

u-remote | u-control library for u-create studio

u-create studio package

Abstract:

This library package contains function blocks and functions for easy use of Weidmüller u-remote modules and the UC20-SL2000-xxx controller system in u-create studio. This document only explains the function blocks that are available to the END USER. Internal function blocks are not explained in more detail.

Hardware reference

No.	Component name	Article No.	Hardware / Firmware version
1	UR20-1COM-232-485-422 V1	1315750000	-
2	UR20-4DI-P	1315170000	-
3	UR20-4DI-P-3W	2009360000	-
4	UR20-4DI-P-3W	2009360000	-
5	UR20-8DI-P-2W	1315180000	-
6	UR20-8DI-P-3W	1394400000	-
7	UR20-8DI-P-3W-HD	1315190000	-
8	UR20-3EM-230V-AC	2007420000	-
9	UR20-4COM-IO-LINK	1315740000	-

Software reference

No.	Software name	Article No.	Software version
1	u-create studio	2660130000	1.20.2 or higher

File reference

No.	Name	Description	Version
1	-	-	-

Contact

Weidmüller Interface GmbH & Co. KG
 Klingenbergstraße 26
 32758 Detmold, Germany
www.weidmueller.com

For any further support please contact your
 local sales representative:
<https://www.weidmueller.com/countries>

Revision history

Library Version	Date	Change log	Author
1.0.0	2021-06	First release	w010521
1.0.0	2022-01	added libWiUr203EM_230V_AC	w010174
2.0.0	2022-04	update to libWiUr203EM_230V_AC	w010174
2.0.0	2022-03	added libWiUr204ComIOLink	w010174
2.1.0	2022-10	update for 2022-Q3 release	w010174

Content

1	Warning and Disclaimer.....	6
2	Standardized behavior of the function blocks.....	7
2.1	Function block variants	7
2.2	Basic states	7
2.3	Inputs and Outputs	7
2.4	Simplified behavior model.....	9
2.5	Usage of the function blocks.....	9
3	libWiUr20ComRs_232_485_422.....	11
3.1	FB_UR20_1ComRs_232_485_422_Control	11
3.2	FB_UR20_1ComRs_232_485_422_ParamEcCanRead	12
3.3	FB_UR20_1ComRs_232_485_422_ParamEcCanWrite	13
3.4	FB_UR20_1ComRs_232_485_422_ParamSysBusRead.....	14
3.5	FB_UR20_1ComRs_232_485_422_ParamSysBusWrite.....	15
4	libWiUr20ModbusRTUMaster	17
4.1	FB_ModbusRTUMaster	17
5	libWiModbusTCPClient.....	19
5.1	FB_ModbusTCPClient	19
6	libWiUr20Digital.....	21
6.1	FB_UR20_4DI_ParamEcCanRead.....	21
6.2	FB_UR20_4DI_ParamEcCanWrite	22
6.3	FB_UR20_4DI_ParamSysBusRead	23
6.4	FB_UR20_4DI_ParamSysBusWrite.....	24
6.5	FB_UR20_8DI_ParamEcCanRead.....	25
6.6	FB_UR20_8DI_ParamEcCanWrite	26
6.7	FB_UR20_8DI_ParamSysBusRead	27
6.8	FB_UR20_8DI_ParamSysBusWrite.....	28
7	libWiUr203EM_230V_AC.....	30
7.1	FB_UR20_3EM_Control	30
7.2	FB_UR20_3EM_ChannelParamEcCanWrite	34
7.3	FB_UR20_3EM_ChannelParamSysBusWrite.....	35
7.4	FB_UR20_3EM_ParamEcCanRead	36

7.5	FB_UR20_3EM_ParamEcCanWrite	37
7.6	FB_UR20_3EM_ParamSysBusRead.....	39
7.7	FB_UR20_3EM_ParamSysBusWrite.....	40
8	libWiUr20Diag.....	42
8.1	FB_UR20_Uremote_Diagdata_Ethercat.....	42
8.2	FB_UR20_Uremote_Diagdata_Sysbus.....	43
9	libWiUr204ComIOLink	45
9.1	FB_UR20_4COM_IO_LINK_EtherCATReadWrite.....	45
9.2	FB_UR20_4COM_IO_LINK_ModbusReadWrite.....	47

1 Warning and Disclaimer

Warning

Controls may fail in unsafe operating conditions, causing uncontrolled operation of the controlled devices. Such hazardous events can result in death and / or serious injury and / or property damage. Therefore, there must be provided safety equipment / electrical safety design or other redundant safety features that are independent from the automation system.

Disclaimer

This software (programs, function blocks, functions, etc.) does not relieve you of the obligation to handle it safely during use, installation, operation and maintenance. Every user is responsible for the correct operation of his control system.

By using this software prepared by Weidmüller, you accept that Weidmüller cannot be held liable for any damage to property and / or personal injury that may occur because of the use.

Note

This software does not represent customer-specific solutions, it is simply intended to help for typical tasks. The user is responsible for the proper operation of the used products. This software does not relieve you of the obligation of safe use, installation, operation and maintenance. This software is not binding and do not claim to be complete in terms of configuration as well as any contingencies.

By using this software, you acknowledge that Weidmueller cannot be held liable for any damages beyond the described liability regime. We reserve the right to make changes to this software at any time without notice.

We assume no liability for this software or the information contained in this document. Our liability, for whatever legal reason, for damages caused by the use of the software or instructions described in this document is excluded.

Security notes

In order to protect equipment, systems, machines and networks against cyber threats, it is necessary to implement (and maintain) a complete state-of-the-art industrial security concept. The customer is responsible for preventing unauthorized access to his equipment, systems, machines and networks. Systems, machines and components should only be connected to the corporate network or the Internet if necessary and appropriate safeguards (such as firewalls and network segmentation) have been taken.

2 Standardized behavior of the function blocks

All Weidmüller function blocks work in a defined manner. This behavior is defined by an internal state machine, which includes a set of standardized inputs and outputs. The following part describes the basic interfaces and behavior of the function blocks, as well as procedures to activate and deactivate the function blocks and how to handle errors.

2.1 Function block variants

There are in total four different variations of the underlying behavior.

- A level-controlled function block changes state by the level of the input signals (enable and execute)
- An edge-controlled function block changes state by evaluating the positive edges of its control inputs (enable, disable, execute, abort)

Both variants can either be implemented as a finite (with additional output "q_xDone") or as a continuous behavior. The finite behavior is used for any kind of action that comes to a defined end, while the continuous behavior is used for any action of an infinite nature.

2.2 Basic states

There are four basic states of a function block:

- **idle**: no action is performed (initial state of each function block).
- **standby**: the function block is initialized and ready to be executed
- **active**: the function block is performing its intended task
- **error**: an error occurred, and the function block had to be stopped.

The states **standby** and **active** are separated into different sub-states, since the function block may take some time to be initialized (entering state **standby**) or to safely shutdown an action (state **active**).

2.3 Inputs and Outputs

The inputs for the level-triggered behavior:

Input	Description
i_xEnable	Set true to enable the FB and start the initialization. The FB is initialized and ready (standby) if q_xStandby is true and q_xBusy is false . If set to false , all actions are stopped immediately, the FB is disabled and possible errors are reset.
i_xExecute	When the FB is initialized and ready (standby), setting i_xExecute to true starts the intended task and the FB becomes active. The output q_xActive indicates that the FB is being executed. If the FB implements a finite behavior, q_xDone indicates that the task is completed. If set to false , the active FB is stopped in a controlled way and the FB switches back to standby state. The output q_xBusy is true during shutdown.

The inputs for edge-triggered behavior:

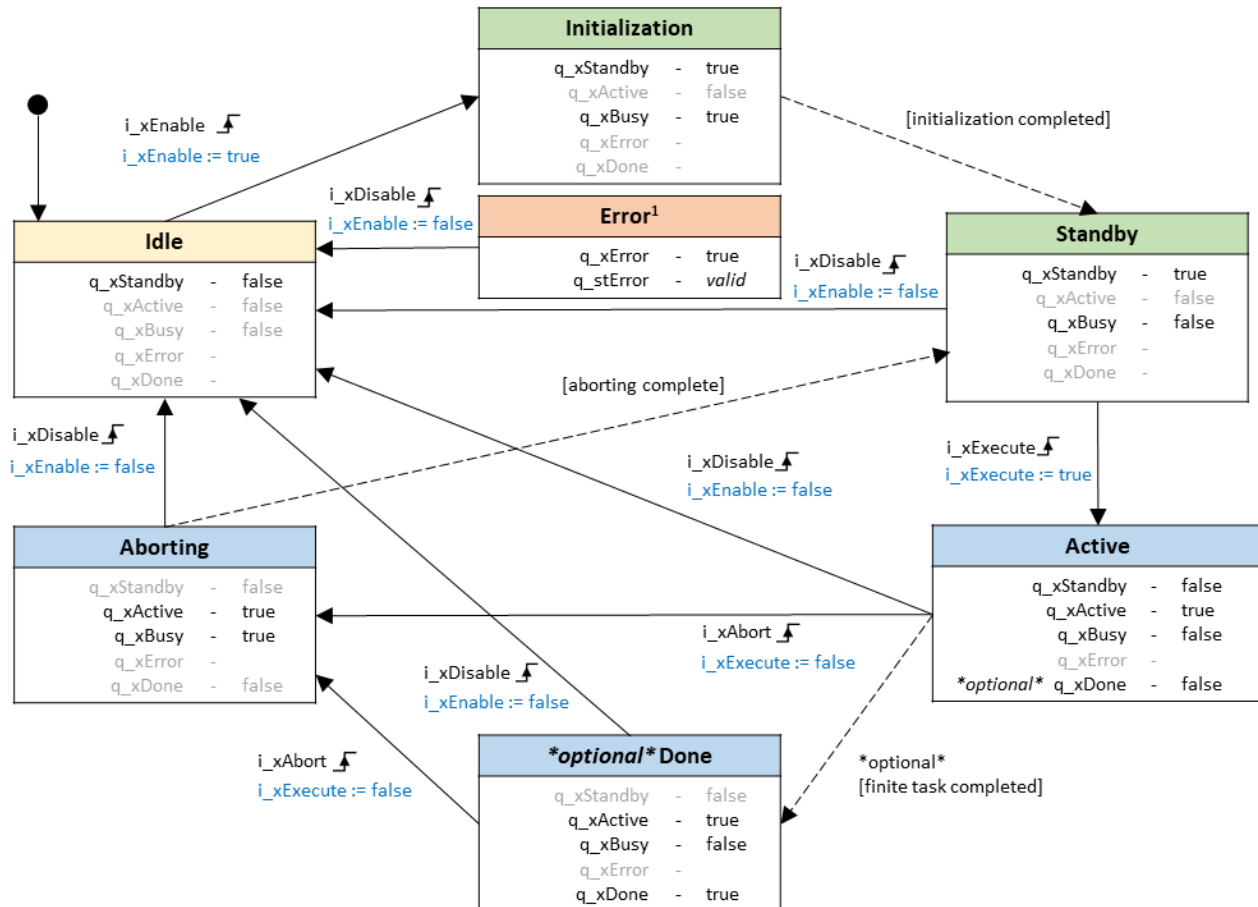
Input	Description
i_xEnable	A rising edge enables the FB and starts the initialization. The FB is initialized and ready (standby) if q_xStandby is true and q_xBusy is false .
i_xExecute	When the FB is initialized and ready (standby), a rising edge starts the intended task and the FB becomes active. The output q_xActive indicates that the FB is being executed. If the FB implements a finite behavior, q_xDone indicates that the task is completed.
i_xAbort	When the FB is active (q_xActive is true), a rising edge stops the FB in a controlled way. The output q_xBusy is true during shutdown. The FB switches back to standby .
i_xDisable	A rising edge disables the FB. All actions are stopped immediately and possible errors are reset.

Outputs:

Output	Description
q_xStandby	If true , the FB is either initializing, which may take multiple cycles (output q_xBusy is true), or already initialized and ready to be started (output q_xBusy is false)
q_xActive	If true , the FB is active. The FB is currently performing the desired task (q_xDone and q_xBusy are both false), or has already completed the desired, finite task (q_xDone is true and q_xBusy is false), or is currently shutting down an ongoing task in a controlled manner (q_xBusy is true)
q_xBusy	If true , the FB performs some internal tasks (shutdown or initialization) which may take multiple cycles. Wait until q_xBusy is false . Disabling (via inputs i_xEnable or i_xDisable) is always possible, but bypassing the usual shutdown sequence might result in undesired behaviour (depending on the specific function block).
q_xError	If true , an error has occurred and the FB is stopped.
q_stError	A struct that contains detailed information in case of an error (if q_xError is true). Collecting relevant error details may take some time, so the information may not be available immediately after q_xError indicates an error. Therefore, the output q_stError.xErrorDataValid is set to true as soon as all information is available in q_stError. Prior to that, q_stError might contain outdated or incorrect information.
q_xDone	(Finite behavior only) If true , a finite task has been completed. Reset and back to Standby state (q_xStandby) by disabling i_xExecute.

2.4 Simplified behavior model

Combined view illustrating both level-triggered and edge-triggered behavior is shown on the figure. Blue labels at transitions with inputs correspond to level-triggered behavior¹.



2.5 Usage of the function blocks

► Activation of the function block

The activation of an FB is controlled by the two boolean inputs of the function block, `i_xEnable` and `i_xExecute`. The activation process is split into two procedures and performed in the same way for every type of behavior:

1. Initialize the function block: switch from state **idle** to **standby**. By setting `i_xEnable`, parameters are validated, and the block is initialized. The parameter validation takes one single PLC cycle and results either into a start-up routine or an error. If needed, the start-up routine may perform some initial actions, which may take multiple PLC cycles and may also include interaction with devices, parameterization, communication, etc. After the input `i_xEnable` has been set, the output `q_xStandby` changes from **false** to **true**.

¹ The state "Error" is reached from any other state in case of a general error. For reasons of simplicity, the corresponding transitions are not shown in the diagrams.

As long as the function block performs its initialization routine, the output `q_xBusy` stays **true**. After the initialization is finished, `q_xBusy` changes back to **false**, which indicates the function block can be executed now.

2. Execute the function block and perform its intended task: switch from state **standby** to state **active**. Once the output `q_xStandby` indicates **true** and `q_xBusy` has been reset to **false**, the main operation can be started. This is always done by setting the input `i_xExecute` to **true**. The next step (internally performed) is to check whether the current process values are ok. If they are not ok, the activation results in an **error** state. If they are ok, the function block starts with its intended action. While this action is performed, the output `q_xActive` is **true**. After a function block that implements a finite behavior has performed its action completely, the output `q_xDone` is set to **true**. A continuous type function block stays in state **active** as long as no disable command is given.

► Deactivation of the function block

Once the intended action of the function block has been done (or an ongoing action needs to be aborted), the function block needs to be deactivated. The procedure differs with regard to the implemented behaviour of the function block:

- for a level-controlled function block, the input `i_xExecute` needs to be set to **false**.
- for an edge-controlled function block, a positive edge on the input `i_xAbort` is required.

Once the deactivation command has been received, the function block will change to state **standby**. In some cases, a shutdown routine is implemented and while this routine is executed, the output `q_xBusy` becomes **true** to indicate that the FB is shutting down. After the shutdown procedure has been finished, the outputs `q_xActive` and `q_xBusy` (and possibly `q_xDone`) will change to **false**, and `q_xStandby` will become **true** to indicate that the function block is now again in state **standby**.

A deactivation can also be achieved by

- setting the `i_xEnable` to **false** (in case of a level-controlled function block), or
 - providing a rising edge on the input `i_xDisable` (in case of an edge-controlled function block).
- This way of deactivating the function block sets it back to its idle state. All outputs are set to **false** and possible errors are reset. This bypasses the usual shutdown sequence and might result in undesired behaviour (depending on the specific function block). For reactivation, the complete activation procedure is required.

► Detect and reset an error

In some cases, errors might occur, and the function block will switch into an **error** state. During this state, the error information is generated and provided via the `q_stError` output. The process of collecting error information is finished once `q_stError.xErrorDataValid` becomes **true**. While this value is **false**, any information contained in `q_stError` is not valid and should not be processed, even if `q_xError` already indicates that an error has occurred. To reset the function block after an error, it needs to be disabled (`i_xEnable` = **false** or positive edge on `i_xDisable`).

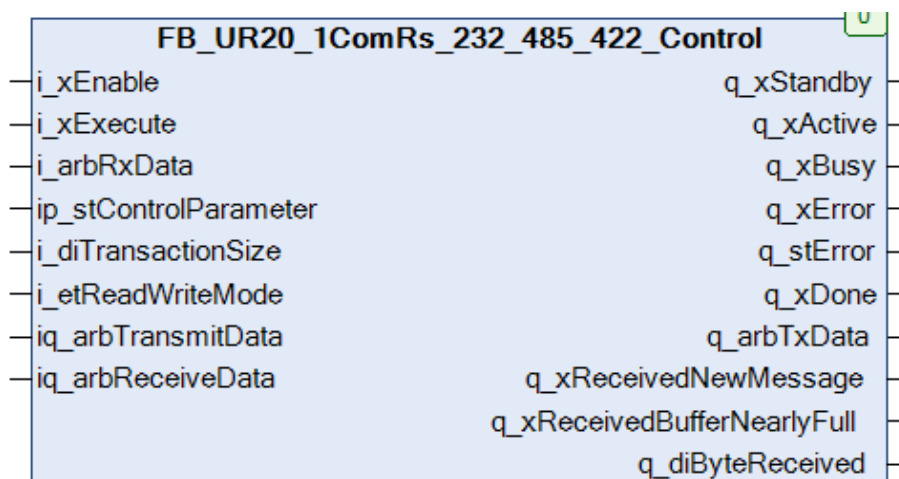
3 libWiUr20ComRs_232_485_422

3.1 FB_UR20_1ComRs_232_485_422_Control

The function block provides the possibility to send and receive data either on *RS232*, *RS485* or *RS422* via UR20-1COM-232-485-422 module.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior Model.

The parametrization of the module must be done by hardware configuration in the PLC project, via web interface or by another function block provided in this library.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	i_arbRxData	ARRAY [0..15] OF USINT	connection to hardware / process data input
	ip_stControlParameter	ST_UR20_1ComRs_232_485_422_Control Param	contains parameter for operating the module
	i_diTransactionSize	DINT	amount of bytes/segments to be send or read
	i_etReadWriteMode	ET_UR20_1ComRs_232_485_422_Read WriteMode	defines the type of operation
Input	iq_arbTransmitData	POINTER TO BYTE	input for all data to be send
	iq_arbReceiveData	POINTER TO BYTE	output of all the data to be read
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task

Scope	Name	Type	Comment
	q_xError	BOOL	function block in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end
Output	q_arbTxData	ARRAY [0..15] OF USINT	connection to hardware / process data output
	q_xReceivedNewMessage	BOOL	status from module, new data available
	q_xReceivedBufferNearlyFull	BOOL	status from module, only 10 bytes left in buffer
	q_diByteReceived	DINT	number of bytes received during read command

Possible Errors

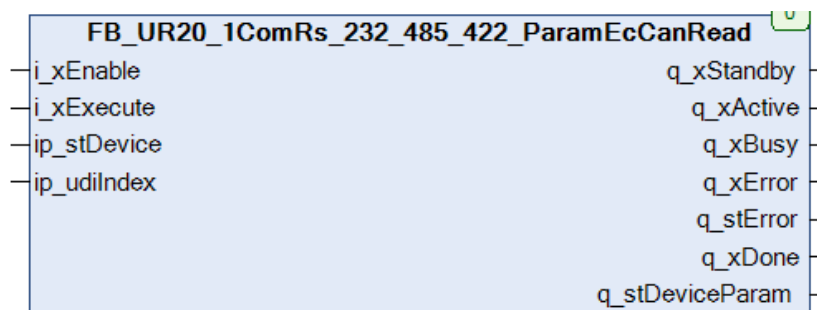
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Communication error	Communication error detected during operation
Frame too long during read	A frame that was received by the module is too long for being read into the output array of the function block
Frame too long during write	A frame that shall be send is too long for the input buffer of the module
Transmission watchdog exceeded	The module did not respond within a specific time during a transmission
Invalid parameter	Watchdog time is too short (> 0)
Watchdog for buffer flush expired	The time to flush the buffer was too long.

3.2 FB_UR20_1ComRs_232_485_422_ParamEcCanRead

The function block reads the current parameter values from a u-remote module connected to an EtherCAT or CANopen fieldbus coupler.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior Model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IO_ECAT_COE_DEVICE_REF	handle to the CANopen or EtherCAT device [coupler]
	ip_udiIndex	UDINT	start address of the Object directory of the module
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end
Output	q_stDeviceParam	ST_UR20_1ComRs_232_485_422_DeviceParam	parameter set that was read

Possible Errors

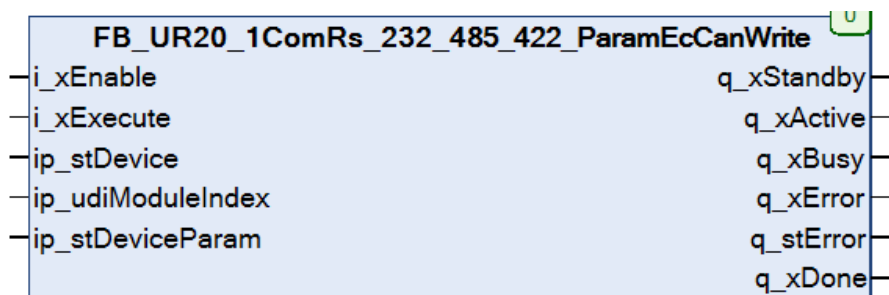
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Reading Ethercat Index Parameter	Problem triggered by communication level function block.
Connection lost during action	Connection error while reading the parameters

3.3 FB_UR20_1ComRs_232_485_422_ParamEcCanWrite

The function block writes the parameter to a u-remote module connected to an EtherCAT or CANopen fieldbus coupler.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior Model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IO_ECAT_COE_DEVICE_REF	handle to the CANopen or EtherCAT device
	ip_udiModuleIndex	UDINT	start address of the object directory of the module
	ip_stDeviceParam	ST_UR20_1ComRs_232_485_422_DeviceParam	parameter set to be written
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end

Possible Errors

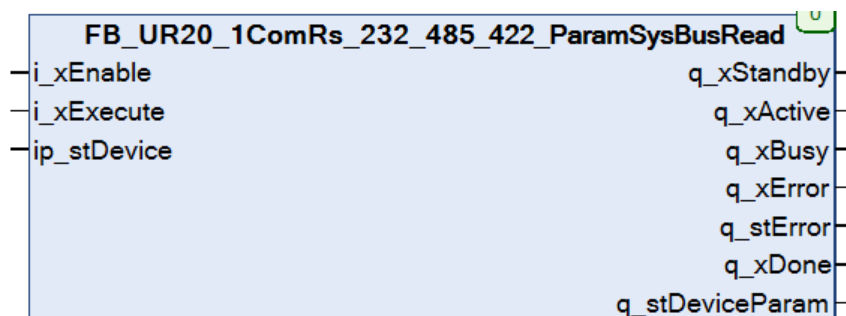
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Writing Ethercat Index Parameter	Problem triggered by communication level function block.
Connection lost during action	Connection error while writing the parameters

3.4 FB_UR20_1ComRs_232_485_422_ParamSysBusRead

The function block reads the parameter from a u-remote module connected directly to the system bus (backplane bus) of the controller.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior Model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IIO	handle to the module
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end
Output	q_stDeviceParam	ST_UR20_1ComRs_232_485_422_DeviceParam	parameter set that was read

Possible Errors

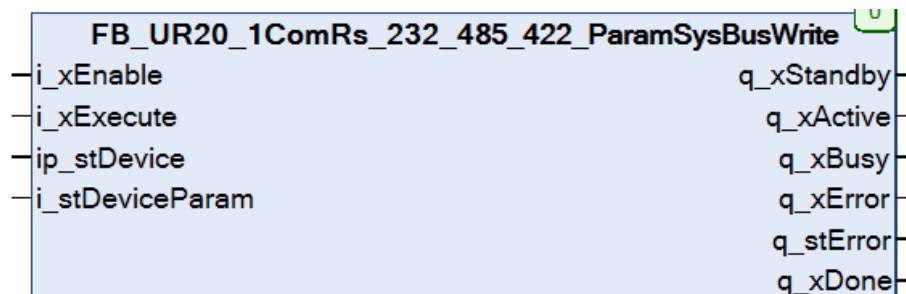
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	input device could not be found during transition to state active.
Reading Ethercat Index Parameter	Problem triggered by communication level function block.
Connection lost during action	Connection error while reading the parameters

3.5 FB_UR20_1ComRs_232_485_422_ParamSysBusWrite

The function block writes the parameter to a u-remote module connected via system bus (backplane bus).

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior Model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IIO	handle to the module
	i_stDeviceParam	ST_UR20_1ComRs_232_485_422_DeviceParam	parameter set to be written
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end

Possible Errors

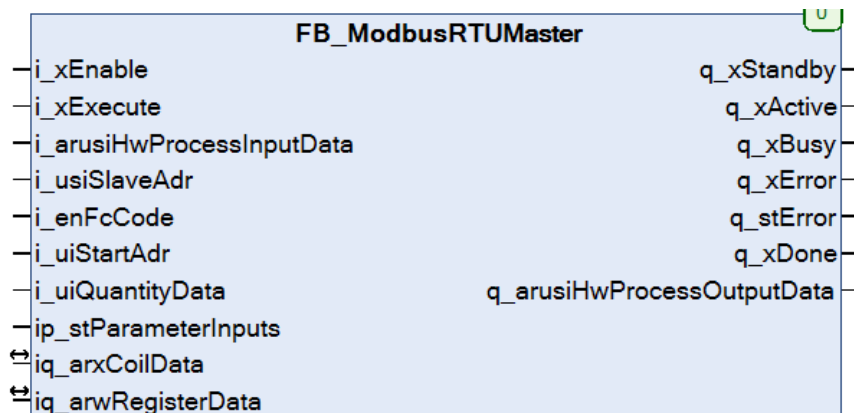
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Writing Ethercat Index Parameter	Problem triggered by communication level function block.
Connection lost during action	Connection error while writing the parameters

4 libWlUr20ModbusRTUMaster

4.1 FB_ModbusRTUMaster

The function block implements a Modbus RTU Master that provides communication with a Modbus slave via the UR20-1COM-232-485-422 u-remote module.



Inputs and Outputs

Scope	Name	Type	Initial	Comment
Input	i_xEnable	BOOL		enables the function block by switching from idle to standby
	i_xExecute	BOOL		activates the function block by switching from standby to active
Input	i_arusiHwProcessInputData	ARRAY [0..15] OF USINT		array of process inputs from UR20-1Com Module
	i_usiSlaveAdr	USINT	1	modbus slave address 1-247
	i_enFcCode	ET_UR20_ModbusRTU_1ComRs_Function Code		modbus function code
	i_uiStartAdr	UINT	0	starting address of modbus register 0-65535
	i_uiQuantityData	UINT	1	quantity of data 1-2000 coils or 1-125 registers
	ip_stParameterInputs	ST_UR20_ModbusRTU_1ComRs_ControlParameter		struct contains control parameter
Inout	iq_arxCoilData	ARRAY [1..2000] OF BOOL		read and send buffer for bit-oriented Modbus coils and discrete inputs
	iq_arwRegisterData	ARRAY [1..125] OF WORD		read and send buffer for word-oriented Modbus registers
Output	q_xStandby	BOOL		waiting for activation
	q_xActive	BOOL		function block is activated
	q_xBusy	BOOL		function block is activated and doing its supposed task

Scope	Name	Type	Initial	Comment
	q_xError	BOOL		function block is in error state
	q_stError	ST_ErrorInfo		detailed error information
Output	q_xDone	BOOL		execution has come to its supposed end
Output	q_arusiHwProcessOutputData	ARRAY [0..15] OF USINT		array of process outputs to UR20-1Com Module

Supported Modbus Function Codes

Function Code	Register Type	Explanation
1	Read Coil Status	Reads binary outputs (coils) from a connected slave. The data is stored in Array iq_arxCoilData[1..2000] of BOOL.
2	Read Input Status	Reads binary inputs from a connected slave. The data is stored in Array iq_arxCoilData[1..2000] of BOOL.
3	Read Holding Registers	Reads register-data from a connected slave. The data is stored in Array iq_arwRegisterData[1..125] of WORD.
4	Read Input Registers	Reads input registers from a connected slave. The data is stored in Array iq_arwRegisterData[1..125] of WORD.
5	Write Single Coil	Sends a binary output (Coil) to a connected slave. The value from array iq_arxCoilData [1] is written to the slave.
6	Write Single Register	Sends a single data word to a connected slave. The value from array iq_arwRegisterData[1] is written to the slave.
8	Diagnostics	Sends a diagnostics request with a user defined subfunction code to a connected slave. The subfunction code is passed to the function by the parameter i_uiStartAdr. Additional data can be passed by Array iq_arwRegisterData[1].
15	Write Multiple Coils	Sends several binary outputs (Coils) to a connected slave. The data must be provided in array iq_arxCoilData[1..2000] of BOOL. The maximum number of coils are 0x07B0.
16	Preset Multiple Registers	Sends data to a connected slave. The data must be provided in array arwRegisterData[1..125] of WORD. The maximum number of data are 0x007B.

Possible Errors

Possible error messages on FB output q_stError:

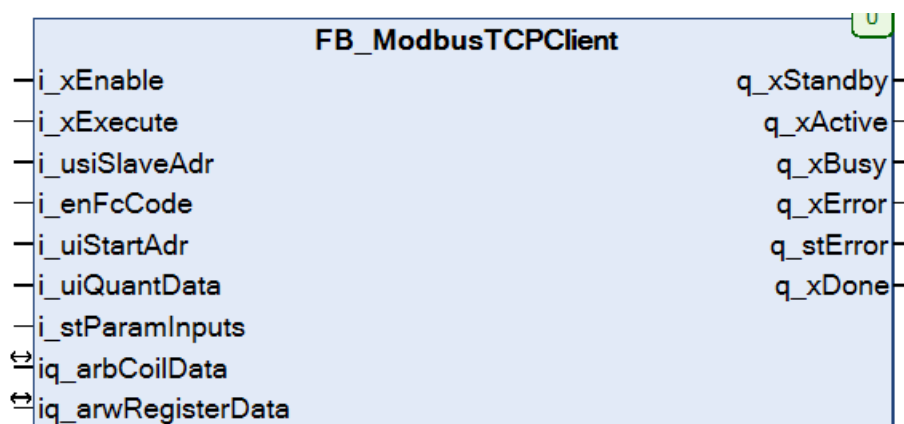
Error interface (Reason)	Description
Invalid parameter	The FB input parameter must be >=0
Invalid process value	Please check the FB input value and range
Timeout 1Com-Driver	Sub-FB doesn't return an answer, please check communication settings with MB-Slave
Invalid Response	Modbus Slave returns an invalid value
Slave Error Response	Modbus Slave returns an Error code

5 libWIModbusTCPClient

5.1 FB_ModbusTCPClient

The function block implements a Modbus TCP client that provides communication with a Modbus TCP server.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior Model.



Inputs and Outputs

Scope	Name	Type	Initial	Comment
Input	i_xEnable	BOOL		enables the function block by switching from idle to standby
	i_xExecute	BOOL		activates the function block by switching from standby to active
Input	i_usiSlaveAdr	USINT	1	modbus slave address 1-247
	i_enFcCode	ET_ModbusTCP_FunctionCode		modbus function code
	i_uiStartAdr	UINT	0	starting address of modbus register 0-65535
	i_uiQuantData	UINT	1	quantity of data 1-250 coils or 1-125 registers
	i_stParamInputs	ST_ModbusTCP_ControlParam		input parameter
Inout	iq_arbCoilData	ARRAY [1..250] OF BYTE		read and send buffer for bit-oriented Modbus coils and discrete inputs
	iq_arwRegisterData	ARRAY [1..125] OF WORD		read and send buffer for word-oriented Modbus registers
Output	q_xStandby	BOOL		waiting for activation
	q_xActive	BOOL		function block is activated
	q_xBusy	BOOL		function block is activated and doing its supposed task
	q_xError	BOOL		function block is in error state

Scope	Name	Type	Initial	Comment
	q_stError	ST_ErrorInfo		detailed error information
Output	q_xDone	BOOL		execution has come to its supposed end

Supported Functions

Function Code	Register Type	Explanation
1	Read Coil Status	Reads Bit-data from a connected slave. The data is stored in Array arbCoilData[1..250] of Byte. Each coil is read into a separate BYTE!
2	Read Input Status	Reads Bit-data from a connected slave. The data is stored in Array arbCoilData[1..250] of Byte. Each coil is read into a separate BYTE!
3	Read Holding Registers	Reads register-data from a connected slave. The data is stored in Array arwRegisterData[1..125] of WORD.
4	Read Input Registers	Reads input registers from a connected slave. The data is stored in Array arwRegisterData[1..125] of WORD.
5	Write Single Coil	Sends a binary output (Coil) to a connected slave. The value from array arbCoilData[1] is written to the slave.
6	Write Single Register	Sends a single data word to a connected slave. The value from array arwRegisterData[1] is written to the slave.
15	Write Multiple Coils	Sends several binary outputs (Coils) to a connected slave. The data must be provided in array arbCoilData[1..250] of Byte. Value for each coil is stored in a separate BYTE!
16	Preset Multiple Registers	Sends data to a connected slave. The data must be provided in array arwRegisterData[1..125] of WORD. The maximum number of data are 0x007B.

Possible Errors

Possible error messages on FB output q_stError:

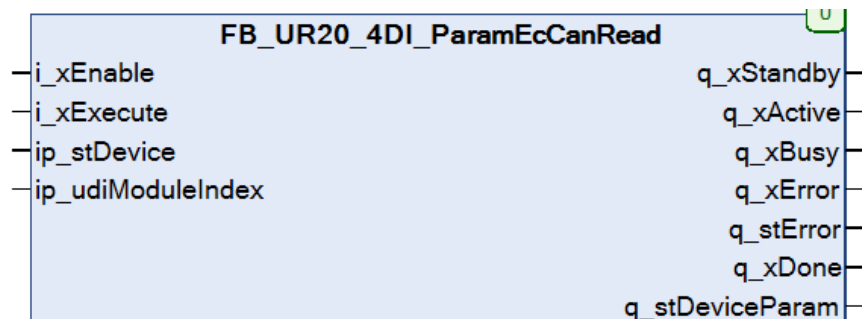
Error interface (Reason)	Description
Invalid parameter	The FB input parameter must be ≥ 0
Invalid process value	Please check the FB input value and range
Timeout MB Request	Sub-FB doesn't return an answer, please check communication settings with MB-Slave
Invalid Response	Modbus Slave returns an invalid value
Slave Error Response	Modbus Slave returns an Error code

6 libWlUr20Digital

6.1 FB_UR20_4DI_ParamEcCanRead

The function block reads the parameter from a u-remote module connected to an EtherCAT or CANopen fieldbus coupler. The function block can be used with the following u-remote Modules: UR20-4DI-P, UR20-4DI-3W, UR20-4DI-N.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior Model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IO_ECAT_COE_DEVICE_REF	handle to the CANopen or EtherCAT device [coupler]
	ip_udiModuleIndex	UDINT	EC/CanOpen Index of the single u-remote module
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end
Output	q_stDeviceParam	ST_UR20_4DI_DeviceParam	parameter set from u-remote module

Possible Errors

Possible error messages on FB output q_stError

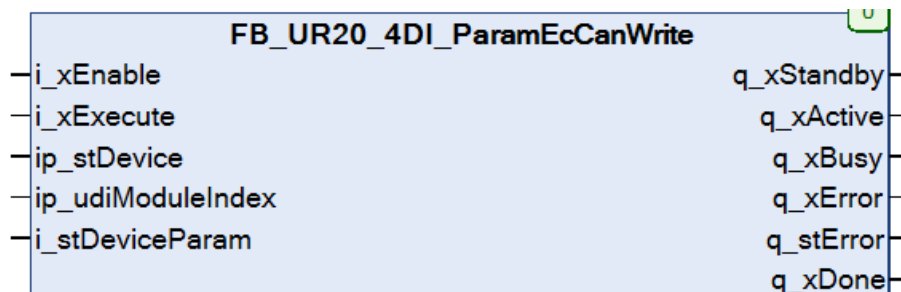
Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Reading Ethercat Index Parameter	Problem triggered by communication levelfunction block.

Error interface (Reason)	Description
Connection lost during action	Connection error while reading the parameters

6.2 FB_UR20_4DI_ParamEcCanWrite

The function block writes the parameter to a u-remote module connected to an EtherCAT or CANopen fieldbus coupler. The function block can be used with the following u-remote modules: UR20-4DI-P, UR20-4DI-3W, UR20-4DI-N.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	<code>i_xEnable</code>	BOOL	enables the function block by switching from idle to standby
	<code>i_xExecute</code>	BOOL	activates the function block by switching from standby to active
Input	<code>ip_stDevice</code>	IO_ECAT_COE_DEVICE_REF	handle to the CanOpen or EtherCAT device [coupler]
	<code>ip_udiModuleIndex</code>	UDINT	EC/CanOpen Index of the single u-remote module
	<code>i_stDeviceParam</code>	ST_UR20_4DI_DeviceParam	parameter set to be written
Output	<code>q_xStandby</code>	BOOL	waiting for activation
	<code>q_xActive</code>	BOOL	function block is activated
	<code>q_xBusy</code>	BOOL	function block is activated and doing its supposed task
	<code>q_xError</code>	BOOL	function block is in error state
	<code>q_stError</code>	ST_ErrorInfo	detailed error information
Output	<code>q_xDone</code>	BOOL	execution has come to its supposed end

Possible Errors

Possible error messages on FB output `q_stError`

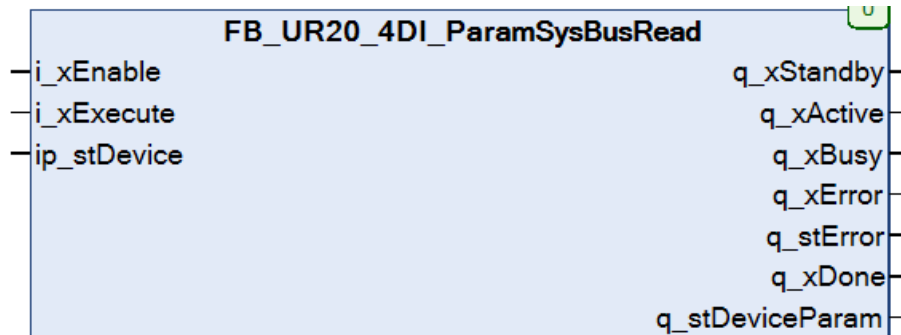
Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Writing Ethercat Index Parameter	Problem triggered by communication level function block.

Error interface (Reason)	Description
Connection lost during action	Connection error while writing the parameters

6.3 FB_UR20_4DI_ParamSysBusRead

The function block reads the parameter from a u-remote module connected over the local bus (SysBus). The function block can be used with the following u-remote modules: UR20-4DI-P, UR20-4DI-3W, UR20-4DI-N.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IO_SYSBUS_DEVICE_REF	handle to the u-remote module
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end
Output	q_stDeviceParam	ST_UR20_4DI_DeviceParam	parameter set from u-remote module

Possible Errors

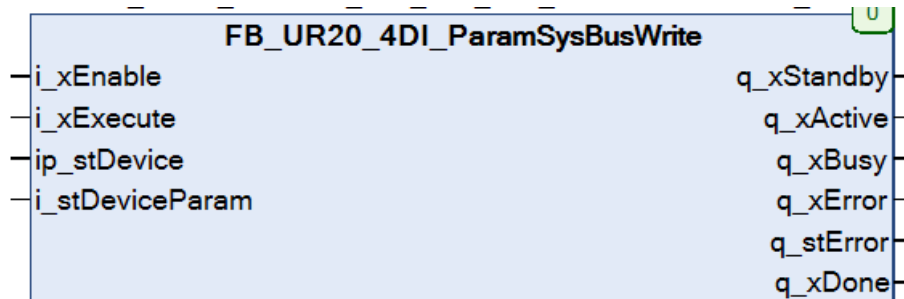
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Reading SysBus Parameter	Problem triggered by communication levelfunction block.
Connection lost during action	Connection error while writing the parameters

6.4 FB_UR20_4DI_ParamSysBusWrite

The function block writes the parameter to a u-remote module via local bus (SysBus). The function block can be used in with the following u-remote modules: UR20-4DI-P, UR20-4DI-3W, UR20-4DI-N.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IO_SYSBUS_DEVICE_REF	handle to the module
	i_stDeviceParam	ST_UR20_4DI_DeviceParam	parameter set to be written
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	funcion block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end

Possible Errors

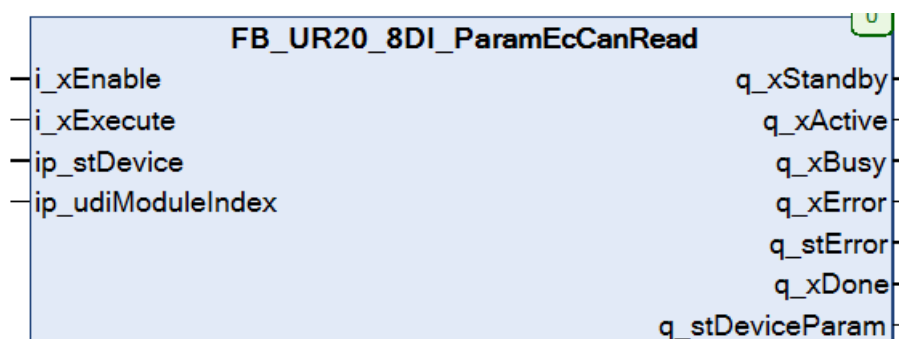
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Writing SysBus Parameter	Problem triggered by communication levelfunction block.
Connection lost during action	Connection error while writing the parameters

6.5 FB_UR20_8DI_ParamEcCanRead

The function block reads the parameter from a u-remote module connected to an EtherCAT or CANopen fieldbus coupler. The function block can be used with the following u-remote modules: UR20_8DI_P_2W, UR20_8DI_P_3W, UR20_8DI_P_3W_HD, UR20_8DI_N_3W, UR20_8DI_ISO_2W.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior Model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IO_ECAT_COE_DEVICE_REF	handle to the CanOpen or EtherCAT device [coupler]
	ip_udiModuleIndex	UDINT	start address of the module
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end

Scope	Name	Type	Comment
Output	q_stDeviceParam	ST_UR20_8DI_DeviceParam	parameter set that was read

Possible Errors

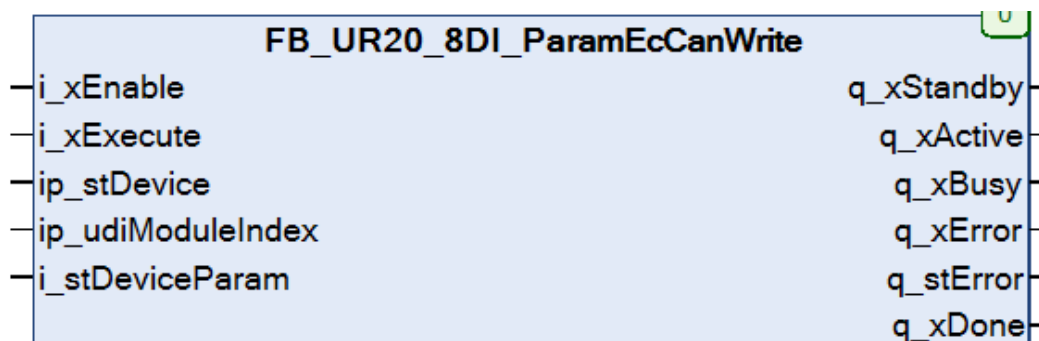
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Reading Ethercat Index Parameter	Problem triggered by communication level function block.
Connection lost during action	Connection error while reading the parameters

6.6 FB_UR20_8DI_ParamEcCanWrite

The function block writes the parameter to a u-remote module connected to an EtherCAT or CANopen fieldbus coupler. The function block can be used with the following u-remote modules: UR20_8DI_P_2W, UR20_8DI_P_3W, UR20_8DI_P_3W_HD, UR20_8DI_N_3W, UR20_8DI_ISO_2W.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IO_ECOT_COE_DEVICE_REF	handle to the CanOpen or EtherCAT device [coupler]
	ip_udiModuleIndex	UDINT	start address of the module
	i_stDeviceParam	ST_UR20_8DI_DeviceParam	parameter set to be written
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task

Scope	Name	Type	Comment
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end

Possible Errors

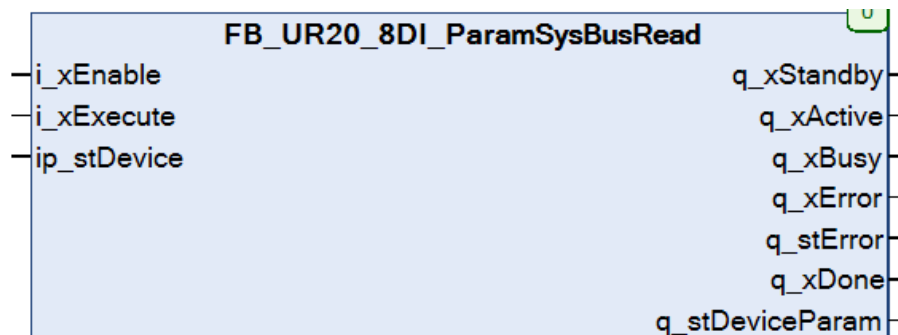
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Writing Ethercat Index Parameter	Problem triggered by communication level function block.
Connection lost during action	Connection error while writing the parameters

6.7 FB_UR20_8DI_ParamSysBusRead

The function block reads the parameter from a u-remote module connected over the local bus (SysBus). The function block can be used with the following u-remote modules: UR20_8DI_P_2W, UR20_8DI_P_3W, UR20_8DI_P_3W_HD, UR20_8DI_N_3W, UR20_8DI_ISO_2W.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IO_SYSBUS_DEVICE_REF	handle to the module
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information

Scope	Name	Type	Comment
Output	q_xDone	BOOL	execution has come to its supposed end
Output	q_stDeviceParam	ST_UR20_8DI_DeviceParam	parameter set that was read

Possible Errors

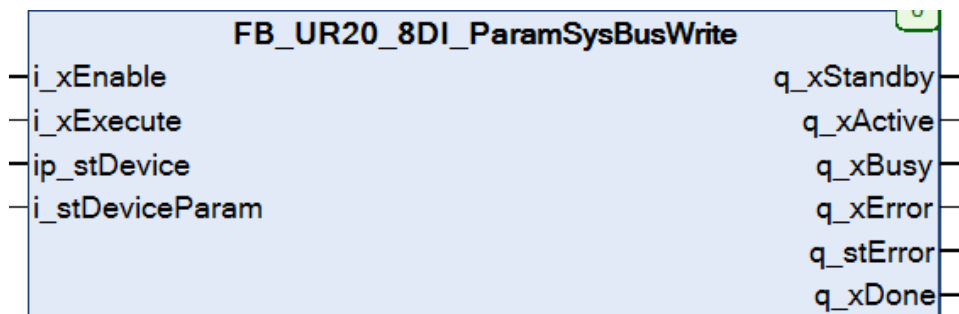
Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires mandatory an input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Reading SysBus Parameter	Problem triggered by communication level function block.
Connection lost during action	Connection error while writing the parameters

6.8 FB_UR20_8DI_ParamSysBusWrite

The function block writes the parameter to a u-remote module via local bus (SysBus). The function block can be used with the following u-remote modules: UR20_8DI_P_2W, UR20_8DI_P_3W, UR20_8DI_P_3W_HD, UR20_8DI_N_3W, UR20_8DI_ISO_2W.

The behavior of the function block bases on the FB_LevelControlledFiniteBehavior model.



Inputs and Outputs

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Input	ip_stDevice	IO_SYSBUS_DEVICE_REF	handle to the module
	i_stDeviceParam	ST_UR20_8DI_DeviceParam	parameter set to be written
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state

Scope	Name	Type	Comment
	q_stError	ST_ErrorInfo	detailed error information
Output	q_xDone	BOOL	execution has come to its supposed end

Possible Errors

Possible error messages on FB output q_stError

Error interface (Reason)	Description
Invalid parameter "ip_stDevice"	The parameter requires a mandatory input
Invalid process "ip_stDevice"	Input device could not be found during transition to state active.
Writing SysBus Parameter	Problem triggered by communication levelfunction block.
Connection lost during action	Connection error while writing the parameters

7 libWiUr203EM_230V_AC

Weidmueller Interface library for parametrization and control of a UR20_3EM_230V_AC energy meter module. Weidmueller Interface strongly recommends that you read the chapter related to the 3EM module in the UR20 manual available on Weidmueller Interface's website before using this library. Further library documentation will abbreviate 'Weidmueller Interface' to 'WI'.

7.1 FB_UR20_3EM_Control

Functional Description

This is the UR20-3EM_230V_AC control function block. FB_UR20_3EM_Control has level-controlled standard behavior. It transfers measurements from the 3EM module to the user's application and resets the 3EM's energy counters on request by the user. The 3EM module provides up to 56 different measurement values related to the electrical energy metered by the 3EM module. You may freely select which and how many measurement values you want this function block to collect for you:

- determine the number of measurement values and parametrize **ip_st3EMParams.uiNumberOfMeasurementValues** accordingly
- if you use current transformers, parametrize their ratio in **ip_st3EMParams.IrCurrentTransformerRatio**
- select the measurement values in **ip_st3EMDeviceParams.aretMeasurementsToChannels**
- use one of the **FB_UR20_3EMParam<field bus>Write** function blocks provided in this library to set the UR20_3EM_230V_AC module's device parameters or
- set the UR20_3EM_230V_AC module's device parameters via the u-create studio project

The 3EM module knows 56 measurement *values* but has only eight *channels* for the transfer of measurement values to the PLC. Therefore, the FB_UR20_3EM_Control has two modes of operation:

1. Straight through: The function block uses this mode of operation if you select eight or less measurement values. In this mode of operation, the function block simply converts the values provided via the 3EM's eight channels to SI units and puts them into **q_arIrMeasurements**.



In this mode, the FB_UR20_3EM_Control writes your measurement value choice in **ip_st3EMDeviceParams.aretMeasurementsToChannels** to the UR20_3EM_230V_AC module *once* after it has entered the active state.

2. Multiplex: The function block uses this mode of operation if you select more than eight measurement values. In this mode of operation, the function block runs through the following steps cyclicly:

- parametrize eight measurement values
- wait **ip_st3EMParams.tDataTransfer** for the 3EM to send the associated measurement values to the PLC.

- c. convert the associated measurement values to SI units and put them into **q_arIrrMeasurements**
- d. repeat the above with the next up to eight measurement values (or go back to the first eight measurement values when no more are left.)



E.g. for a module directly attached to the UC20..OLAC, the above cycle takes four PLC cycles per measurement value plus **ip_st3EMParams.tDataTransfer** plus one PLC cycle. For example, 20 measurement values require 81 PLC cycles plus **ip_st3EMParams.tDataTransfer**. The timing depends on the choice of fieldbus via which the UR20_3EM_230V_AC communicates with the PLC. Since the function block collects the measurement values in sets of eight, your measurement values seen as the entire set will not be atomic anymore. However, they do are atomic over sets of eight: Counting from the start of **ip_st3EMDeviceParams.areMeasurementsToChannels**, each eight measurement values are atomic. Thus, you may want to arrange the measurement value configuration so that related data (like current, power and voltage of the three phases) are in atomic sets.

Note on usage: WI strictly encapsulates I/O module parametrization in fieldbus specific parameter read and -write function blocks. Therefore, you need to connect the FB_UR20_3EM_Control with a suitable **FB_UR20_3EM_ChannelParam<field bus>Write** function block provided in this library. This is simple: Declare an instance of the **FB_UR20_3EM_ChannelParam<field bus>Write** in your POU or FB. Pass **ip_stDevice** and, if needed, **ip_udilIndex** to that instance. Then connect the instance to FB_UR20_3EM_Control's input **i_fbChannelParamAnyWrite**. Here is a piece of example code for a UR20_3EM_230V_AC attached to the UControl's SysBus:

```
fbChannelWrite3EMSysBus.ip_stDevice := UR20_3EM_230V_AC; (* tell channel parameter write FB
which device to write to *)
fb3EMControl( <other i./o. omitted here for brevity>, i_fbChannelParamAnyWrite :=
fbChannelWrite3EMSysBus); (* execute FB *)
```

The FB_UR20_3EM_Control also provides a conversion of the UR20_3EM_230V_AC module's raw values to SI units in its output variable **q_arIrrMeasurements**. Please observe that the conversion to SI requires a correctly set current transformer ratio in **ip_st3EMParams.IrCurrentTransformerRatio**:

- you use current transformers: their ratio is e.g. 40:1 then set **ip_st3EMParams.IrCurrentTransformerRatio** to *LREAL#40.0*
- you do not use current transformers: set **ip_st3EMParams.IrCurrentTransformerRatio** to *LREAL#1.0*

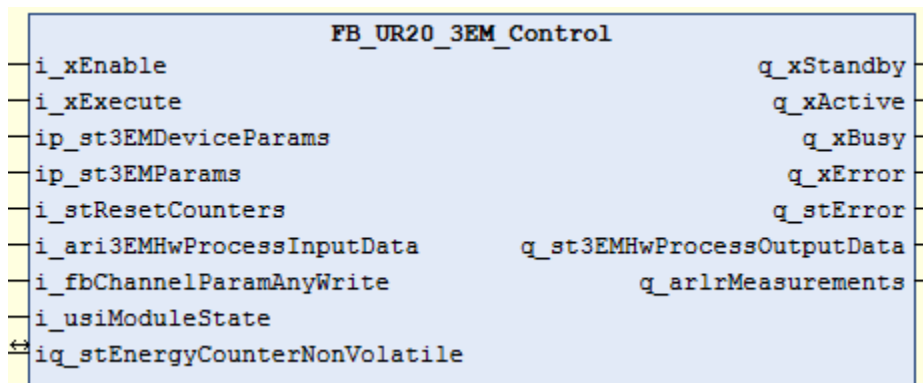
The UR20_3EM_230V_AC module has integer 16 bit energy counters. These overflow at 10000 Wh (the actual unit scales with the current transformer ratio). To provide the user with more flexibility, the FB_UR20_3EM_Control extends the energy counters to 32 bit integer FB-internally and provides the needed overflow management: Provided that the user has configured an energy counter measurement value, the FB checks continuously if an overflow has occurred. If so, it

calculates new internal extension value(s) and issues an energy counter reset pulse to the UR20_3EM_230V_AC module. The FB_UR20_3EM_Control calculates the SI values for energy counter measurement values from the sum of its internal energy counter extension and the actual value read from the UR20_3EM_230V_AC module. Because CoDeSys (and thus u-create studio) provide a choice of approaches for more-or-less non-volatile variable storage on a PLC, the FB_UR20_3EM_Control exposes in its **iq_stEnergyCounterNonVolatile** structure the accumulated overflown energy values. If you desire non- volatile storage, please declare and handle the variable connected to **iq_stEnergyCounterNonVolatile** according to your choice of persistence; See also “Data Persistence” in the help of u-create studio.

The FB_UR20_3EM_Control function block also provides you with a means to reset the extended energy counters: Rising edges on the four boolean flags in **i_stResetCounters** reset the associated energy counters in the 3EM module and the FB’s internal energy counter extension. Remember to reset the boolean flags to false after you issued an energy counter reset or the FB’s automatic overflow management will not work as expected.



The FB_UR20_3EM_Control converts UR20_3EM_230V_AC module raw values to SI units for you. For the correct function of this feature, it is necessary that the UR20_3EM_230V_AC module has been parametrized with the same parameters as you input into FB_UR20_3EM_Control via **ip_st3EMDeviceParams** and **ip_st3EMParams**. An easy way to achieve this is to set up the parameters in the associated variables **ip_st3EMDeviceParams** and **ip_st3EMParams** and connect these variables to both your instances of FB_UR20_3EM_Control and **FB_UR20_3EM_Param<field bus>Write**. Then execute the **FB_UR20_3EM_Param<field bus>Write** and wait for its **xDone** signal before you activate FB_UR20_3EM_Control. Please also refer to the documentation of **FB_UR20_3EM_Param<Field Bus>Read** and **FB_UR20_3EM_Param<Field Bus>Write**.



Possible Errors

Reason	Description
Invalid parameter	<ul style="list-style-type: none"> a) ip_st3EMParams.uiNumberOfMeasurementValues is greater than 56. b) ip_st3EMParams.lCurrentTransformerRatio is zero or negative. c) ip_st3EMParams.tDataTransfer is zero. d) iq_stEnergyCounterNonVolatile is no valid reference. e) i_fbChannelParamAnyWrite is no valid reference. <i>Will cause PLC exception at start of application.</i>

Reason	Description
invalid device parameter	a) an element of aretMeasurementsToChannels in the used range is set to the value reactiveEnergyLeadingCounterL<n> or to the value reactiveEnergyLaggingCounterL<n> , and ip_st3EMDeviceParams.bHarmonicSelect is $\neq 1$. b) an element of aretMeasurementsToChannels beyond the parametrized number of measurements is not set to ET_UR20_3EM_Measurement-Values#deactivated .
invalid reference	iq_stEnergyCounterNonVolatile became invalid in active state.
UR20_3EM_230V_AC module error	The UR20_3EM module reported an error in active state.

Inputs and Outputs

Scope	Name	Type	Comment
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
	ip_st3EMDeviceParams	<u>ST_UR20_3EM_DeviceParam</u>	multiplex operation only: FB_UR20_3EM_Control cycles the 3EM module through the measurement values list in aretMeasurementsToChannels
	ip_st3EMParams	<u>ST_UR20_3EM_Param</u>	number of measurement values in uiNumberOfMeasurementValues ; current transformer ratio in IrCurrentTransformerRatio ; multiplex operation only: tDataTransfer time.
	i_stResetCounters	<u>ST_UR20_3EM_ResetCounters</u>	rising edge on the four boolean flags in this struct resets the associated energy counters and FB-internal extension structures
	i_ari3EMHwProcessInputData	ARRAY [0..7] OF INT	connect to the eight channels in 3EM module -> i/o mapping -> input data
	i_fbChannelParamAnyWrite	REFERENCE TO <u>FB_UR20_3EM_ChannelParamAnyWrite</u>	connect to an instance of FB_ChannelParam <field bus> Write
	i_usiModulestate	USINT	connect to the 3EM's "module state" process input where exists or set to 1 where not.
Output	q_st3EMHwProcessOutputData	<u>ST_UR20_3EM_OutputData</u>	connect to the 3EM's i/o mapping -> output data
	q_ar1rMeasurements	ARRAY [0..55] OF LREAL	your SI measurement values in an array of LREAL
Inout	iq_stEnergyCounterNonVolatile	<u>ST_UR20_3EM_EnergyCounterNonVolatile</u>	energy counters: these structs contain the accumulated overflow energy values, for optional non-volatile storage.

7.2 FB_UR20_3EM_ChannelParamEcCanWrite

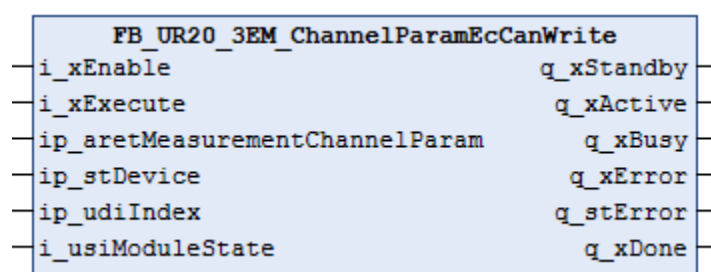
Functional Description

This function block writes a set of eight measurement channel device parameters to an UR20_3EM_230V_AC module attached to an UR20 EtherCAT fieldbus coupler connected to the PLC. Please also read the documentation of FB_UR20_3EM_Control regarding usage of this FB.

The function block can be used in combination with the following u-remote modules:
UR20_3EM_230V_AC



The module check is done for the whole coupler. The single module is not checked!
FB_UR20_3EM_ChannelParamEcCanWrite has level-controlled finite behavior.



Possible Errors

Reason	Description
Invalid parameter	ip_stDevice must not be zero.
invalid process value	Invalid value in ip_aretMeasurementChannelParam[] during standby-exit.
UR20_3EM_230V_AC module error	The UR20_3EM module reported an error in active state.
EtherCAT parameter write error	A parameter write access to the UR20_3EM_230V_AC module has failed.

Inputs and Outputs

Scope	Name	Type	Comment
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Output	q_xDone	BOOL	execution has come to its supposed end

Scope	Name	Type	Comment
Input	ip_aretMeasurementChannelParam	ARRAY [0..7] OF ET_UR20_3EM_MeasurementValues	maps measurement values on measurement channels.
	ip_stDevice	IloCopDevice	handle to the CanOpen or EtherCAT device, i.e. the fieldbus coupler
	ip_udiIndex	UDINT	start address of the module

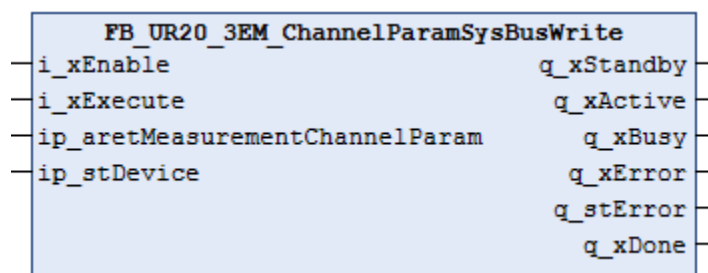
7.3 FB_UR20_3EM_ChannelParamSysBusWrite

Functional Description

This function block writes a set of eight measurement channel device parameters to an UR20_3EM_-230V_AC module directly attached to a UC20_SL2000_OLAC(..) PLC via the PLC's SysBus. Please also read the documentation of FB_UR20_3EM_Control regarding usage of this FB.

The function block can be used in combination with the following u-remote modules: UR20_3EM_230V_AC.

FB_UR20_3EM_ChannelParamSysBusWrite has level-controlled finite behavior.



Possible Errors

Reason	Description
Invalid parameter	ip_stDevice must not be zero.
invalid process value	Invalid value in ip_aretMeasurementChannelParam[] during standby-exit.
UR20_3EM_230V_AC module error	a) The UR20_3EM module reported an error in active state. b) The FB has lost the connection to the UR20_3EM module in active state.
Sysbus parameter write error	A parameter write access to the UR20_3EM_230V_AC module has failed.

Inputs and Outputs

Scope	Name	Type	Comment
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Output	q_xDone	BOOL	execution has come to its supposed end
Input	ip_aretMeasurementChannelParam	ARRAY [0..7] OF ET_UR20_3EM_MeasurementValues	maps measurement values on measurement channels.
	ip_stDevice	IIO	handle to the 3EM module

7.4 FB_UR20_3EM_ParamEcCanRead

Functional Description

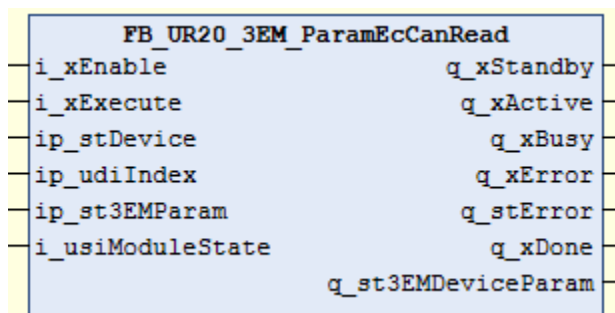
This function block reads all parameters from an UR20_3EM_230V_AC module attached to a PLC via an UR20 EtherCAT fieldbus coupler.

The function block can be used in combination with the following u-remote modules: UR20_3EM_230V_AC



The module check is done for the whole coupler. The single module is not checked!

FB_UR20_3EM_ParamEcCanRead has level-controlled finite behavior.



Possible Errors

Reason	Description
Invalid parameter	a) ip_stDevice must not be zero. b) ip_st3EMParams.lfCurrentTransformerRatio must not be zero or negative.
UR20_3EM_230V_AC module error	a) The UR20_3EM module reported an error in active state. b) The FB has lost the connection to the UR20_3EM module in active state.
EtherCAT parameter read error	A parameter read access to the UR20_3EM_230V_AC module has failed.

Inputs and Outputs

Scope	Name	Type	Comment
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Output	q_xDone	BOOL	execution has come to its supposed end
Input	ip_stDevice	lloCopDevice	handle to the CanOpen or EtherCAT device [coupler]
	ip_udiIndex	UDINT	start address of the module
	lp_st3EMParam	ST_UR20_3EM_Param	The fb's parameters.
	i_usiModulestate	USINT	The UR20_3EM 'module state' process input.
Output	q_stDeviceParam	ST_UR20_3EM_DeviceParam	The device parameter set read from the UR20_3EM_230V_AC module.

7.5 FB_UR20_3EM_ParamEcCanWrite

Functional Description

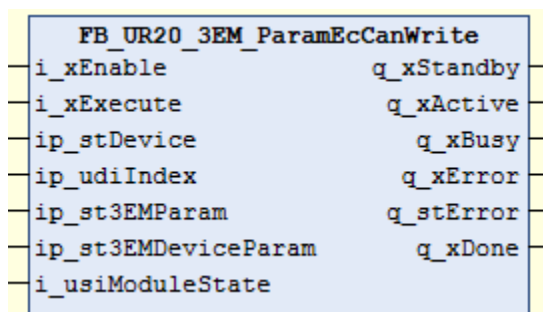
This function block writes a set of parameters to an UR20_3EM_230V_AC module attached to an UR20 EtherCAT fieldbus coupler connected to the PLC.

The function block can be used in combination with the following u-remote modules: UR20_3EM_230V_AC.



The module check is done for the whole coupler. The single module is not checked!

FB_UR20_3EM_ParamEcCanWrite has level-controlled finite behavior.



Possible Errors

Reason	Description
Invalid parameter	a) ip_stDevice must not be zero b) ip_st3EMParams.IrCurrentTransformerRatio is zero or negative
Invalid process value	a) bHarmonicSelect > 31 or bHarmonicSelect < 1 b) IrLowerVoltageLimit > 300.0 or < 0 c) IrUpperVoltageLimit > 300.0 or < 0 d) IrLowerCurrentLimit > 5.0 or < 0 (with 5A current range and IrTransformerRatio = 1) e) IrUpperCurrentLimit > 5.0 or < 0 (with 5A current range and IrTransformerRatio = 1) f) IrCurrentUnbalanceLimit > 1.0 or < 0 g) IrFrequencyAlarmLowerLimit < 45 or IrFrequencyAlarmLowerLimit > 65 h) IrFrequencyAlarmUpperLimit < 45 or IrFrequencyAlarmUpperLimit > 65 i) IrPowerFactorAlarmLowerLimit > 1.0 or < 0 j) an element of aretMeasurementsToChannels is set to the value <i>reactiveEnergyLeadingCounterL<n></i> or to the value <i>reactiveEnergyLaggingCounterL<n></i> , and ip_st3EMDeviceParams.bHarmonicSelect is <> 1.
UR20_3EM_230V_AC module error	a) The UR20_3EM module reported an error in active state. b) The FB has lost the connection to the UR20_3EM_230V_AC module.
EtherCAT parameter write error	A parameter write access to the UR20_3EM_230V_AC module has failed.

Inputs and Outputs

Scope	Name	Type	Comment
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Output	q_xDone	BOOL	execution has come to its supposed end
Input	ip_stDevice	IloCopDevice	handle to the CanOpen or EtherCAT device, i.e. the field bus coupler
	ip_udiIndex	UDINT	start address of the module
	Ip_st3EMParam	ST_UR20_3EM_Param	The fb's parameters.
	ip_st3EMDeviceParam	ST_UR20_3EM_DeviceParam	parameter set to be written
	i_usiModulestate	USINT	The UR20_3EM 'module state' process input.

7.6 FB_UR20_3EM_ParamSysBusRead

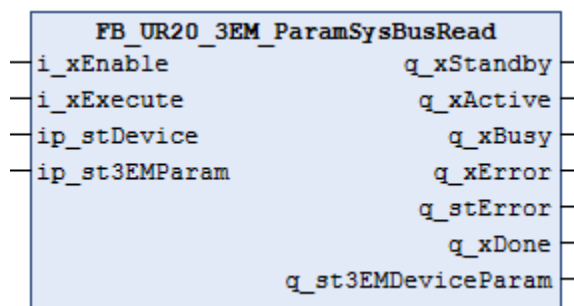
Functional Description

This function block reads all parameters from an UR20_3EM_230V_AC module directly attached to a UC20_SL2000_OLAC(..) PLC via the PLC's SysBus.

The function block can be used in combination with the following u-remote modules:

UR20_3EM_230V_AC

FB_UR20_3EM_ParamSysBusRead has level-controlled finite behavior.



Possible Errors

Reason	Description
Invalid parameter	a) ip_stDevice must not be zero. b) ip_st3EMParams.lfCurrentTransformerRatio must not be zero or negative.
UR20_3EM_230V_AC module error	The FB has lost the connection to the UR20_3EM module in active state.
Sysbus parameter read error	A parameter read access to the UR20_3EM_230V_AC module has failed.

Inputs and Outputs

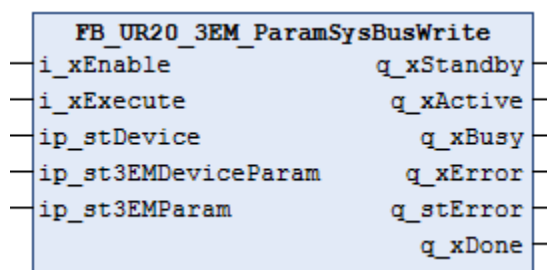
Scope	Name	Type	Comment
Output	<code>q_xStandby</code>	BOOL	waiting for activation
	<code>q_xActive</code>	BOOL	function block is activated
	<code>q_xBusy</code>	BOOL	function block is activated and doing its supposed task
	<code>q_xError</code>	BOOL	function block is in error state
	<code>q_stError</code>	ST_ErrorInfo	detailed error information
Input	<code>i_xEnable</code>	BOOL	enables the function block by switching from idle to standby
	<code>i_xExecute</code>	BOOL	activates the function block by switching from standby to active
Output	<code>q_xDone</code>	BOOL	execution has come to its supposed end
Input	<code>ip_stDevice</code>	IIO	handle to the 3EM module
	<code>ip_st3EMParam</code>	ST_UR20_3EM_Param	The fb's parameters.
Output	<code>q_st3EMDeviceParam</code>	ST_UR20_3EM-DeviceParam	The parameter set that was read.

7.7 FB_UR20_3EM_ParamSysBusWrite

Functional Description

This function block writes a set of parameters to an UR20_3EM_230V_AC module directly attached to a UC20_SL2000_OLAC(..) PLC its SysBus. The function block can be used in combination with the following u-remote modules: UR20_3EM_230V_AC

FB_UR20_3EM_ParamSysBusWrite has level-controlled finite behavior.



Possible Errors

Reason	Description
Invalid parameter	a) ip_stDevice must not be zero b) ip_st3EMParams.IrCurrentTransformerRatio must not be zero or negative
Invalid process value	a) bHarmonicSelect > 31 or bHarmonicSelect < 1 b) IrLowerVoltageLimit > 300.0 or < 0 c) IrUpperVoltageLimit > 300.0 or < 0 d) IrLowerCurrentLimit > 5.0 or < 0 (with 5A current range and IrTransformerRatio = 1) e) IrUpperCurrentLimit > 5.0 or < 0 (with 5A current range and IrTransformerRatio = 1) f) IrCurrentUnbalanceLimit > 1.0 or < 0 g) IrFrequencyAlarmLowerLimit < 45 or IrFrequencyAlarmLowerLimit > 65 h) IrFrequencyAlarmUpperLimit < 45 or IrFrequencyAlarmUpperLimit > 65 i) IrPowerFactorAlarmLowerLimit > 1.0 or < 0 j) an element of aretMeasurementsToChannels is set to the value <i>reactiveEnergyLeadingCounterL<n></i> or to the value <i>reactiveEnergyLaggingCounterL<n></i> , and ip_st3EMDeviceParams.bHarmonicSelect is <> 1.
UR20_3EM_230V_AC module error	a) The UR20_3EM module reported an error in active state. b) The FB has lost the connection to the UR20_3EM_230V_AC module.
Sysbus parameter write error	A parameter write access to the UR20_3EM_230V_AC module has failed.

Inputs and Outputs

Scope	Name	Type	Comment
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Output	q_xDone	BOOL	execution has come to its supposed end
Input	ip_stDevice	IIO	handle to the 3EM module
	ip_st3EM-DeviceParam	ST_UR20_3EM-DeviceParam	parameter set to be written
	Ip_st3EMParam	ST_UR20_3EM_Param	The fb's parameters.

8 libWiUr20Diag

Weidmueller Interface library for reading diagnostic data of a u-remote module. Weidmueller Interface strongly recommends reading the UR20 manual related to the deployed module available on Weidmueller website before using this library.

Each u-remote module can generate diagnostic data, e.g. in the event of a channel error. The diagnostic data is always an array of 47 bytes.

8.1 FB_UR20_Uremote_Diagdata_Ethercat

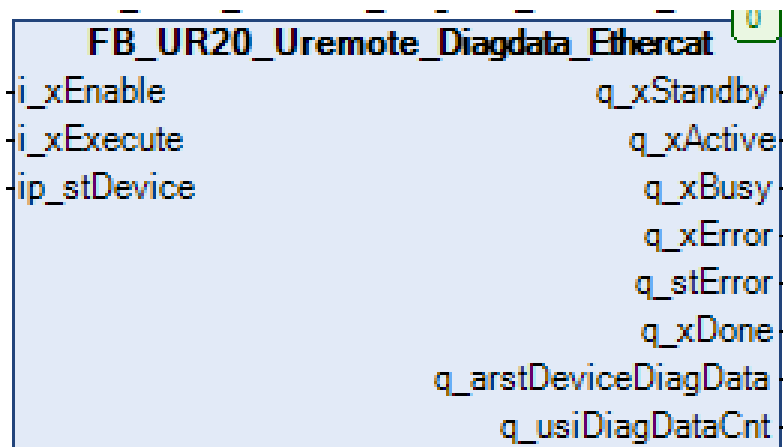
Functional Description

The function block can read the diagnostic data from a u-remote module connected to an EtherCAT coupler.

The function block reads all available messages at once. The output array element `q_arstDeviceDiagData[0]` shows the newest diagnostic message.

The diagnostic functionality must be activated inside the EtherCAT coupler. Without that activation the function block returns an error message.

After all messages are loaded, the function block changes to **done** state.



Inputs and Outputs

Scope	Name	Type	Comment
Output	<code>q_xStandby</code>	BOOL	waiting for activation
	<code>q_xActive</code>	BOOL	function block is activated
	<code>q_xBusy</code>	BOOL	function block is activated and doing its supposed task
	<code>q_xError</code>	BOOL	function block is in error state
	<code>q_stError</code>	ST_ErrorInfo	detailed error information

Scope	Name	Type	Comment
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Output	q_xDone	BOOL	execution has come to its supposed end
Input	ip_stDevice	IO_DEVICE_REF	device reference to the Ethercat coupler
Output	q_arstDeviceDiagData	ARRAY [0..20] OF ST_UR20_DiagData-EcUremote	output array contains the diagnostic data
	q_usiDiagDataCnt	USINT	counter how many diagnostic data messages are available

Possible Errors

Reason	Description
invalid parameter "ip_stDevice"	The parameter requires mandatory an input
invalid process "ip_stDevice"	input device could not be found during transition to state active.
Sub-FB returns an error	Please check the error parameter output for more details

8.2 FB_UR20_Uremote_Diagdata_Sysbus

The function block can read the diagnostic data from a u-remote module connected directly to the SysBus.

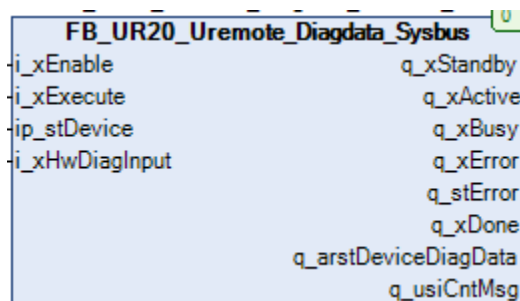
To get the data it is necessary to activate the switch "Activate diagnosis for all devices" inside the SysBus editor from u-create studio.

If multiple messages are available and the input i_xHwDiagInput (Module diagnosis input) is connected to the hardware diagnostic bit of the module, the fb reads all available messages at once.

The diagnostic bit can be found in the module IO-mapping. If the input is not connected, the function block reads only one message.

The output array element q_arstDeviceDiagData[0] shows the newest diagnosis message.

After all messages are loaded, the function block changes to **done** state.



Inputs and Outputs

Scope	Name	Type	Comment
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Output	q_xDone	BOOL	execution has come to its supposed end
Input	ip_stDevice	IO_SYSBUS_DEVICE_REF	handle to the u-remote module
	i_xHwDiagInput	BOOL	handle to the diagnostic bit inside u-remote module IO mapping
Output	q_arstDeviceDiagData	ARRAY [0..20] OF ST_UR20_DiagDataUremote	array of struct contains diagnostic data outputs from u-remote module
	q_usiCntMsg	USINT	Number of diagnostic messages

Possible Errors

Reason	Description
invalid parameter "ip_stDevice"	The parameter requires mandatory an input
invalid process value "ip_stDevice"	input device could not be found during transition to state active.
Method GetDiagData (diag not activated)	The diagnosis is not activated inside the sysbus editor
Method GetDiagData (bad result)	The Method returns an error, please check error parameter

9 libWiUr204ComIOLink

This library provides function blocks for non-cyclic communication with an UR20-4COM-IO-LINK module connected to an EtherCAT- or ModBus TCP fieldbus coupler.

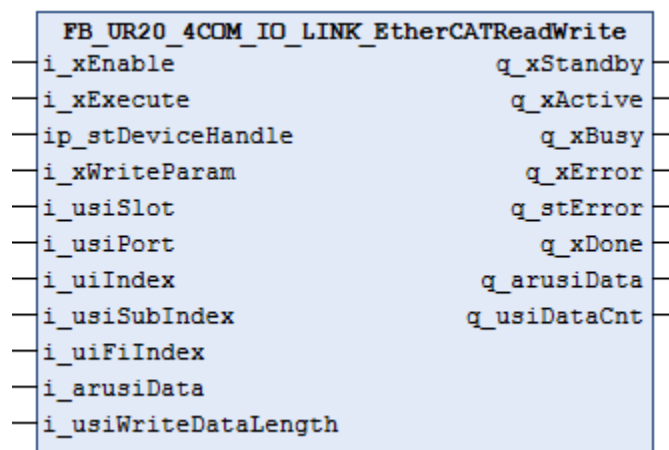
9.1 FB_UR20_4COM_IO_LINK_EtherCATReadWrite

Functional Description

The function block provides non-cyclic communication with UR20-4COM-IO-LINK module and IO-Link slave device over EtherCAT. It is possible to read and write data from/to a connected IO-Link device.

The inputs of the function block must contain the position of the IO-Link device and which parameter (Index + Subindex) shall be read or written.

Please read the IO-Link Device manual to get more information about the parameter index.



Possible Errors

Reason	Description
Invalid parameter	ip_stDeviceHandle must be a valid UR20_4COM_IO_LINK handle.
Invalid process value	a) i_usiSlot < 1 or > 64 b) i_usiPort < 1 or > 4 c) i_uiIndex < 1 d) i_usiSubIndex < 0 e) i_uiFiIndex < 0 f) i_xWriteParam is true and (usiWriteDataLength <= 0 or > 2333)
Read / Write timeout	The Read/Write operation takes more than 5s, please check communication settings.
Read / write error	The sub-fb reports an error during the io-link data write or read process in active state.
Received error via IO-Link call	IO-Link Module responds with an error.

Inputs and Outputs

Scope	Name	Type	Comment
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Output	q_xDone	BOOL	execution has come to its supposed end
Input	ip_stDeviceHandle	IoApi_Hdl	UR20 EtherCAT coupler hardware module reference
	i_xWritePram	BOOL	False = Read; True = Write
	i_usiSlot	USINT	Hardware slot of the station where the UR20 module is connected; 1..64
	i_usiPort	USINT	hardware port of the UR20 module where IO-Link device is connected; 1 ... 4
	i_uiIndex	UINT	IO-Link Device parameter Index (See IO-Link slave manual)
	i_uiSubIndex	UINT	IO-Link Device parameter Sub-Index (See IO-Link slave manual)
	i_uiFiIndex	UINT	IO-Link FI Index (See IO-Link slave manual)
	i_arusiData	ARRAY [0..232] OF USINT	Data to be written to the IO-Link Device (see IO-Link slave manual)
	i_usiWriteDataLength	USINT	Data length to be written to the IO-Link Device
Output	q_arusiData	ARRAY [0..232] OF USINT	Data read from the IO-Link Device (see IO-Link slave manual)
	q_usiDataCnt	USINT	Length of the read IO-Link data in byte

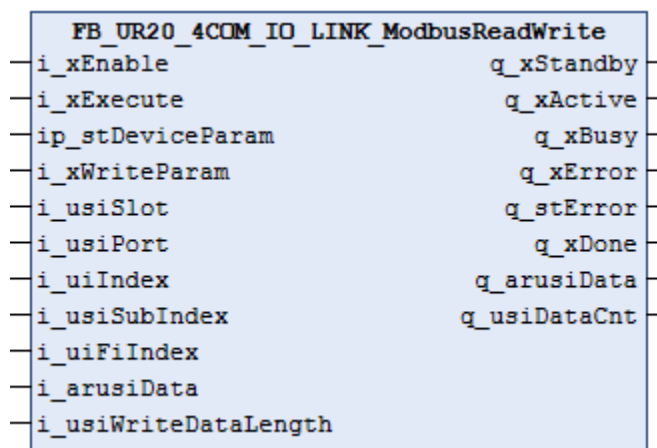
9.2 FB_UR20_4COM_IO_LINK_ModbusReadWrite

Functional Description

The function block provides non-cyclic communication with UR20-4COM-IO-LINK module and IO-Link slave device over Modbus TCP. It is possible to read and write data from / to a connected IO-Link device.

The inputs of the function block must contain the position of the IO-Link device and which parameter (Index + Subindex) shall be read or written.

Please read the IO-Link Device manual to get more information about the parameter index.



Possible Errors

Reason	Description
Invalid parameter	ip_stDeviceHandle is not connected to a device.
Invalid process value	g) i_usiSlot < 1 or > 64 h) i_usiPort < 1 or > 4 i) i_uiIndex < 1 j) i_usiSubIndex < 0 k) i_uiFiIndex < 0 l) i_xWriteParam is true and (usiWriteDataLength <= 0 or > 2333)
Read / Write timeout	The Read/Write operation takes more than 5s, please check communication settings.
Read / write error	The sub-fb reports an error during the io-link data write or read process in active state.
Receive error via IO-Link call	IO-Link Module responds with an error.

Inputs and Outputs

Scope	Name	Type	Comment
Output	q_xStandby	BOOL	waiting for activation
	q_xActive	BOOL	function block is activated
	q_xBusy	BOOL	function block is activated and doing its supposed task

Scope	Name	Type	Comment
	q_xError	BOOL	function block is in error state
	q_stError	ST_ErrorInfo	detailed error information
Input	i_xEnable	BOOL	enables the function block by switching from idle to standby
	i_xExecute	BOOL	activates the function block by switching from standby to active
Output	q_xDone	BOOL	execution has come to its supposed end
Input	ip_stDeviceHandle	libWIModbusTCPClient. ST_ModbusTCP_ControlParam	UR20 Modbus TCP coupler hardware module reference
	i_xWritePram	BOOL	False = Read; True = Write
	i_usiSlot	USINT	Hardware slot of the station where the UR20 module is connected; 1..64
	i_usiPort	USINT	hardware port of the UR20 module where IO-Link device is connected; 1 ... 4
	i_uiIndex	UINT	IO-Link Device parameter Index (See IO-Link slave manual)
	i_uiSubIndex	UINT	IO-Link Device parameter Sub-Index (See IO-Link slave manual)
	i_uiFiIndex	UINT	IO-Link FI Index (See IO-Link slave manual)
	i_arusiData	ARRAY [0..232] OF USINT	Data to be written to the IO-Link Device (see IO-Link slave manual)
	i_usiWriteDataLength	USINT	Data length to be written to the IO-Link Device
Output	q_arusiData	ARRAY [0..232] OF USINT	Data read from the IO-Link Device (see IO-Link slave manual)
	q_usiDataCnt	USINT	Length of the read IO-Link data in byte