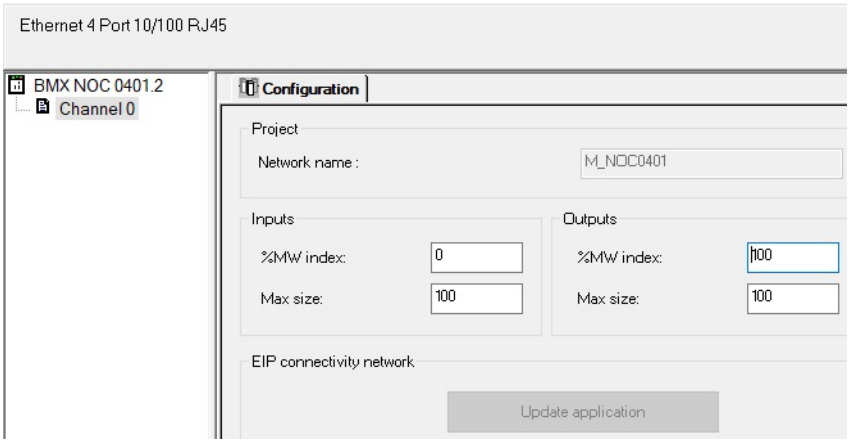


Illustration 22: Configure the NOC Module Memory Size

17. From the menu bar, select *Edit⇒Validate* to validate the modification and build the changes.

A If more number of modbus devices DTM is added, increase each memory size every time. Then maximum memory size 100 is reserved.

B If basic operation function block (DUT1) is configured, the NOC module memory size is occupied as *Input: (%MW Index: 0, Max Size: 20) and Output: (%MW Index: 20, Max Size: 20)*.

C If flexible control function block (DUT1) is configured, then the NOC module memory size is occupied as *Input: (%MW Index: 0,Max Size: 26) and Output: (%MW Index: 26, Max Size: 26)*.

18. From the menu bar, select *Build⇒Rebuild All Project*.

19. Navigate to *Project Browser⇒Variables & FB Instances⇒Derived Variables*, variables are generated automatically as shown in the following illustration.

- NOC DTM input variable⇒M_NOC0401_IN.
- NOC DTM output variable⇒M_NOC0401_OUT.

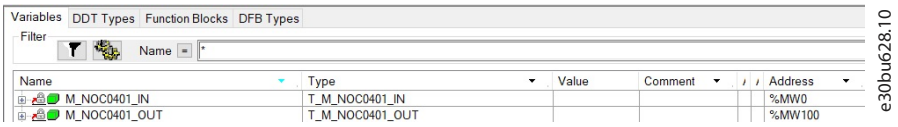


Illustration 23: NOC DTM M_NOC0401_IN & M_NOC0401_OUT Variable

20. From the menu bar, select *Tools⇒DTM Browser*.

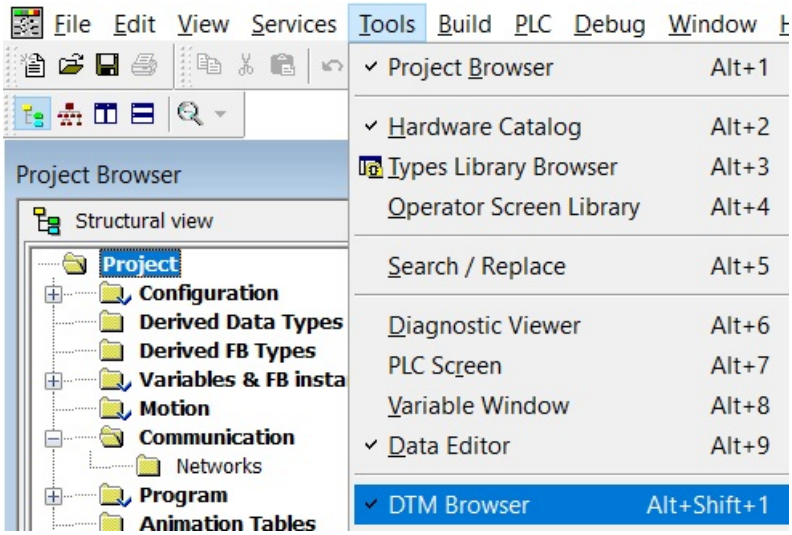


Illustration 24: Select DTM Browser

21. In *DTM Browser*, select the NOC module *M_NOC0401* DTM.

22. Right-click and select *Open*.

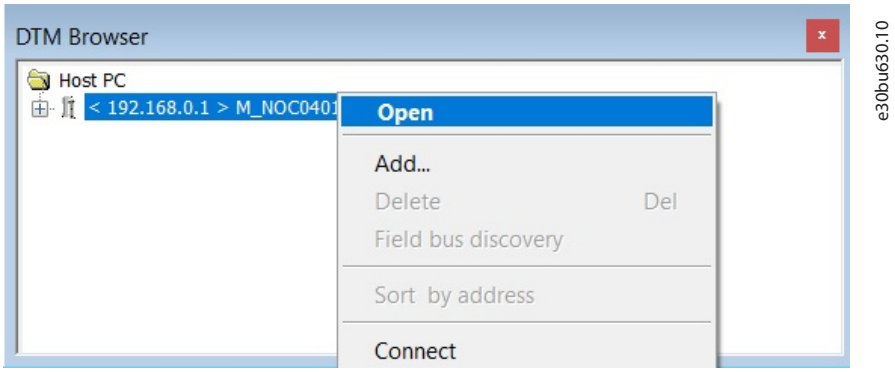


Illustration 25: Open NOC Module DTM

23. In *NOC Module DTM⇒Channel Properties⇒TCP/IP*, assign the IP address as *192.168.0.1*.

24. Assign the subnet as *255.255.255.0*.

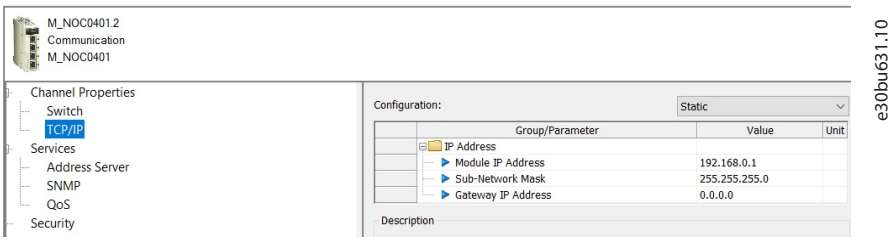


Illustration 26: NOC DTM IP Address Configuration

2.3 Adding Function Blocks to the Main Program

Procedure

1. Navigate to *Project Browser⇒Structural View⇒Project⇒Program⇒TASK⇒MAST⇒Sections*.
2. Right-click and select *New Section*….

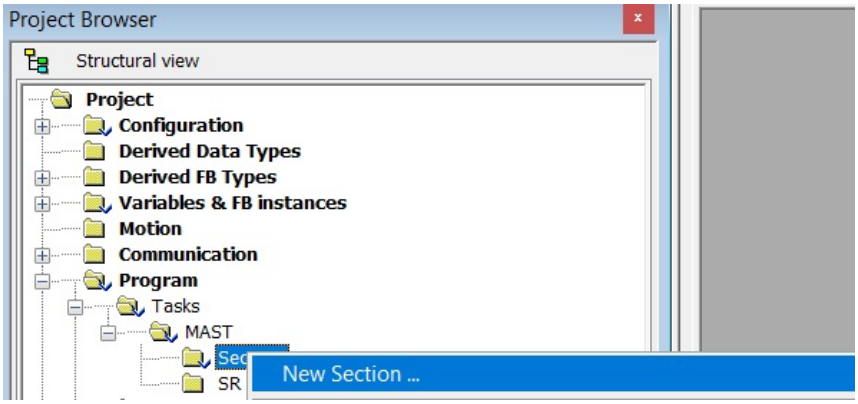


Illustration 27: New Section Creation

3. Enter *MAIN* as the name, and select the language *FBD* based on the project requirement.
4. Click *OK*.

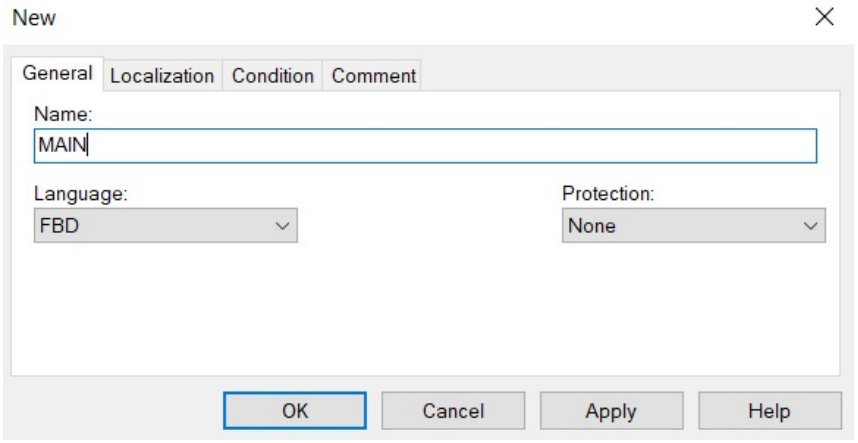
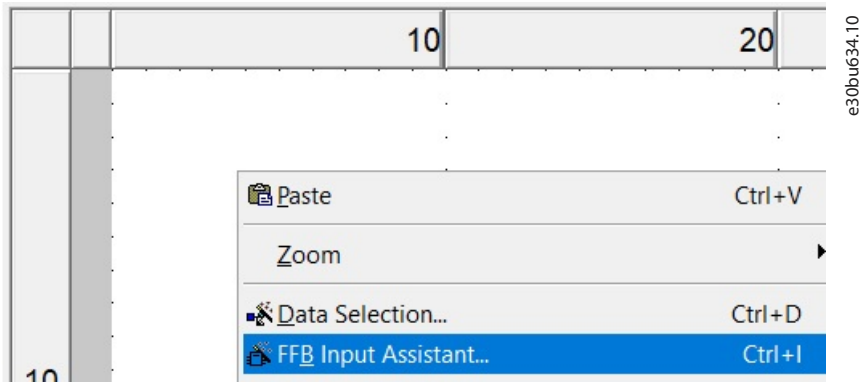


Illustration 28: Main Section Creation

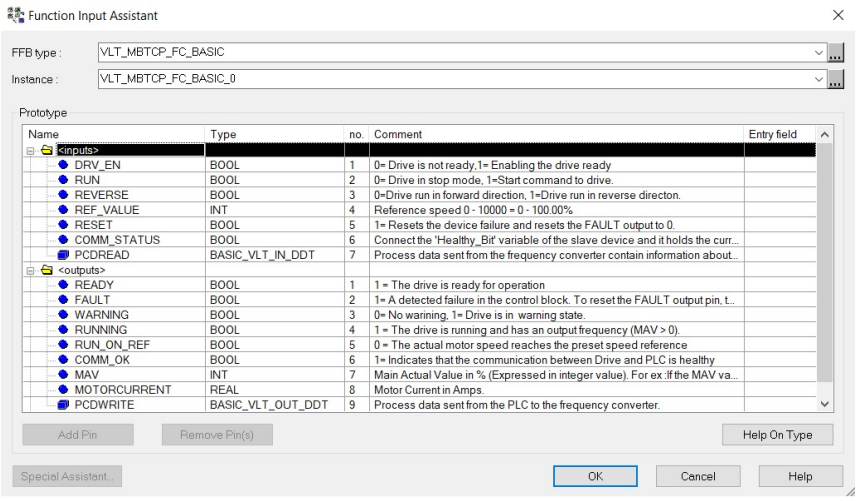
5. In *Sections⇒Main*, import the Danfoss function blocks.
6. Press *Ctrl+I*, or right-click the empty section.
7. Select *FFB Input Assistant*....



e30bu634.10

Illustration 29: FFB Input Assistant

8. From *Function Input Assistant*, in *FFB type* text field enter the function block name as *VLT_MBTCP_FC_BASIC* (or *VLT_MBTCP_FC_PARAM_ACCESS*, or *VLT_MBTCP_FC_DIAGNOSTICS*, or *VLT_MBTCP_FC_FLEXIBLE_CTRL*).
9. In *Instance* text field enter the instance name for the instantiated function block.



e30bu635.10

Illustration 30: VLT_MBTCP_FC_BASIC Function Block Input Assistant

→ *VLT_MBTCP_FC_BASIC* function block successfully called under *Sections* folder.

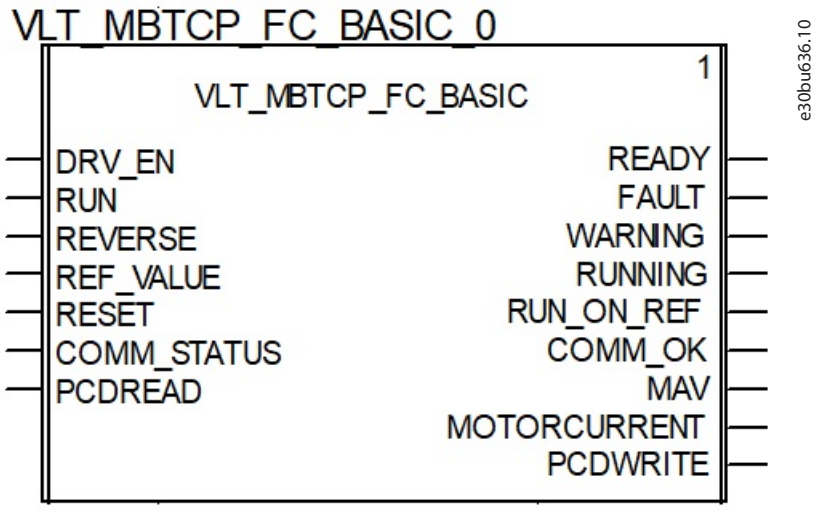


Illustration 31: VLT_MBTCP_FC_BASIC Function Block Instances

2.4 Creating Process Variables for the Function Block Instance

Procedure

1. Create a variable *DRV_EN* for the input pin *DRV_EN* of *VLT_MBTCP_FC_BASIC _0* and press *Enter*.

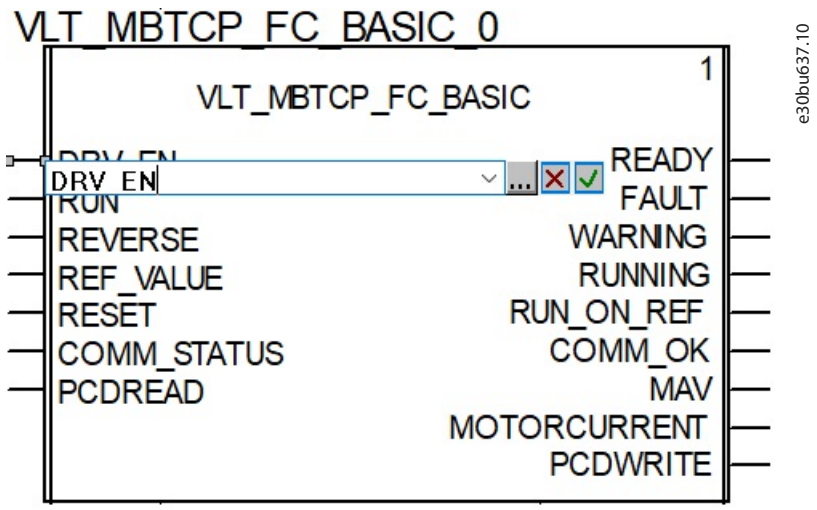


Illustration 32: VLT_MBTCP_FC_BASIC Function Block Input Pin Variable Creation

→ *Create variable?* window appears.

2. Click the green button.

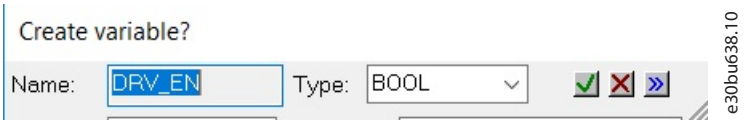


Illustration 33: DRV_EN Variable Creation

3. Repeat the same steps to create variables for I/O pins as shown in the following illustration.

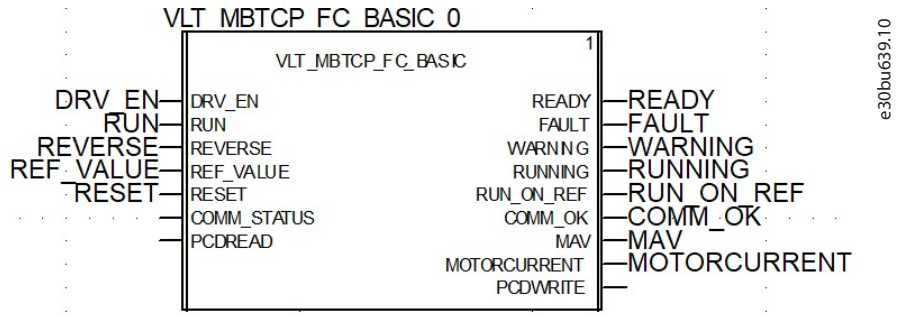


Illustration 34: VLT_MBTCP_FC_BASIC Function Block with All Variables

NOTICE

It is not necessary to create variables for all I/O pins. However, it is mandatory to map the variables for *PCDREAD*, *PCDWRITE*, and *COMM_STATUS* pins of the function block instance.

4. Follow the same steps to create the variables for *VLT_MBTCP_FC_PARAM_ACCESS*, *VLT_MBTCP_FC_DIAGNOSTICS*, and *VLT_MBTCP_FC_FLEXIBLE_CTRL* function blocks.

2.5 DTM Configuration for VLT_MBTCP_FC_BASIC or VLT_MBTCP_FC_FLEXIBLE_CTRL Function Block

Procedure

1. From the menu bar, select *Tools⇒DTM Browser*.
2. In *DTM Browser*, select the NOC module DTM.
3. Right-click and select *Add*.

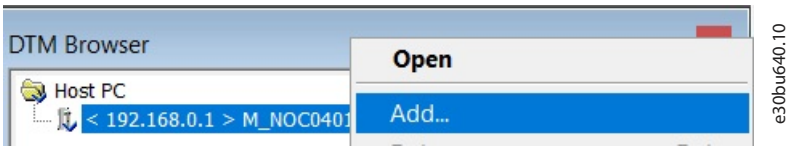


Illustration 35: Add DTM

→ DTM devices list dialog box appears.

4. Select the *Generic Modbus Device DTM*.
5. Click *Add DTM*.

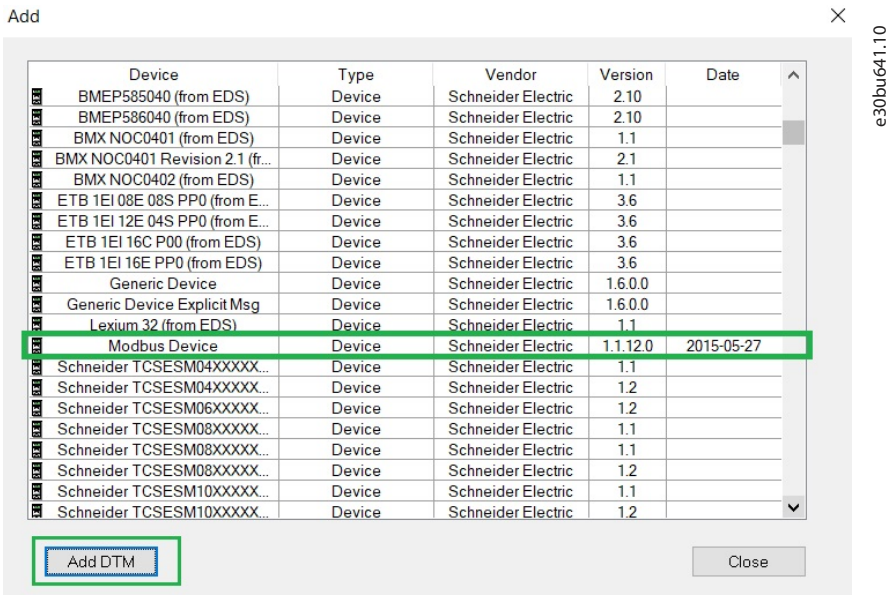


Illustration 36: Add Modbus DTM

6. In *Properties of device* tab, enter the DTM name *DANFOSS_DRIVE_VLT1*.
7. Click *OK*.

NOTICE

Follow this step only when *VLT_MBTCP_FC_BASIC* function block is used.

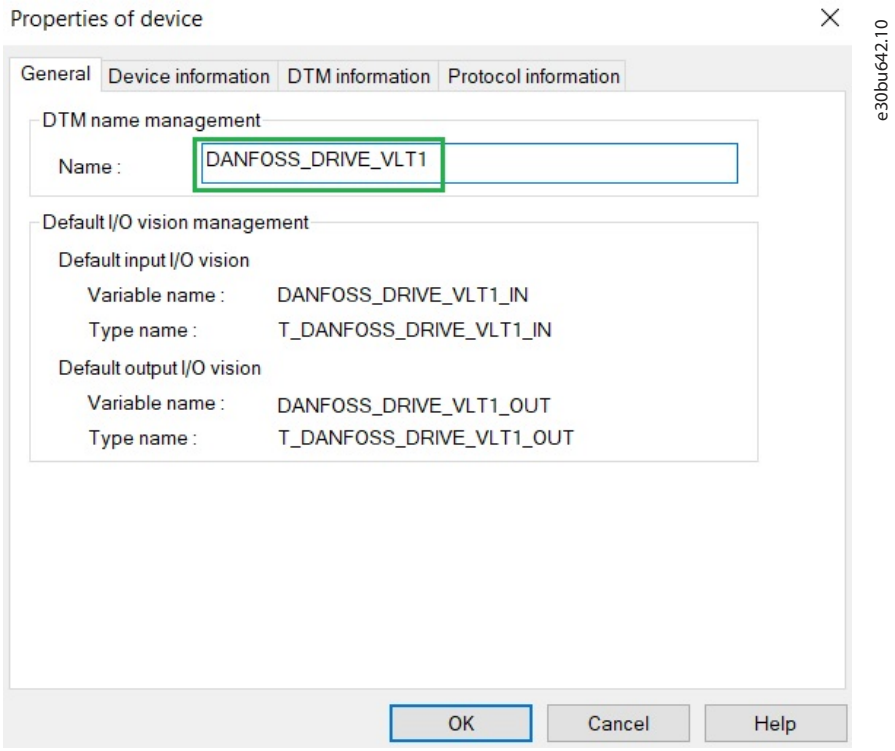


Illustration 37: Modbus Device DTM Name Creation for VLT_MBTCP_FC_BASIC Function Block

8. In *Properties of device* tab, enter the DTM name *DANFOSS_DRIVE_VLT1_FLEX*.
9. Click *OK*.

NOTICE

Follow this step only when *VLT_MBTCP_FC_FLEXIBLE_CTRL* function block is used.

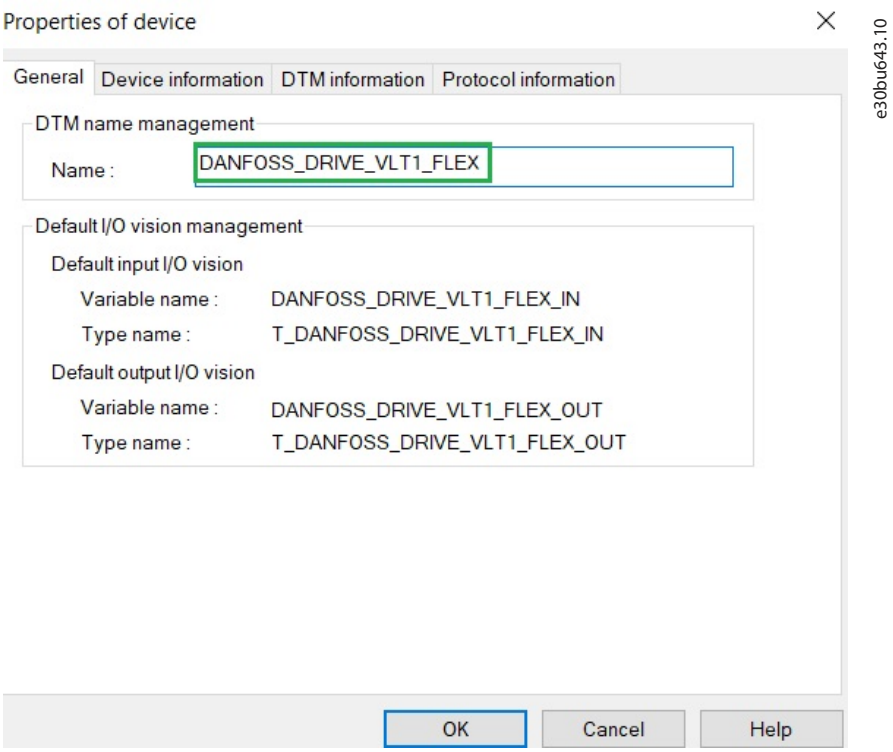


Illustration 38: Modbus Device DTM Name Creation for VLT_MBTCP_FC_FLEXIBLE_CTRL Function Block

10. In *DTM Browser*, select the NOC device *M_NOC0401* DTM.

11. Right-click and select *Open*.

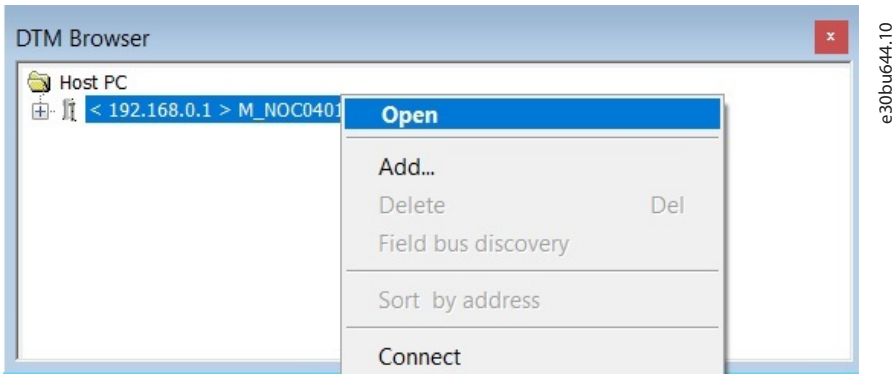
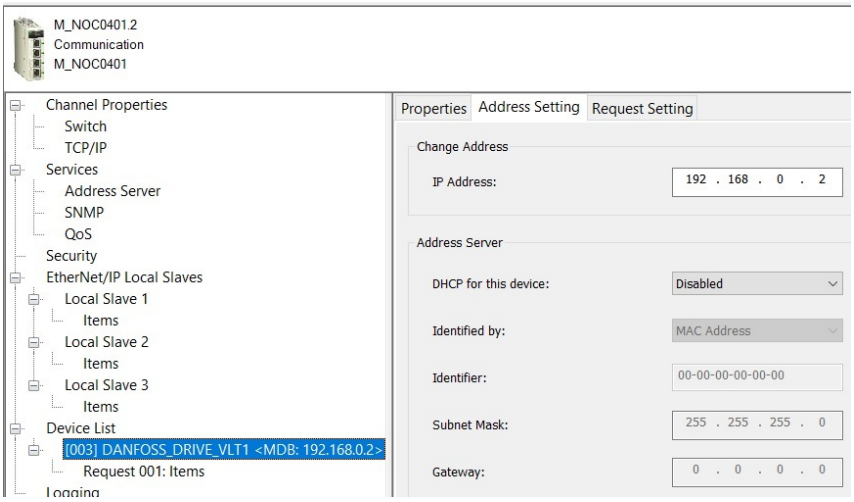


Illustration 39: Open NOC Device DTM

12. From *NOC Module DTM⇒Device list⇒*select *DANFOSS_DRIVE_VLT1* or *DANFOSS_DRIVE_VLT1_FLEX*.

13. Assign the IP address as *192.168.0.2* in *Address Setting* tab.



e30bu645.10

Illustration 40: Modbus DTM IP Address Configuration

14. In *NOC Module DTM⇒Device list⇒DANFOSS_DRIVE_VLT1⇒Request Setting* tab, click the *Add Request* button.

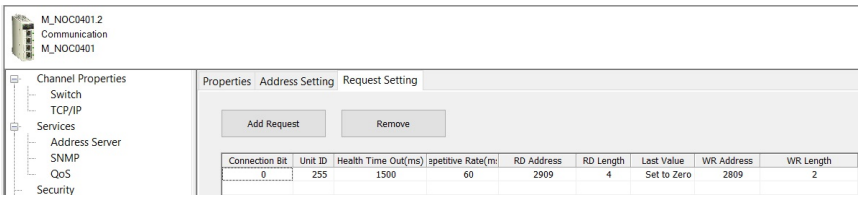
15. Configure the following data: slave (VLT drive) IP address, read reference slave address (RD ref slave), read data length (RD length), write reference slave address (WR ref slave), and write data length (WR length).

NOTICE

Follow this step only when *VLT_MBTCP_FC_BASIC* function block is used.

Address and length assignment for *VLT_MBTCP_FC_BASIC* function block:

- Read reference slave address=2909
- Read RD length=4
- Write reference slave address=2809
- Write RD length=2
- Last value=set to zero



e30bu646.10

Illustration 41: Modbus DTM Address and Length Assignment for VLT_MBTCP_FC_BASIC Function Block

16. In *NOC Module DTM⇒Device list⇒DANFOSS_DRIVE_VLT1⇒Request Setting* tab, click the *Add Request* button.

17. Configure the following data: slave (VLT drive) IP address, read reference slave address (RD ref slave), read data length (RD length), write reference slave address (WR ref slave), and write data length (WR length).

NOTICE

Follow this step only when *VLT_MBTCP_FC_FLEXIBLE_CTRL* function block is used.

Address and length assignment for *VLT_MBTCP_FC_FLEXIBLE_CTRL* function block:

- Read reference slave address=2909
- Read RD length=10
- Write reference slave address=2809
- Write RD length=10
- Last value=set to zero

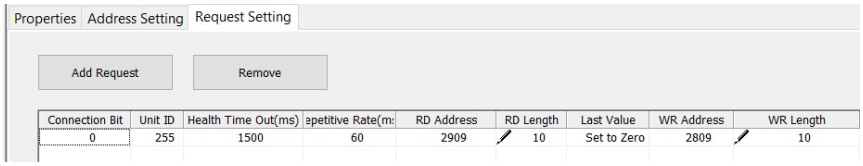


Illustration 42: Modbus DTM Address and Length Assignment for VLT_MBTCP_FC_FLEXIBLE_CTRL Function Block

18. Navigate to *Project Browser⇒Project⇒Configuration⇒PLC bus⇒Rack 0⇒BMX NOC 0401.2*, select the NOC module.

19. Right-click and select *Open*.

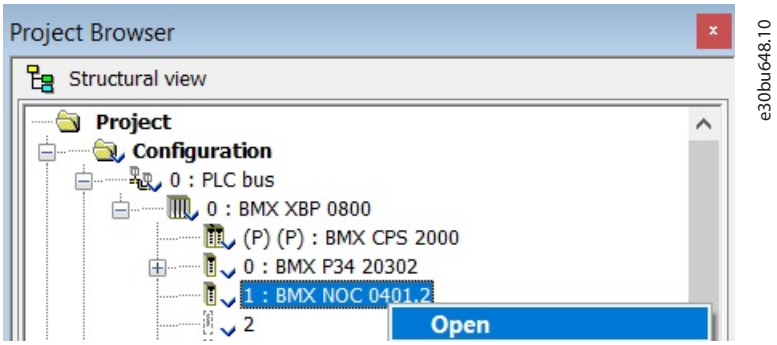


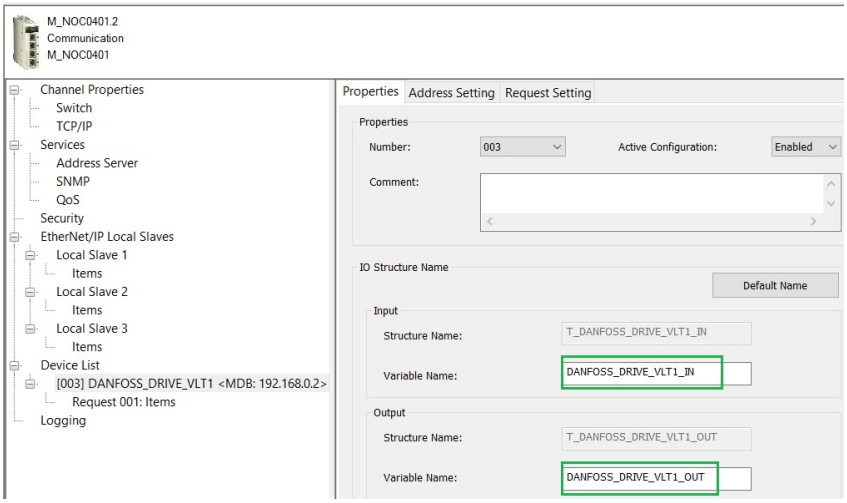
Illustration 43: NOC Module Selection

20. From the menu bar, select *Edit⇒Validate* to validate the modification and build the changes.

21. In *NOC Module DTM⇒Device list⇒DANFOSS_DRIVE_VLT1⇒Properties* tab⇒*IO Structure Name*, verify that the input variable name is *DANFOSS_DRIVE_VLT1_IN* and the output variable name is *DANFOSS_DRIVE_VLT1_OUT*.

NOTICE

Follow this step only when *VLT_MBTCP_FC_BASIC* function block is used.

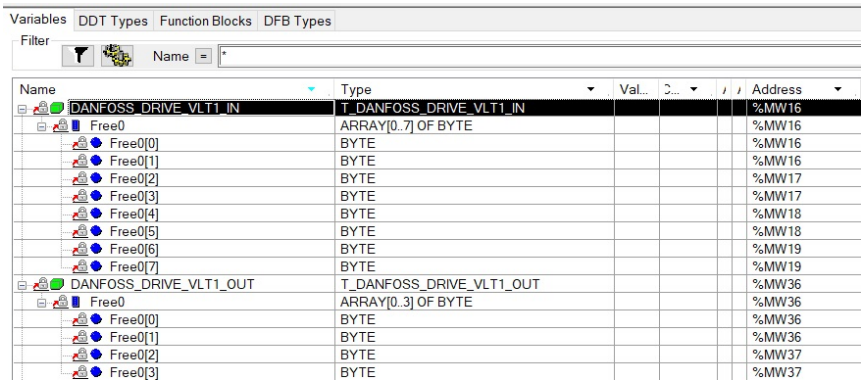


e30bu649.10

Illustration 44: Modbus DTM VLT_MBTCP_FC_BASIC FB Input & Output Variable Name

22. Build the project, and navigate to *Project Browser⇒Variables & FB Instances⇒Derived Variables*.

23. Variables generate automatically as shown in the following illustration.



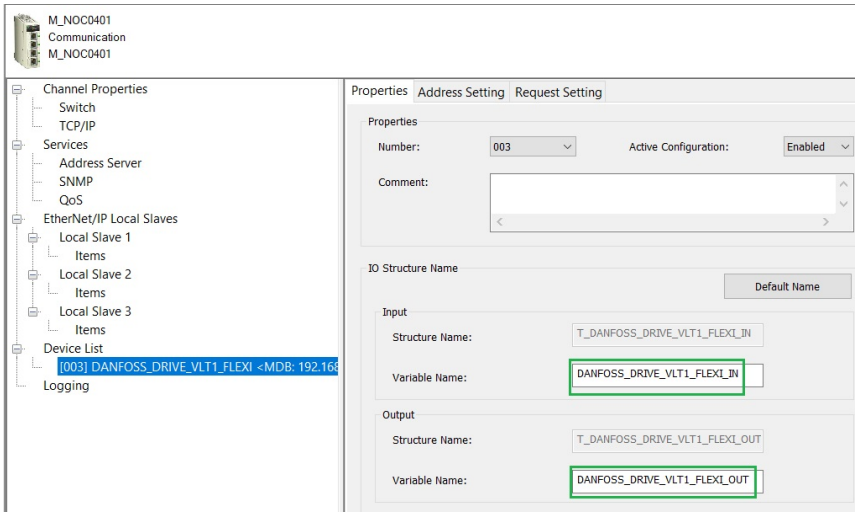
e30bu650.10

Illustration 45: Modbus DTM VLT_MBTCP_FC_BASIC FB Input & Output Variables

24. In *NOC Module DTM⇒Device list⇒DANFOSS_DRIVE_VLT1_FLEXI⇒Properties* tab⇒*IO Structure Name*, verify that the input variable name is *DANFOSS_DRIVE_VLT1_FLEXI_IN* and the output variable name is *DANFOSS_DRIVE_VLT1_FLEXI_OUT*.

NOTICE

Follow this step only when *VLT_MBTCP_FC_FLEXIBLE_CTRL* function block is used.

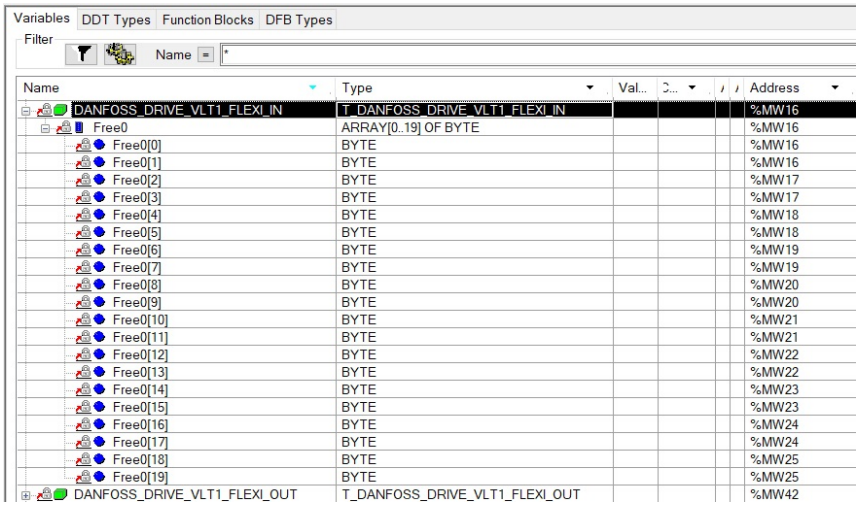


e30bu651.10

Illustration 46: Modbus DTM VLT_MBTCP_FC_FLEXIBLE_CTRL FB Input & Output Variable Name

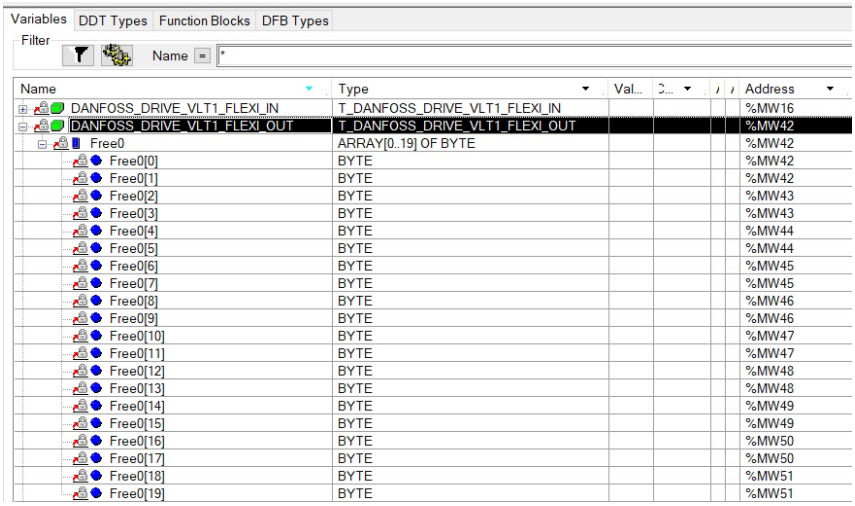
25. Build the project, and navigate to *Project Browser⇒Variables & FB Instances⇒Derived Variables*.

26. Variables generate automatically as shown in the following illustrations.



e30bu652.10

Illustration 47: Modbus DTM VLT_MBTCP_FC_FLEXIBLE_CTRL FB Input Variable



e30bu653.10

Illustration 48: Modbus DTM VLT_MBTCP_FC_FLEXIBLE_CTRL FB Output Variable

2.6 Mapping Variables

2.6.1 PCDREAD and PCDWRITE Variable Mapping

Procedure

1. Once variables configuration is completed successfully, select *Project Browser⇒Variables & FB Instances⇒Variables*.
2. *DANFOSS_DRIVE_VLT1_IN* and *DANFOSS_DRIVE_VLT1_OUT* variables are created as shown in the following illustration.

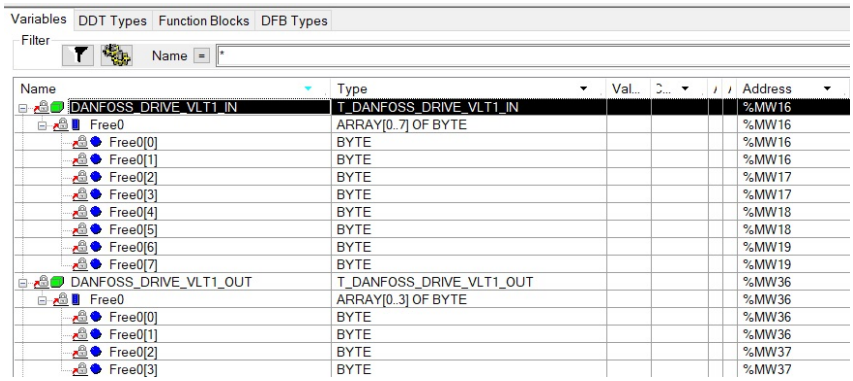


Illustration 49: DANFOSS_DRIVE_VLT1_IN & DANFOSS_DRIVE_VLT1_OUT Variables

3. From *Program⇒TASK⇒MAST⇒Sections⇒Main*, select *VLT_MBTCP_FC_BASIC* function block.
4. Double-click the *PCDREAD* pin, and select the dotted button.

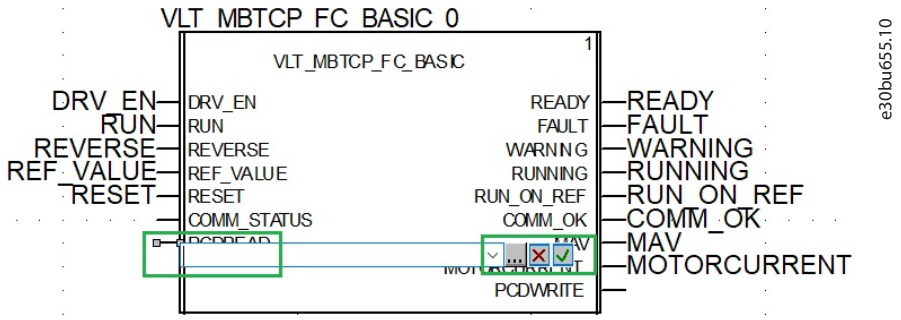


Illustration 50: VLT_MBTCP_FC_BASIC Function Block with PCDREAD Configuration

5. In *FBD-Editor: Instance Selection*, put the check mark on *Inside structure*.
6. Select *DANFOSS_DRIVE_VLT1_IN* variable and click *OK*.
7. Do similarly for *PCDWRITE* with *DANFOSS_DRIVE_VLT1_OUT* variable.
8. The variables are mapped with *PCDREAD* and *PCDWRITE*.
9. *VLT_MBTCP_FC_BASIC* function block variable are successfully configured and ready for use.

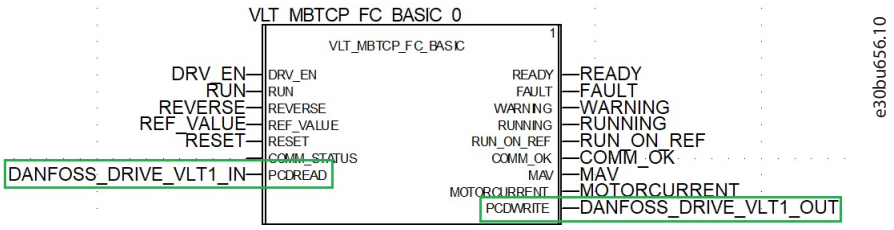


Illustration 51: VLT_MBTCP_FC_BASIC Function Block with PCDREAD & PCDWRITE

10. Follow the same steps for mapping the *PCDREAD* with *DANFOSS_DRIVE_VLT1_FLEX_IN* input variable and *PCDWRITE* with *DANFOSS_DRIVE_VLT1_FLEX_OUT* output variable to configure the *VLT_MBTCP_FC_FLEXIBLE_CTRL* function block.

2.6.2 COMM_STATUS Variable Mapping

Procedure

1. From the menu bar, select *Tools⇒DTM Browser*.

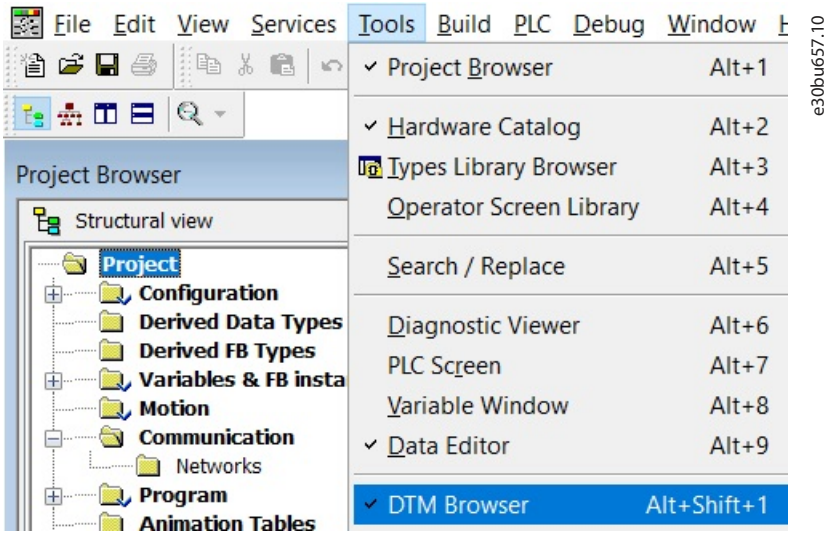


Illustration 52: Select DTM Browser

2. From *DTM Browser*, select the NOC module *M_NOC0401* DTM.
3. Right-click and press *Open*.

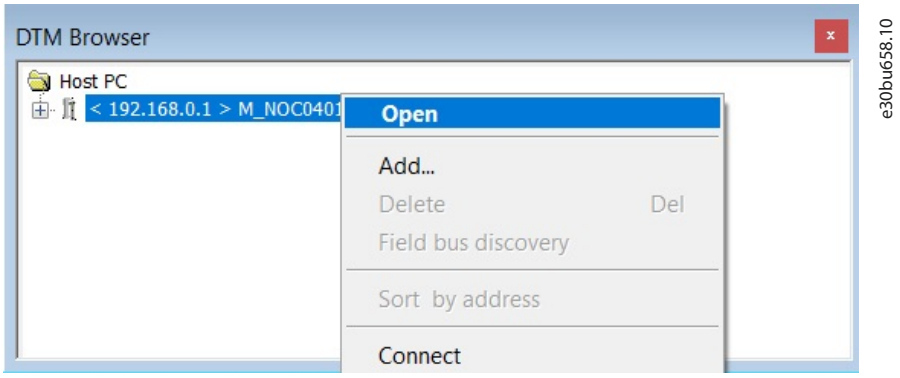
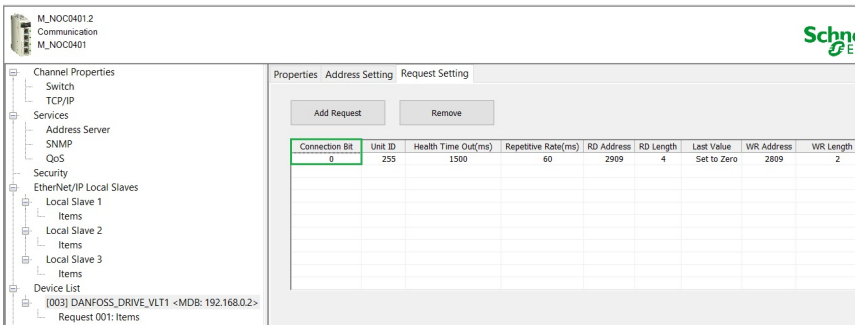


Illustration 53: Open NOC Module DTM

4. From *NOC Module DTM⇒Device list⇒DANFOSS_DRIVE_VLT1⇒Request Setting*, verify if *Connection Bit* is set as *0*, which means the Modbus device DTM slave is in healthy status.

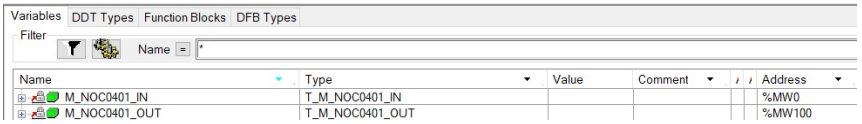


e30bu659.10

Illustration 54: Modbus Device DTM Slave Connection Bit

5. Once variables configuration is completed successfully, select *Project Browser⇒Variables & FB Instances⇒Variables*.

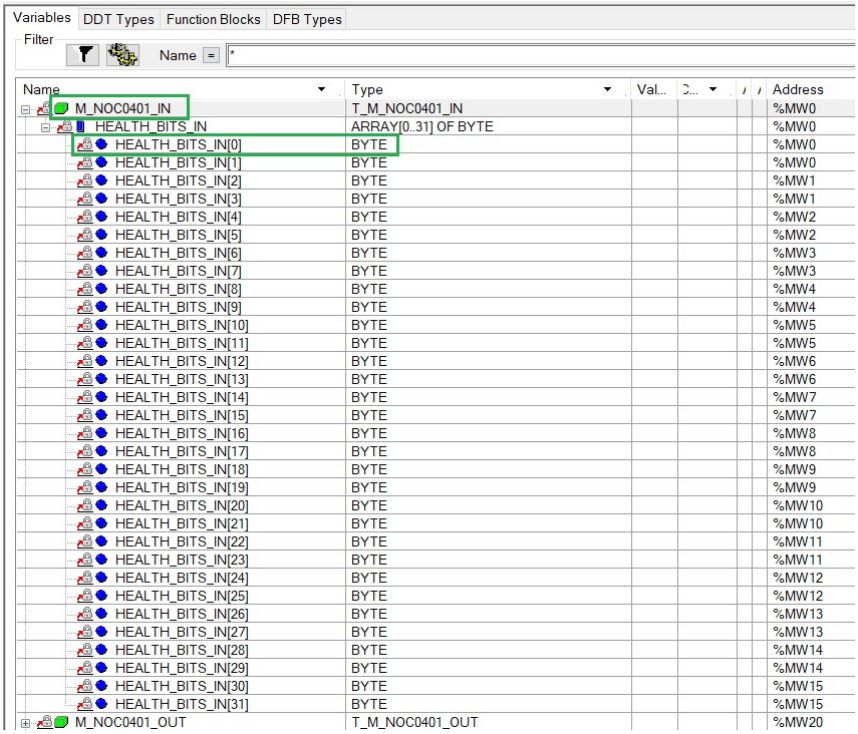
   → *M_NOC0401_IN* and *M_NOC0401_OUT* variables are created as shown in the following illustration.



e30bu660.10

Illustration 55: M_NOC0401_IN Input & M_NOC0401_OUT Output Variables

6. Expand the *M_NOC0401_IN* input variable⇒*HEALTH_BIT_IN*⇒*HEALTH_BIT_IN[0]*, verify whether variable *HEALTH_BIT_IN[0]* is created in byte data type as shown in the following illustration.



e30bu661.10

Illustration 56: M_NOC0401_IN Input Variable

7. From *Main Program*, select *VLT_MBTCP_FC_BASIC* function block.
8. Double-click *COMM_STATUS* pin, and select the dotted button.
9. In *FBD-Editor: Instance Selection*, put the check mark on *Inside structure*.

10. Select and expend the variable *M_NOC0401_IN* by pressing the plus button.

11. Select and expend the variable *HEALTH_BITS_IN* by pressing the plus button.

12. Select the variable *HEALTH_BITS_IN[0]* and add *.0* as shown in the following illustration.

13. Press *OK* to assign the variable to *COMM_STATUS* input.

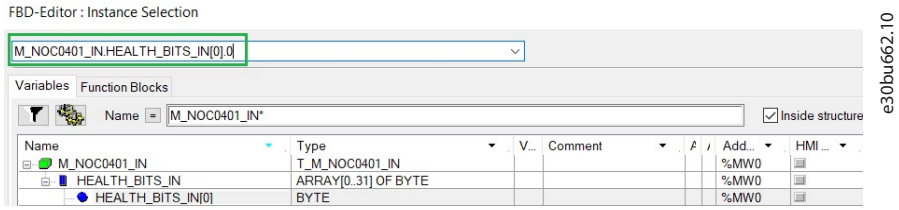


Illustration 57: NOC M_NOC0401_IN Input Variable with HEALTH_BITS_IN[0].0

NOTICE

Modbus Device DTM⇒DANFOSS_DRIVE_VLT1⇒Request Setting, Connection Bit is *0*, therefore, add *.0* with *HEALTH_BITS_IN[0]* variable for *COMM_STATUS* input mapping.

14. The variables are mapped with *COMM_STATUS* as shown in the following illustration.

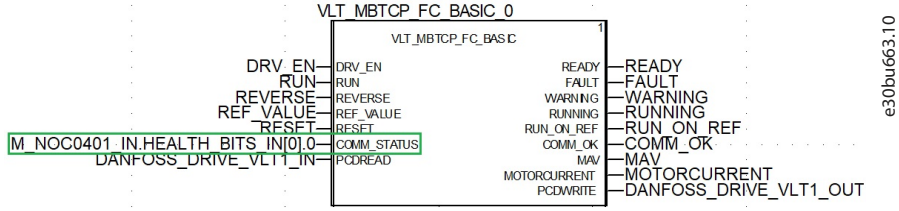


Illustration 58: COMM_STATUS Input is Mapped with M_NOC0401_IN.HEALTH_BITS_IN[0].0

15. Follow the same steps for mapping the *COMM_STATUS* input for *VLT_MBTCP_FC_FLEXIBLE_CTRL* function block.

2.7 Modbus TCP Slave Address

For addressing a station on Ethernet, the string can be made up of: *r.m.c{hostAddr}*

Table 10: Address Description

r	Rack number (rack).
m	Module position.
c	Channel number (channel).
hostAddr	Modbus TCP slave IP address.

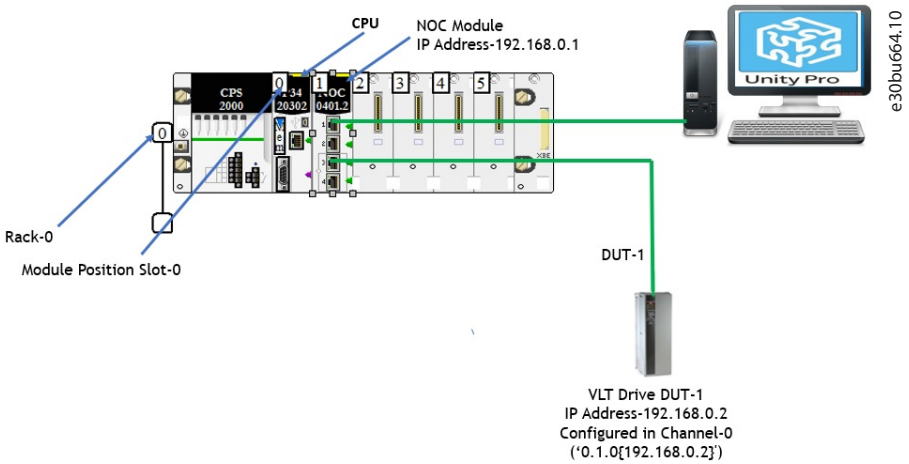


Illustration 59: Hardware Configuration

For the hardware configuration, DUT1 Modbus TCP slave address is ('0.1.0{192.168.0.2}'), and the same address can be used in *VLT_MBTCP_FC_PARAM_ACCESS* and *VLT_MBTCP_FC_DIAGNOSTICS* function blocks as shown in the following illustrations.

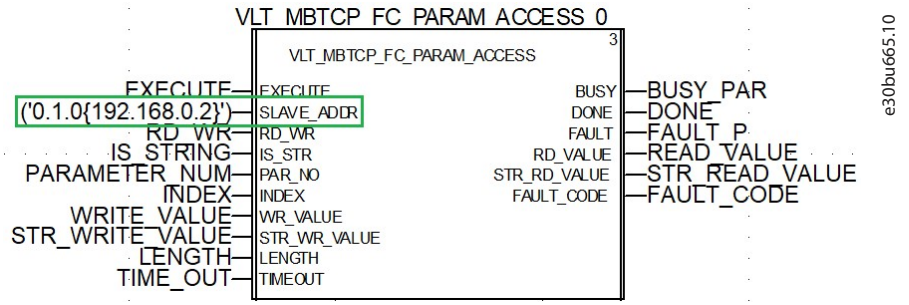


Illustration 60: VLT_MBTCP_FC_PARAM_ACCESS Function block

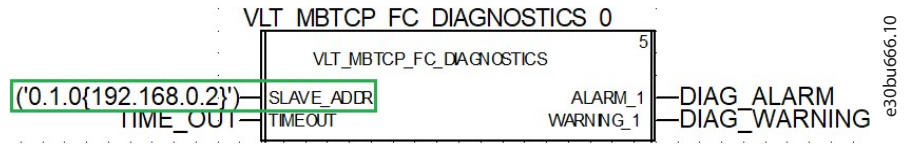


Illustration 61: VLT_MBTCP_FC_DIAGNOSTICS Function Block

2.8 Dynamic Array Configuration

Context:

Dynamic array is used in *VLT_MBTCP_FC_PARAM_ACCESS* function block, this section describes how to enable the dynamic array in the project setting.

Procedure

1. From the menu bar, select *Tools⇒Project Settings⇒General⇒Variables*.
2. Turn on *Directly represented array variables* and *Allow dynamic arrays (ANY_ARRAY_XXX)*.
3. Select *OK* and build changes.

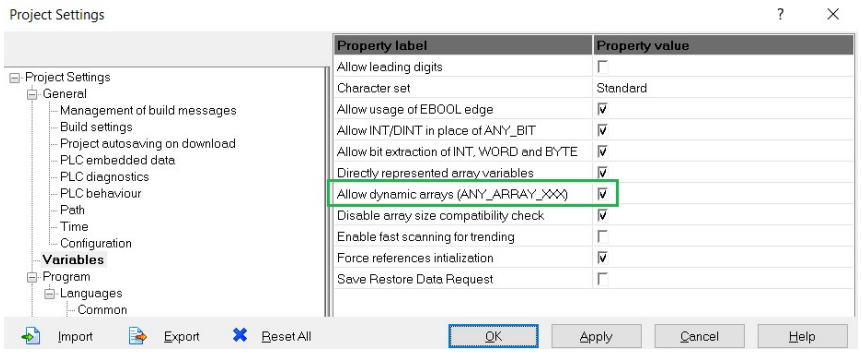


Illustration 62: Dynamic Array Setting

2.9 Downloading a Project to PLC

Procedure

1. From the menu bar, select *File⇒Save*.

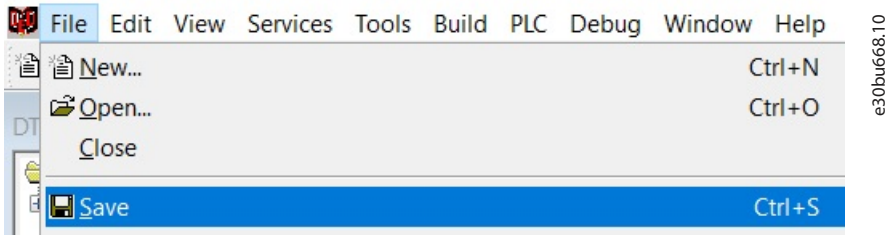


Illustration 63: Project Save

2. From the menu bar, select *Build⇒Build Changes*.

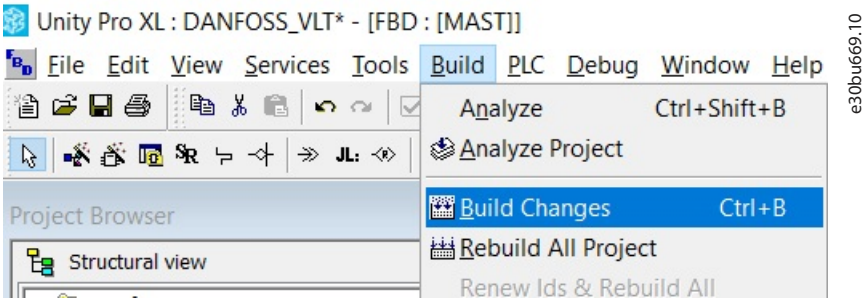


Illustration 64: Build Changes

3. From the menu bar, select *PLC⇒Set Address*….

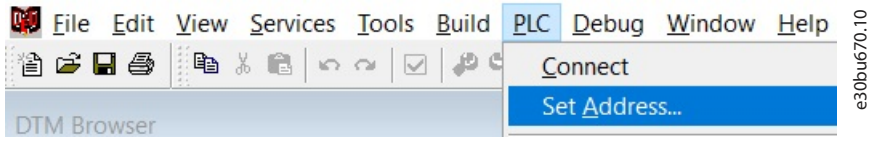


Illustration 65: Set Address

4. In *Set Address* tab, enter IP address *192.168.0.1* in the address field.

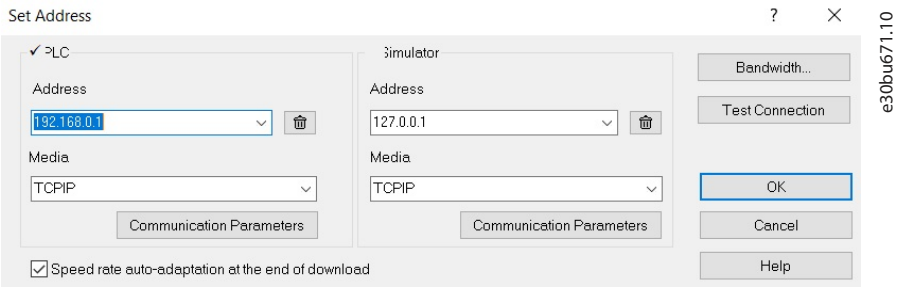


Illustration 66: Set PLC IP Address

5. Click *Test Connection*.

   → The target connects successfully as shown in the following illustration.

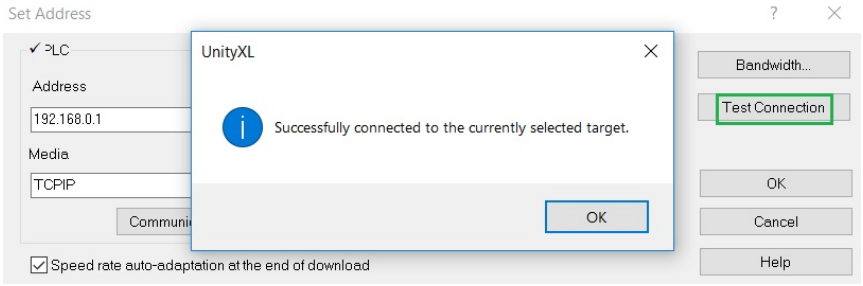


Illustration 67: PLC IP Address Configured

6. Press *OK*.
7. From the menu bar, select *PLC⇒Connect*.

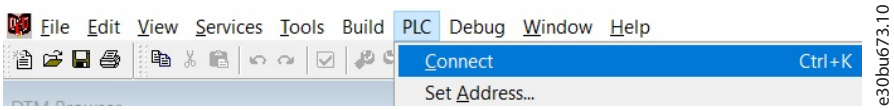


Illustration 68: PLC Connect

→ PLC connection establishes successfully.

8. From the menu bar, select *PLC⇒Transfer Project to PLC*.

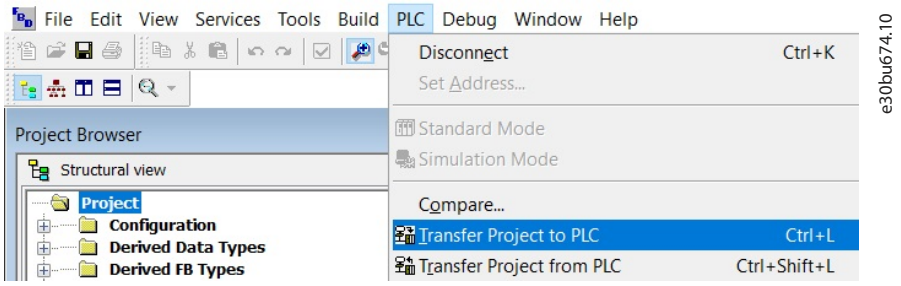


Illustration 69: Select Transfer Project to PLC

9. From the *Transfer Project to PLC* tab popup, select the *PLC Run after Transfer* checkbox.
10. Click *Transfer*.

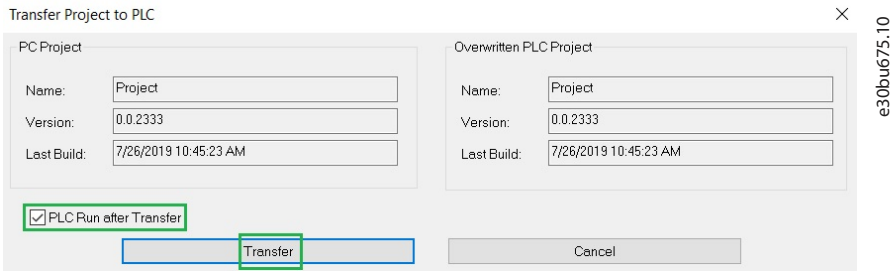


Illustration 70: Transfer Project to PLC

11. Click *OK* in the *Stop* popup window.

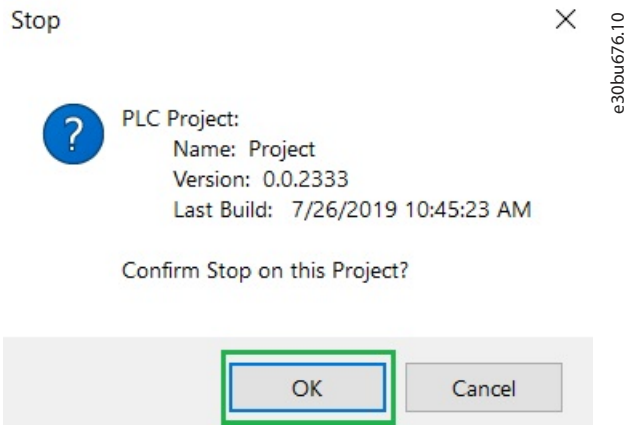


Illustration 71: Confirm PLC for Transfer

12. Click *OK* in the *Run* popup window.

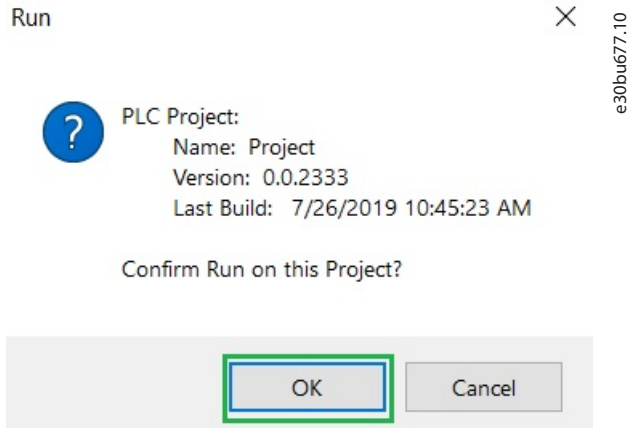


Illustration 72: Confirm Run on This Project

3 Examples

3.1 General Configuration of the Drive

Procedure

1. When the drive is commissioned, set *parameter 0-03 Regional Settings* before any other changes are made to the drive through the LCP.
2. Verify the following parameter settings to ensure the PLC has control of the drive.

Table 11: Parameter Settings

Parameter	Value
Parameter 8-01 Control Site	*[0] Digital and ctrl.word*, or *[2] Control word only*
Parameter 8-02 Control Word Source	*[3] Option A*

3. When *parameter 8-01 Control Site* is set to *[0] Digital and Ctrl. Word*, establish a connection between terminal 12/13 and terminal 27 to control the motor.
4. The default setting of the drive allows the drive to continue operation if the communication is lost to the PLC. If this operation is not wanted, change *parameter 8-04 Control Word Timeout Function* via the Main Menu.

Table 12: Parameter Settings

Parameter	Value
Parameter 8-04 Control Word Timeout Function	*[0] Off*, or *[1] Freeze output*, or *[2] Stop*, or *[3] Jogging*, or *[4] Max. speed*, or *[5] Stop and trip*

5. Verify that *parameter 8-10 Control Word Profile* is set correctly via the Main Menu.

The function block requires that *parameter 8-10 Control Word Profile* is set to *[0] FC Profile (DEFAULT)*. If *parameter 8-10 Control Word Profile* is set to *PROFIdrive Profile*, the function block does not work as expected and leads to malfunction.

6. Ensure physically that LCP mode is set to *Auto On* mode.

3.2 Basic Operation Function Block

Context:

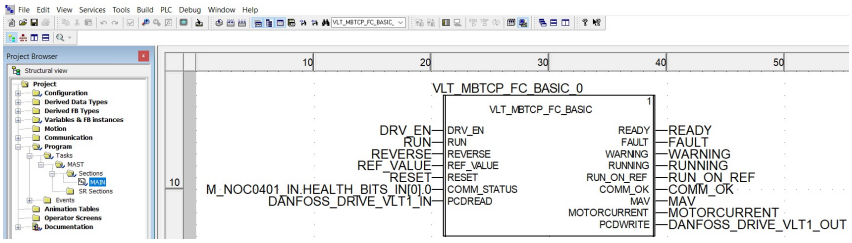


Illustration 73: Basic Operation Function Block

Procedure

1. Map the *PCDREAD*, *PCDWRITE* & *COMM_STATUS* variables with the flexible control function block.

Table 13: Mapping Variables

Type	Pin name	Data type	Mapping
Input	PCDREAD	FLEXI_CTRL_VLT_IN_DDT	DANFOSS_DRIVE_VLT1_FLEXI_IN
Input	COMM_STATUS	BOOL	M_NOC0401_IN.HEALTH_BITS_IN[0].0
Output	PCDWRITE	FLEXI_CTRL_VLT_OUT_DDT	DANFOSS_DRIVE_VLT1_FLEXI_OUT

2. Enable the drive to ready state.
   - **A** Verify the value *TRUE* is set to the *COMM_OK* output pin of the function block. It means the communication between PLC and the drive is OK.
   - **B** Set the value *TRUE* to *DRV_EN* input pin of the function block.
   - **C** Verify the following values with the function block output.

Table 14: Pin Name and Expected Value

Pin name	Expected value
READY	TRUE
FAULT	FALSE
WARNING	FALSE
RUNNING	FALSE
RUN_ON_REF	FALSE
COMM_OK	TRUE
MAV	0
MOTORCURRENT	Same as *parameter 16-14 Motor Current*.

   - **D** Set the value *TRUE* to the *RUN* input pin of the function block.
   - **E** Set the value *10000* to the *REF_VALUE* input pin of the function block.
   - **F** Verify the following values with the function block output.

Table 15: Pin Name and Expected Value

Pin name	Expected value
READY	TRUE
FAULT	FALSE
WARNING	FALSE
RUNNING	TRUE
RUN_ON_REF	TRUE
COMM_OK	TRUE
MAV	10000
MOTORCURRENT	Same as *parameter 16-14 Motor Current*.

   - **G** Set the value *FALSE* to the *DRV_EN* input pin of the function block.

H Verify the following values with the function block output.

Table 16: Pin Name and Expected Value

Pin name	Expected value
READY	FALSE
FAULT	FALSE
WARNING	FALSE
RUNNING	FALSE
RUN_ON_REF	FALSE
COMM_OK	TRUE
MAV	0
MOTORCURRENT	Same as *parameter 16-14 Motor Current*.

I Set the value *FALSE* to the *RUN* input pin of the function block.

3. Start the motor in forward direction.

A Set the value *TRUE* to the *DRV_EN* input pin of the function block.

B Set the value *TRUE* to the *RUN* input pin of the function block.

C Set the value *10000* to the *REF_VALUE* input pin of the function block.

D Verify the following values with the function block output.

Table 17: Pin Name and Expected Value

Pin name	Expected value
READY	TRUE
FAULT	FALSE
WARNING	FALSE
RUNNING	TRUE
RUN_ON_REF	TRUE
COMM_OK	TRUE
MAV	10000
MOTORCURRENT	Same as *parameter 16-14 Motor Current*.

4. Start the motor in reverse direction.

A Set the value *10000* to the *REF_VALUE* input pin of the function block.

B Set the value *TRUE* to the *REVERSE* input pin of the function block. Ensure that *parameter 4-10 Motor Speed Direction* is set to *[2] Both directions*.

C Wait until the motor ramps down and runs in the reverse direction.

D Verify the following values with the function block output.

Table 18: Pin Name and Expected Value

Pin name	Expected value
READY	TRUE

Pin name	Expected value
FAULT	FALSE
WARNING	FALSE
RUNNING	TRUE
COMM_OK	TRUE
MAV	-10000 [±1%]
MOTORCURRENT	Same as *parameter 16-14 Motor Current*.

5. Stop the motor.

A Set the value *FALSE* to the *RUN* input pin of the function block.

B Set the value *0* to the *REF_VALUE* input pin of the function block.

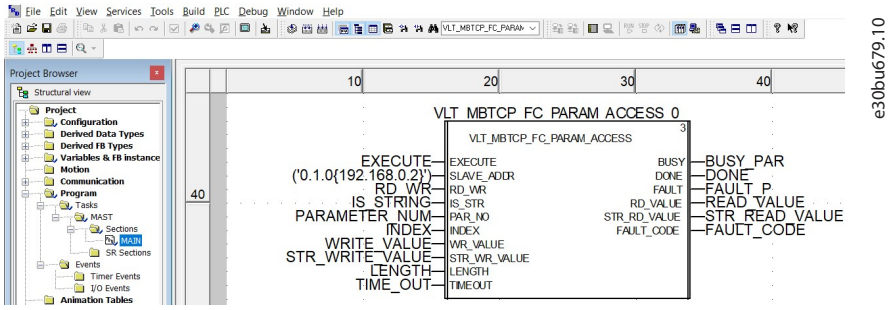
C Verify the following values with the function block output.

Table 19: Pin Name and Expected Value

Pin name	Expected value
READY	TRUE
FAULT	FALSE
WARNING	FALSE
RUNNING	FALSE
COMM_OK	FALSE
MAV	0
MOTORCURRENT	Same as *parameter 16-14 Motor Current*.

3.3 Parameter Access Function Block

Context:



e30bu679.10

Illustration 74: Parameter Access Function Block

Procedure

1. To identify Modbus TCP Slave Address, refer to 2.7 Modbus TCP Slave Address.
2. Read non-array parameters.
   - **A** Set the value *FALSE* to *EXECUTE* input pin of the function block.
   - **B** Set *parameter 3-41 Ramp 1 Ramp Up Time* to value *2.00* using the LCP.
   - **C** Set the following values to the input pin of the function block.

Table 20: Pin Name and Set Value

Pin Name	Set value for *parameter 3-41 Ramp 1 Ramp Up Time*
RD_WR	FALSE
PAR_NO	341
INDEX	0
LENGTH	2
TIMEOUT	1 (100 ms)
EXECUTE	TRUE

   - **D** Verify the following values with the function block output.

Table 21: Pin Name and Expected Value

Pin name	Expected value
RD_VALUE	200
FAULT_CODE	0

3. Write non–array parameters.
   - **A** Set the value *FALSE* to *EXECUTE* input pin of the function block.
   - **B** Set the following values to the input pin of the function block to write the value in the *parameter 3-41 Ramp 1 Ramp Up Time*.

Table 22: Pin Name and Set Value

Pin Name	Set value for *parameter 3-41 Ramp 1 Ramp Up Time*
RD_WR	TRUE
PAR_NO	341
INDEX	0
LENGTH	2
TIMEOUT	1 (100 ms)
WR_VALUE	500
EXECUTE	TRUE

   - **C** Verify *parameter 3-41 Ramp 1 Ramp Up Time* is *5.00 s* using the LCP.
4. Read array type parameters with index.
   - **A** Set the value *FALSE* to *EXECUTE* input pin of the function block.
   - **B** Set the value *90* to the *parameter 3-10 [2] Preset Reference* using the LCP.
   - **C** Set the following values to the input pin of the function block to read the array parameter *parameter 3-10 [2] Preset Reference*.

Table 23: Pin Name and Set Value

Pin name	Set value for *parameter 3-10 [2] Preset Reference*
RD_WR	FALSE
PAR_NO	310
INDEX	2
LENGTH	2
TIMEOUT	1 (100 ms)
EXECUTE	TRUE

D Verify the following values with the function block output.

Table 24: Pin Name and Expected Value

Pin name	Expected value
RD_VALUE	9000
FALUT_CODE	0

5. Write array type parameters with index.

A Set the value *FALSE* to *EXECUTE* input pin of the function block.

B Set the following values to the input pin of the function block to write the value in the non-array *parameter 3-10 [3] Preset Reference*.

Table 25: Pin Name and Set Value

Pin name	Set value for *parameter 3-10 [3] Preset Reference*
RD_WR	TRUE
PAR_NO	310
INDEX	3
LENGTH	2
TIMEOUT	1 (100 ms)
WR_VALUE	7000
EXECUTE	TRUE

C Verify on LCP that *parameter 3-10 [3] Preset Reference* is set to *70.00 s*.

6. Read string type parameters.

A Set the value *FALSE* to *EXECUTE* input pin of the function block.

B Set the string characters *DANFOSSDRIVE0123456789012* to the *parameter 0-37 Display Text 1* using LCP.

C Set the following values to the input pin of the function block.

Table 26: Pin Name and Set Value

Pin name	Set value for *parameter 0-37 Display Text 1*
RD_WR	FALSE
IS_STR	TRUE

Pin name	Set value for *parameter 0-37 Display Text 1*
PAR_NO	37
INDEX	0
LENGTH	13
TIMEOUT	2 (200 ms)
EXECUTE	TRUE

D Verify the following values with the function block output (string parameter reads maximum 25 characters).

Table 27: Pin Name and Expected Value

Pin name	Expected value
STR_RD_VALUE	*DANFOSSDRIVE0123456789012*
FALUT_CODE	0

7. Write string type parameters.

A Set the value *FALSE* to *EXECUTE* input pin of the function block.

B Set the following values to the input pin of the function block to write the value in *parameter 0-38 Display Text 2*.

Table 28: Pin Name and Set Value

Pin name	Set value for *parameter 0-38 Display Text 2*
RD_WR	TRUE
IS_STR	TRUE
PAR_NO	38
INDEX	0
LENGTH	13
TIMEOUT	2 (200 ms)
STR_WR_VALUE	*DANFOSSVLTDRIVE0123456789*
EXECUTE	TRUE

C Verify on LCP that *parameter 0-38 Display Text 2* is set to *DANFOSSVLTDRIVE0123456789* (string parameter writes maximum 25 characters).

3.4 Diagnostics Function Block

Context:

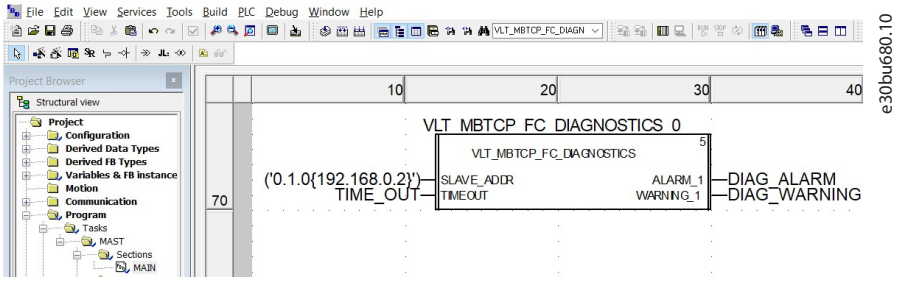


Illustration 75: Diagnostics Function Block

Procedure

1. To identify Modbus TCP Slave Address, refer to 2.7 Modbus TCP Slave Address.
2. Set the *parameter 8-07 Diagnosis Trigger* to *[1] Trigger on alarms* using LCP.
3. Generate *Live Zero Error Alarm (A2)*.
   - **A** Set the following values in the drive parameters for *Live Zero Error Alarm (A2)* using LCP.

Table 29: Live Zero Error Configuration Settings Using LCP

Parameter number	Modify value
Parameter 6-00 Live Zero Timeout Time	1 s
Parameter 6-01 Live Zero Timeout Function	[5] Stop and trip
Parameter 6–10 Terminal 53 Low Voltage	2.00 V

   - **B** Verify that *parameter 16-90 Alarm Word* remains *0x00000000* for 10 s using the LCP to ensure that no alarm is generated for 10 s.
   - **C** Verify the following values in the function block outputs.

Table 30: Pin Name and Expected Value

Pin name	Expected values
ALARM_1	16#00010000 in (HEX)
WARNING_1	16#00000000 in (HEX)

4. Remove *Live Zero Error Alarm (A2)*.
   - **A** Set *parameter 6-10 Terminal 53 Low Voltage* to *0.07 V* using the LCP.
   - **B** Press *Reset* to remove *Live Zero Error Alarm (A2)*.
   - **C** Verify the following values in the function block outputs.

Table 31: Pin Name and Expected Value

Pin name	Expected values
ALARM_1	16#00000000 in (HEX)
WARNING_1	16#00000000 in (HEX)

3.5 Flexible Control Function Block

Context:

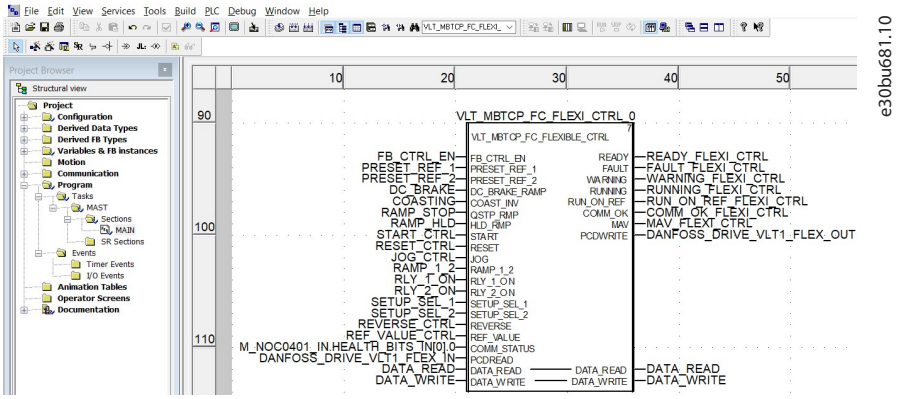


Illustration 76: Flexible Control Function Block

Procedure

1. Refer 2.4 Creating Process Variables for the Function Block Instance to create the process variable.
2. Map the *PCDREAD*, *PCDWRITE* & *COMM_STATUS* variables with the flexible control function block.

Table 32: Mapping Variables

Type	Pin name	Data type	Mapping
Input	PCDREAD	FLEXI_CTRL_VLT_IN_DDT	DANFOSS_DRIVE_VLT1_FLEXI_IN
Input	COMM_STATUS	BOOL	M_NOC0401_IN.HEALTH_BITS_IN[0].0
Output	PCDWRITE	FLEXI_CTRL_VLT_OUT_DDT	DANFOSS_DRIVE_VLT1_FLEXI_OUT

3. Enable the drive to *Ready* state.
   - **A** Verify the value *TRUE* is set to the *COMM_OK* output pin of the function block. It means the communication between PLC and the drive is OK.
   - **B** Set the following values to the function block input pins.

Table 33: Pin Name and Set Value

Pin name	Set value
FB_CTRL_EN	1
PRESET_REF_1	0
PRESET_REF_2	0
DC_BRAKE_RAMP	1
COAST_INV	1
QSTP_RMP	1
HLD_RMP	1
START	0
RESET	0
JOG	0
RAMP_1_2	0
RLY_1_ON	0
RLY_2_ON	0
SETUP_SEL_1	0
SETUP_SEL_2	0
REVERSE	0
REF_VALUE	0

   - **C** Verify the following output values of the function block.

Table 34: Pin Name and Expected Value

Pin name	Expected value
READY	1
FAULT	0

Pin name	Expected value
WARNING	0
RUNNING	0
RUN_ON_REF	0
COMM_OK	1
MAV	0

D Set the value *1* to the *START* input pin of the function block.

E Set the value *8192* to the *REF_VALUE* input pin of the function block.

F Verify the following output values of the function block.

Table 35: Pin Name and Expected Value

Pin name	Expected value
READY	1
FAULT	0
WARNING	0
RUNNING	1
RUN_ON_REF	1
COMM_OK	1
MAV	8192

G Set the value *0* to the *COAST_INV* input pin of the function block.

H Verify the following output values of the function block.

Table 36: Pin Name and Expected Value

Pin name	Expected value
READY	0
FAULT	0
WARNING	0
RUNNING	0
RUN_ON_REF	0
COMM_OK	1
MAV	0

I Set the value *0* to the *START* input pin of the function block.

4. Start the motor in forward direction.

A Set the value *1* to the *COAST_INV* input pin of the function block.

B Verify the following output values of the function block.

Table 37: Pin Name and Expected Value

Pin name	Expected value
READY	1
FAULT	0
WARNING	0
RUNNING	0
RUN_ON_REF	0
COMM_OK	1
MAV	0

C Set the value *1* to the *START* input pin of the function block.

D Set the value *12288* to the *REF_VALUE* input pin of the function block.

E Verify the following output values of the function block.

Table 38: Pin Name and Expected Value

Pin name	Expected value
READY	1
FAULT	0
WARNING	0
RUNNING	1
RUN_ON_REF	1
COMM_OK	1
MAV	12288

F Set the value *16384* to the *REF_VALUE* input pin of the function block.

G Verify the following output values of the function block.

Table 39: Pin Name and Expected Value

Pin name	Expected value
READY	1
FAULT	0
WARNING	0
RUNNING	1
RUN_ON_REF	1
COMM_OK	1
MAV	16384

5. Start the motor in reverse direction.

A Set the value *1* to the *REVERSE* input pin of the function block.

B Wait until the motor ramps down and runs in the reverse direction.

C Verify the following output values of the function block.

Table 40: Pin Name and Expected Value

Pin name	Expected value
READY	1
FAULT	0
WARNING	0
RUNNING	1
RUN_ON_REF	1
COMM_OK	1
MAV	-16384

D Set the value *0* to the *START* input pin of the function block.

E Set the value *0* to the *REVERSE* input pin of the function block.

F Verify the following output values of the function block.

Table 41: Pin Name and Expected Value

Pin name	Expected value
READY	1
FAULT	0
WARNING	0
RUNNING	0
RUN_ON_REF	0
COMM_OK	1
MAV	0

G Set the value *1* to the *START* input pin of the function block.

H Set the value *-16384* to the *REF_VALUE* input pin of the function block.

I Verify the following output values of the function block.

Table 42: Pin Name and Expected Value

Pin name	Expected value
READY	1
FAULT	0
WARNING	0
RUNNING	1
RUN_ON_REF	1
COMM_OK	1
MAV	-16384

6. Stop the motor.

A Set the value *0* to the *START* input pin of the function block.

B Set the value *0* to the *REF_VALUE* input pin of the function block.

C Verify the following output values of the function block.

Table 43: Pin Name and Expected Value

Pin name	Expected value
READY	1
FAULT	0
WARNING	0
RUNNING	0
RUN_ON_REF	0
COMM_OK	1
MAV	0

7. Set the parameter values for PCD read and PCD write on the drive.

Table 44: Parameter Settings

Parameter number	Parameter name	Parameter value
12–21 [2]	Process Data Config Write	3–12 Catch Up/Slow Down Value
12–21 [3]	Process Data Config Write	4–12 Motor Speed Low Limit [Hz]
12–21 [4]	Process Data Config Write	3–41 Ramp 1 Ramp Up Time
12–21 [5]	Process Data Config Write	3–41 Ramp 1 Ramp Up Time
12–21 [6]	Process Data Config Write	Depends on the drive configuration (default setting is *None*).
12–21 [7]	Process Data Config Write	Depends on the drive configuration (default setting is *None*).
12–21 [8]	Process Data Config Write	Depends on the drive configuration (default setting is *None*).
12–21 [9]	Process Data Config Write	Depends on the drive configuration (default setting is *None*).
12–22 [2]	Process Data Config Read	16–14 Motor Current
12–22 [3]	Process Data Config Read	16–14 Motor Current
12–22 [4]	Process Data Config Read	Depends on the drive configuration (default setting is *None*).
12–22 [5]	Process Data Config Read	Depends on the drive configuration (default setting is *None*).
12–22 [6]	Process Data Config Read	Depends on the drive configuration (default setting is *None*).
12–22 [7]	Process Data Config Read	Depends on the drive configuration (default setting is *None*).
12–22 [8]	Process Data Config Read	Depends on the drive configuration (default setting is *None*).
12–22 [9]	Process Data Config Read	Depends on the drive configuration (default setting is *None*).

8. Read process data.

A Verify the *DATA_READ* in-out parameter values.

Table 45: Parameter Values

Parameter name	***DATA_READ*** **variable in-out parameter name (animation table)**	Expected value
Process Data Config Read	DATA_READ⇒STW	Same as *parameter 16-03 Status Word* value.

Parameter name	*DATA_READ* variable in-out parameter name (animation table)	Expected value
Process Data Config Read	DATA_READ⇒MAV	Same as *parameter 16-82 Fieldbus REF 1* value.
Process Data Config Read	DATA_READ⇒PCD_02	Same as *parameter 16-14 Motor Current* value.
Process Data Config Read	DATA_READ⇒PCD_03	Same as *parameter 16-14 Motor Current* value.
Process Data Config Read	DATA_READ⇒PCD_04	Depends on the drive configuration.
Process Data Config Read	DATA_READ⇒PCD_05	Depends on the drive configuration.
Process Data Config Read	DATA_READ⇒PCD_06	Depends on the drive configuration.
Process Data Config Read	DATA_READ⇒PCD_07	Depends on the drive configuration.
Process Data Config Read	DATA_READ⇒PCD_08	Depends on the drive configuration.
Process Data Config Read	DATA_READ⇒PCD_09	Depends on the drive configuration.

9. Write process data.

A Set the value *80* to the *DATA_WRITE* variable in-out parameter⇒*DATA_WRITE*⇒*PCD_04*.

B Verify that the expected value of *parameter 3-41 Ramp 1 Ramp Up Time* is *8.00 s*.

Index

ENGINEERING
TOMORROW

Danfoss can accept no responsibility for possible errors in catalogues, brochures and other printed material. Danfoss reserves the right to alter its products without notice. This also applies to products already on order provided that such alterations can be made without subsequential changes being necessary in specifications already agreed. All trademarks in this material are property of the respective companies. Danfoss and the Danfoss logotype are trademarks of Danfoss A/S. All rights reserved.

Danfoss A/S
Ulsnaes 1
DK-6300 Graasten
vlt-drives.danfoss.com

Danfoss A/S © 2019.12 M0021801 AU326537836110en-000101/ 130R0925