

UC20-SL2000-OLAC-EC | UR20-1COM-232-485-422

Application Note | Modbus RTU Master

Abstract:

The Application Note describes how to set up a Modbus RTU communication on a UC20-SL2000 controller. The document contains instructions on how to integrate the Weidmüller Modbus RTU library “libWIUR20ModbusRTUMaster” into u-create studio software and how to use the Modbus RTU Function Block “FB_ModbusRTUMaster” containing in the library to create a Modbus RTU communication via u-remote module UR20-1COM-232-485-422.

Hardware reference

No.	Component name	Article No.	Hardware / Firmware version
1	UC20-SL2000-OLAC-EC	2638920000	-
2	UR20-1COM-232-485-422	1315750000	FW: V 1.00.12 (or higher)
3	POWER MONITOR 51A	1470260000	-

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	AN0042-UC20-SL2000 Modbus RTU.zip	The file contains a Modbus RTU package and an example project related to this document	-

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>

Content

1 Warning and Disclaimer 4

2 Requirements 5

3 Install Modbus RTU Package 6

4 FB_ModbusRTUMaster 7

5 Create a Modbus RTU communication 9

5.1 Example Program 10

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 safety equipment provided / electrical safety design or other redundant safety features that are independent from the automation system.

Disclaimer

This Application Note / Quick Start Guide / Example Program does not relieve you of the obligation to handle it safely during use, installation, operation and maintenance. Each user is responsible for the correct operation of his control system. By using this Application Note / Quick Start Guide / Example Program 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

The given descriptions and examples do not represent any customer-specific solutions, they are simply intended to help for typical tasks. The user is responsible for the proper operation of the described products. Application notes / Quick Start Guides / Example Programs are not binding and do not claim to be complete in terms of configuration as well as any contingencies. By using this Application Note / Quick Start Guide / Example Program, you acknowledge that we cannot be held liable for any damages beyond the described liability regime. We reserve the right to make changes to this application note / quick start guide / example at any time without notice. In case of discrepancies between the proposals Application Notes / Quick Start Guides / Program Examples and other Weidmüller publications, like manuals, such contents have always more priority to the examples. We assume no liability for the information contained in this document. Our liability, for whatever legal reason, for damages caused using the examples, instructions, programs, project planning and performance data, etc. described in this Application Note / Quick Start Guide / Example 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 Requirements

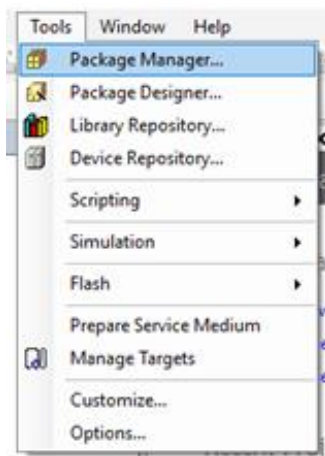
To work through this document, it is necessary to install the u-create studio software on an engineering PC and connect the patch cable between the controller and the PC. Furthermore, the UR20-1COM-232-485-422 module must be connected to the Modbus RTU Slave device.



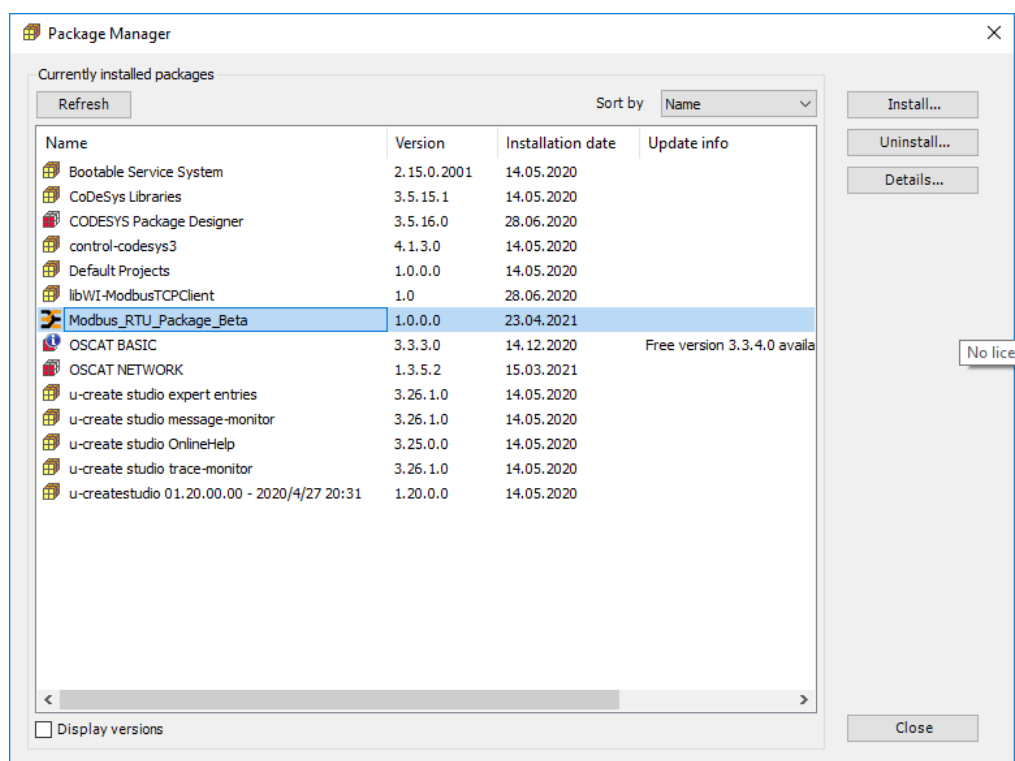
To understand the content of this document, basic knowledge about handling projects in u-create studio is required.

3 Install Modbus RTU Package

- Open u-create studio software and start the Package Manager

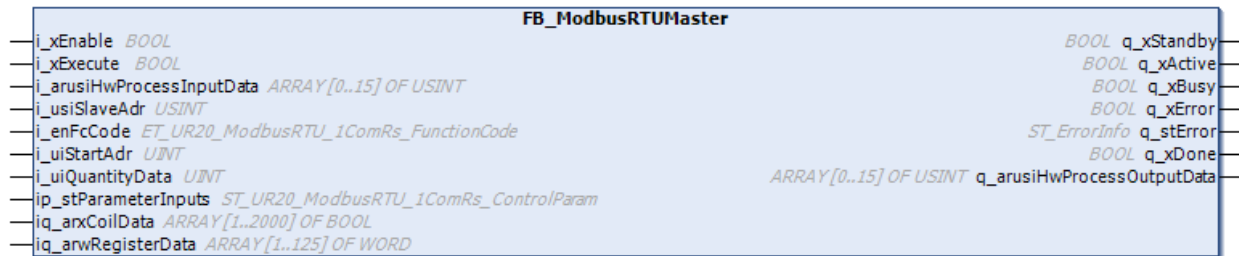


- Install the Modbus RTU library package. (Typical setup)



4 FB_ModbusRTUMaster

The function block implements a Modbus RTU Master that provides communication with a Modbus RTU Slave via UR20-1COM-232-485-422.



The behavior and the inputs/outputs of the function block are explained in the documentation of the function block. You can find some information inside the online help of u-create studio.

FB_ModbusRTUMaster (FB)

FUNCTION_BLOCK FB_ModbusRTUMaster EXTENDS FB_LevelControlledFiniteBehavior

Weidmüller

Functional Description

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

Supported Functions

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_usiStartAdr. 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":

ErrorInterface (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 Errorcode

In/Out

Scope	Name	Type	Initial	Comment	Inherited from
Output	q_xStandby	BOOL		waiting for activation	FB_StandardBehavior

Index results

The information can also be found directly inside the library repository.

Add Library X Delete Library Properties Details Placeholders Library Repository Icon legend...

Name	Namespace	Effective version
IoStandard = IoStandard, 3.5.15.0 (System)	IoStandard	3.5.15.0
K_Jo = K_Jo, " (KEBA AG)	K_Jo	4.1.4.0
Modbus_RTU_Master = libWtlr20ModbusRTUMaster, 1.1.0.9 (Weidmueller)	libWtlr20ModbusRTUMaster	1.1.0.9
Standard = Standard, 3.5.15.0 (System)	Standard	3.5.15.0
SymbolicVarsBase, 3.5.15.0 (System)	SymbolicVarsBase	3.5.15.0


libWtlr20ModbusRTUMaster, 1.1.0.9 (Weidmueller)

10_Enumerations20_Structure_Types40_Function_BlocksFB_ModbusRTUMaster60_Images

Inputs/OutputsGraphicalDocumentation

FB_ModbusRTUMaster (FB)

FUNCTION_BLOCK FB_ModbusRTUMaster EXTENDS FB_LevelControlledFiniteBehavior



Functional Description

The function block FB_ModbusRTUMaster implements a Modbus RTU Master that provides communication with a Modbus slave via u-remote I/O.

Supported Functions

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..20].
2	Read Input Status	Reads binary inputs from a connected slave. The data is stored in Array iq_arxCoilData[1..20].
3	Read Holding Registers	Reads register-data from a connected slave. The data is stored in Array iq_arvRegisterData[1..125].
4	Read Input Registers	Reads input registers from a connected slave. The data is stored in Array iq_arvRegisterData[1..125].
5	Write Single Coil	Sends a binary output (Coil) to a connected slave. The value from array iq_arxCoilData[1] is used.
6	Write Single Register	Sends a single data word to a connected slave. The value from array iq_arvRegisterData[1] is used.
8	Diagnostics	Sends a diagnostics request with a user defined subfunction code to a connected slave. The value from array iq_arvRegisterData[1] is used.
15	Write Multiple Coils	Sends several binary outputs (Coils) to a connected slave. The data must be provided in array iq_arxCoilData[1..125].
16	Preset Multiple Registers	Sends data to a connected slave. The data must be provided in array iq_arvRegisterData[1..125].

Possible Errors

5 Create a Modbus RTU communication

► Proceed with the following steps to create a Modbus RTU communication:

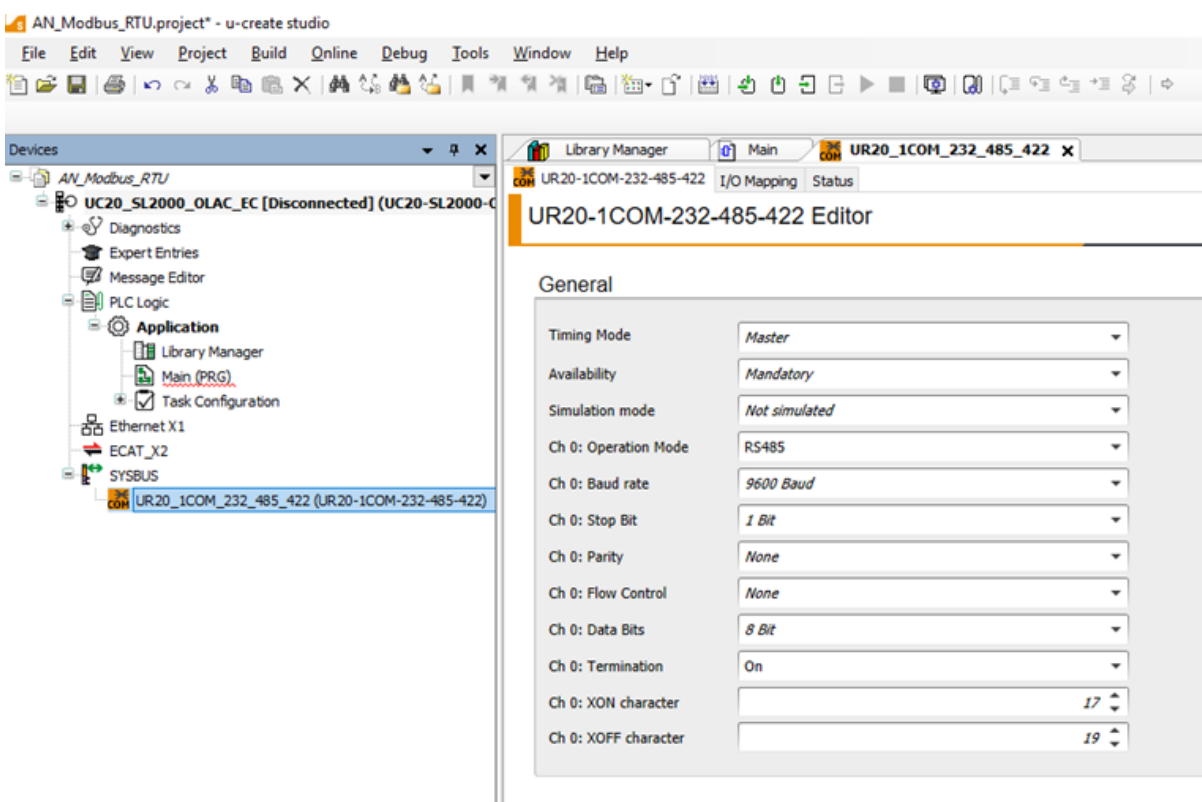
1. Add the Modbus RTU library (libWIUR20ModbusRTUMaster) to your PLC project.
2. It is **essential** to know the parameters and settings of the Modbus slave. The Modbus communication parameters must be adjusted.

In the example a Weidmüller Power Monitor 51A is used with the following parameters:

Example:

Slave device address: 11
 Baud rate: 9600 kbps
 Parity: none
 Stop bits: 1

3. Insert the UR20-1COM-232-485-422 module in u-create studio and adjust the parameters for the communication.



4. The IO-Mapping can be done by mapping the associated Data-Address to a PLC variable.

I/O Mapping

Endpoint	Variable	Type	Address	Value
Output data				
Ch 0: Output Data		ARRAY[0..15] OF USINT	%QB0	
Input data				
Ch 0: Input Data		ARRAY[0..15] OF USINT	%IB0	
Diagnosis				

```

14 arusiControlInput AT%IB0 : ARRAY[0..15] OF USINT;
15 arusiControlOutput AT%QB0 : ARRAY[0..15] OF USINT;
16

```

5. Import the function block **FB_ModbusRTUMaster** to your PLC program and create a program sequence to execute the function block.

In the example the following settings are used to read registers from the Power Monitor:

```

20: xEnable          := TRUE;
    uiSlaveAdr       := 11;      (* Slave Address *)
    enFC             := ET_UR20_ModbusRTU_lComRs_FunctionCode.enReadHoiReg; (* FC code *)
    uiStartAdr       := 16#A4;   (* Start address *)
    uiCntData        := 2;      (* Length *)
    stParameter.tReqTimeout := T#2S; (* Timeout Error *)
    stParameter.tHwMemFlushTimer := T#20MS; (* delete tImer should be 2 Cycles*)

```

Information about Modbus Register from Power Monitor 51 can be downloaded from Weidmueller [website](#).

5.1 Example Program

The program permanently reads the three input voltage values via Modbus RTU.

Parameter	Unit	Data register	Type	Range: Hexadecimal	Function code
Instantaneous apparent power (3)	0.01kVA	00A0H <LSB> 00A1H <MSB>	unsigned 32bit	0H to 5F5E0FFH	03H
Total instantaneous apparent power	0.01kVA	00A2H <LSB> 00A3H <MSB>	unsigned 32bit	0H to 11E1A2FDH	03H
Voltage 1	0.1V	00A4H <LSB> 00A5H <MSB>	unsigned 32bit	0H to 3B9AC9FFH	03H
Voltage 2	0.1V	00A6H <LSB> 00A7H <MSB>	unsigned 32bit	0H to 3B9AC9FFH	03H
Voltage 3	0.1V	00A8H <LSB> 00A9H <MSB>	unsigned 32bit	0H to 3B9AC9FFH	03H

Inside the PLC program the values must be changed to one 32 Bit value, because the Power Monitor transmit the voltage value inside two Modbus registers.

```
50: IF( xdoneTRUE ) THEN
    xExecuteTRUE := FALSE;
    arwRegisterDataSave := arwRegisterData;
    arxCoilDataSave := arxCoilData;

Voltage_1 234 ▸ := TO_REAL(TO_DWORD(arwRegisterDataSave[1] 2339) OR SHL(TO_DWORD(arwRegisterDataSave[2] 0),16))/10.0;
Voltage_2 234 ▸ := TO_REAL(TO_DWORD(arwRegisterDataSave[3] 2339) OR SHL(TO_DWORD(arwRegisterDataSave[4] 0),16))/10.0;
Voltage_3 0 ▸ := TO_REAL(TO_DWORD(arwRegisterDataSave[5] 0) OR SHL(TO_DWORD(arwRegisterDataSave[6] 0),16))/10.0;
```