INTRODUCTION

The second serial port (port B, 6 pole RJ45 connector) provide a Modbus master protocol functionality. Only a subset of the Modicon Asynchronous Link Protocol is implemented. The Modbus RTU framing mode is used. The following two Modbus commands are used: Read Holding Registers (command 3), and Preset Multiple Registers (command 16).

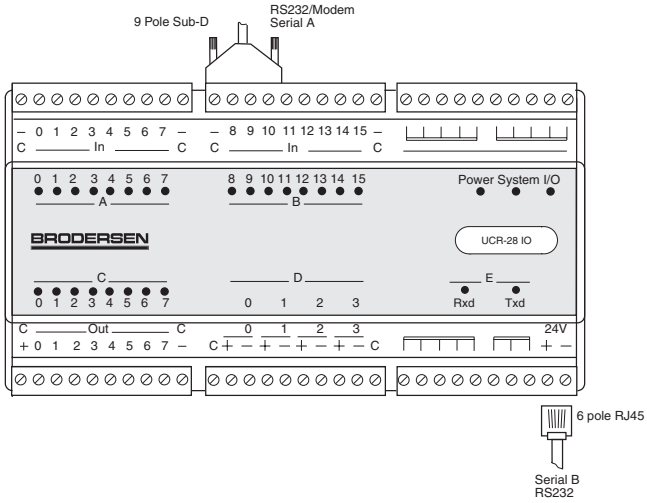
The Modbus Slave protocol on the RTU8 is tolerant to odd characters added to the Modbus frame e.g. generated when passing through a radio link connection.

The Modbus Master port support dial-up features for switched telephone systems. Only I/O are accessible. Log is not accessible through this port.

The port is configurable from 300 to 9600 baud, 8 data bit, non/even/odd parity.

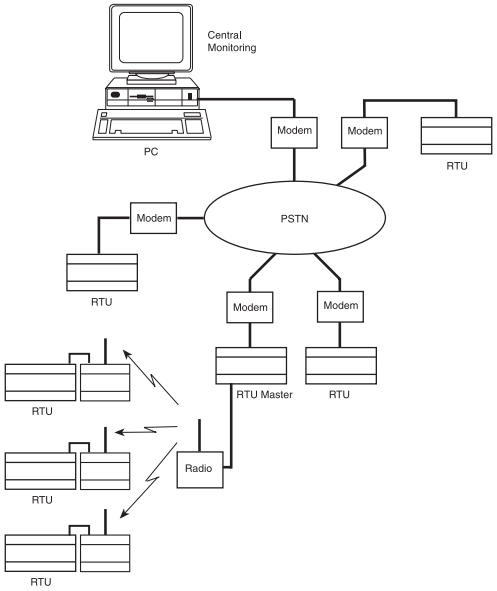
For additional RTU8 data, please see the general RTU8 data sheet.

RTU8 with two serial ports

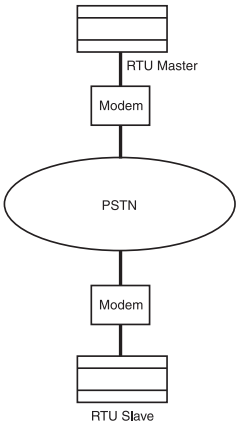


TYPICAL APPLICATIONS

Multi point application



Point-to-point application



VERSIONS/ORDERING CODES

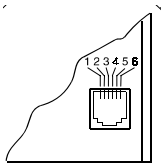
The code for indicating the presence of the second serial port on the RTU8 module is added to the normal RTU8 ordering/type code.

		UCR-xxx/RTUX21.Dx
Type		
UCR	UCR-xxx/RTU	
Serial port options		
No port	blank	
Port with Modbus Master protocol	1	

TECHNICAL DESCRIPTION

General
The Modbus master driver is implemented as a separate task in the RTU8 firmware. The driver will operate at the second RS232 port of the Dallas 80C320 controller in the RTU8. If the driver is enabled in the configuration menu, it occupies a number of BM registers. BM 30..35 are used for driver control, and BM600..999 are used for Modbus data buffers. If the driver is disabled all the BM registers are free for application use.

Hardware/serial interface
The second serial RS232 port is based on a 6 pole RJ45 connector, and therefore not full blown. Only one input, and one output handshake signals are provided. The data signals are connected to the additional UART on the Dallas 80C320 micro controller.
The RS232 port provide the following signals:



Pin 1	SG	Signal Ground	Electronic GND
Pin 2	RTS	Request To Send	Output
Pin 3	RX	Receive data	Input
Pin 4	TX	Transmit data	Output
Pin 5*	DCD/CTS	Data Carrier Detect	Input
Pin 6		Shield Module housing	

*CTS when in null modem mode.

The driver will use RX, TX , DCD/CTS and RTS (and GND) signals to communicate with a dialup modem or a null modem connection (e.g. radio modem).
If dial-up mode is enabled, the driver activates the RTS signal permanently, and use the DCD to determinate when a connect is established. When DCD is activated. The master driver send Modbus request to the slave station, no matter if the master initiate a connection, or it is receiving an incoming call. If dialup mode is disabled, the CTS and RTS is activated. This is used to changes the signal direction in multi drop systems. Handshake is setup in configuration table.

Communication modes, Modem and Null-Modem
The terms 'Modem Mode' or 'dialup' defines systems where some kind of dialling a number is necessary to establish a connection between two stations. This is not limited to standard PSTN modems, but could be GSM modems or Ethernet adapters as well.
The terms 'Null Modem Mode' or 'non dialup' defines systems where no dialling is necessary to establish a connection between two stations. However some kind of modem could still be involved (e.g. radio modems).

Modem mode (Dial-up mode)
Both the RTU8 master, and RTU8 slave (substation) is able to initiate a dialup to each other. The master may dial a number of substations at certain intervals to collect data, while in case of an alarm the substation may dial up the master, and deliver data.
The application related modem control, and conditions for dialup, are handled by the B-CON application programme in the RTU8. Dialup is control by 6 internal registers (BM30..BM35).
The calls are made to already prestored numbers in the RTU8 telephone book. The numbers are changed via the IOTOOL32 configuration menu. If a connection fails, the RTU8 will try a number of times (Retry Counts) to connect. If this fails, the RTU8 will suspend dialup, and the error is reported to the application (BM32.7 is set to '1'). The error flag is reset when the RTU8 receives an incoming call, or when the dial command register (BM30) is activated (changed from 0 to 1) again to force a new dial up.

Low level modem control is performed by the firmware, which include modem initialisation, executed at power up and every time a dialup is initiated.
Standard Hayes AT commands are used to control (e.g. ATD for dial) the modem. The following string is by default used by the RTU8 to initialise the modem.

AT &C1 &D0 S0=1 E0 V0

&C1	Track presence of data carrier (DCD)
&D0	Modem ignores Data Terminal Ready signal (DTR)
S0=1	Modem in Auto answer mode
E0	Modem commands are not echoed
V0	Modem in numerical response mode

Many modem manufactures makes minor additions or variation from the standard, which make it necessary to consult the actual modem manual, to ensure correct initialisation. The initialisation string is changed via the IOTOOL32 configuration tool.
In most modems, the initialisation string could be stored in the modems internal non volatile memory. In this case the RTU8 string could be omitted.

The RS232 port is not full blown, which means all the normal modem handshake signals are not available. Only one input signal (DCD) and one output signal (RTS) are available. It is then important to setup the modem to ignore the DTR signal (typical &D0), or connect DSR and DTR locally on the modem. In dialup mode the RTS output signal is always on.

Null Modem mode (dial-up procedure disabled)
In applications where point to point, multi drop or radio modem are used, the dialup configuration should be disabled. Telephone numbers, modem init string and AT commands are then disabled. In this mode are all communication initiated by the master station. The slave station is not able initiate a connection, like in dialup mode. All substation addressing are done by the Modbus slave address byte (BM 33). RTS/CTS function is available as on the primary port.

Command registers
A number of BM registers are used by the B-CON application program, in conjunction with some configuration registers, to control the Modbus driver. The next will describe the registers in table format, followed by a description in text.

The BM registers are defined as follows:

Register	Description
BM30	Modbus Command register m30.0..3 0 = Idle mode 1 = initiate data transfer (dialup) 2 = stop data transfer (hang-up) m30.4 '0' = don't send Modbus output '1' = send Modbus output m30.5 Not used m30.6 Not used m30.7 Not used
BM31	Telephone number to dial (0..29)
BM32	
	m32.0 Not used m32.1 Not used m32.2 Not used m32.3 DCD Data Carrier Detect (Input) m32.4..5 Communication state (Input) 00 Idle No communication (com counter=0) 01 Active: outgoing communication (com counter >= 1) 10 Active: Incoming Communication (com counter >=1) m32.6 Not used m32.7 Dial request suspended (Input)
BM33	Communication counter (Input)
BM34	Modbus Node address (Output)
BM35	Modbus data buffer offset (Output)

Please see the description for each register below:

Command register (BM30 output)
Used to initiate and terminate a connection. This in either dialup mode or non dialup mode. When in dialup mode, the call is made to a prestored telephone number, which is selected in the telephone register. When in non dialup mode, the telephone numbers and telephone registers are don't care. A data transfer is initiated by changing the registers low nibble from 0 to 1 for at least one scan. The automatic dial firmware will make the connection, and keep it until 2 is written to the registers low nibble, or com counter value has expired (see later).
Bit 4 defines if Modbus output is transmitted to the slave or not. If set to '0' no output is send. If set to '1' the output is send to the slave. When

a connection is terminated the bit is always cleared to '0'.

Telephone number register (BM31 output)
Used to select which of the up to 30 prestored telephone numbers to dial. The prestored telephone numbers are defined in the IOTOOL 32 configuration menu.
The prestored telephone numbers (telephone book) are shared by both Modbus ports.

DCD (BM32 input)
Handshake signal from the modem. May be used by the B-CON application program.

Communication state register (BM32.4..5 input)
Used to monitor the state of an connection.
00: The system is idling.
01: A connection initiated by the master (outgoing call) is success fully established. Output holding registers are read back, and input holding registers are read once.
10: A connection initiated by the slave (incoming call) is successfully established. Input holding registers are read once.

Dial suspend register (BM32.7 input)
Set to '1' if a connection to a station has failed, after the predefined numbers of retries.

Communication counter register (BM33 input / output)
Used to monitor the I/O data transfer. The counter is incremented in every Modbus cycle. A cycle is one the following 3 cases:

1. Read back output holding registers, and read input holding registers first time.
   When the master connects to a slave, if first read back output holding registers (this could be disabled in the configuration table, see later), and next read input holding registers. The communication counter is incremented from 0 to 1, and communication state bits is set to 01 (active outgoing call).

2. Reading input holding registers regularly during connect.
   If no Modbus output should be send (BM30.4 = '0'), the communication counter is incremented every time inputs are read.

3. Reading input holding registers, and writing output holding registers during connect.
   If Modbus output should be send (BM30.4 = '1'), the communication counter is incremented. After the output is send, M30.4 is cleared to '0'.

This is used by the B-CON application program to terminated connection after a number of cycles, or to automatic terminate when the predefined 'Max comm cnt' configuration value is reached (see configuration section).
If 'Max comm cnt' is set to e.g. 4, the application program could prevent automatic termination by keeping communication counter down (e.g. writing 0 to the register) as long as needed. If the connection is open for a long time, the counter will roll over at 255 and start from 0 again. When the connection is terminated, communication state and communication counter registers are returned to 0.

Modbus node register (BM34 output)
Used to select the Modbus node to address. The Modbus protocol provide the possibility to operate multiple nodes in a multi drop network. BM34 selects the node .
Note! When using dial-up to a RTU8 substation, the node number must always be set to one.

Modbus data buffer offset register (BM35 output)
Used to set a offset into the Modbus receive and transmit buffers. Data received from the Modbus are located from BM register 600..799 (100 words input), and data transmitted on the Modbus are located from BM register 800..999 (100 words output). If BM35 is set to 0 the first Modbus input word is placed in BM 600..601, and the first Modbus output word is read from BM 800..801. If BM35 is set to e.g. 20 the first Modbus input word is placed in BM 620..621, and the first Modbus output word is read in BM 820..821 (note the buffer offset is in bytes). This could be useful if the master is collecting 10 data word from e.g. 5 substations, then data from all stations could be in the buffers at the same time.

CONFIGURATION

A number of configuration fields are provided. The field values are changed and downloaded into the RTU8 flash memory, using the IOTOOL 32 configuration menu. The fields are used for values which are programmed once when setting up the module (e.g. telephone numbers, and baud rate).
The following fields are provided to control the Modbus master port driver in addition to the already existing fields in the configuration menu.

Configuration table

Field	Type	Description	Min	Max	Default value	Unit
n	S	Port B cfg	1	1	1:	Modbus
n+1	S	Use Driver	1	2	1: No	Port B
n+2	S	Baud rate	1	6	6: 9600	Port B
n+3	S	Parity	1	3	1: None	Port B
n+4	S	Handshake	1	4	3 RTS On/off	PortB
n+5	W	RTS Leading	0	500	1:	of 10msec
n+6	W	RTS Trailing	0	50	0:	of 10msec
n+7	S	Use Dial up	1	2	1: No	Port B
n+8	W	PLC RX Dreg	40001	49999	40001	Holding reg
n+9	W	PLC RX size	1	100	10	Holding reg
n+10	W	PLC TX Dreg	40001	49999	40301	Holding reg
n+11	W	PLC TX size	1	100	10	Holding reg
n+12	S	TX ReadBack	1	2	2: Enabled	Holding reg
n+13	W	Retry count	0	10	3	Dialup
n+14	W	Max comm cnt			1..256 256	Port B
n+15	W	Redial delay	2	120	90	Sec
n+16	T	Modem init	0	0		Port B
n+17	W	Resp timeout	1	100	15	of 100 ms
n+18	W	Scan Delay	1	100	5	of 10 ms

n
Header-Non configurable.

n+1
Define if the Modbus master port is enabled or not. If disabled, the occupied BM registers could be used freely by the application program.

n+2
Define the port baud rate. 300, 600, 1200, 2400, 4800, 9600 baud is possible.

n+3
Define port parity. None, even, odd is possible.
Number of data bit is fixed to 8.

n+4
Second serial port handshake. Select RTS, CTS functions.

n+5
Defines the delay from the RTU is activating RTS to transmission of the first character.

n+6
Defines the delay from transmission of the last character to deactivating RTS.

n+7
Define if dial-up is used or not. If dial-up is enabled, the telephone number and modem init should be entered as well. If disabled they are don't care.

n+8
Defines the first Modbus input data holding register, to be used for data transfer from the slave RTU8 to the Master RTU8 (input words received). This value should normally not be changed.

n+9
Defines the number of Modbus input data holding registers to transfer.

n+10
Defines the first Modbus output data holding register, to be used for data transfer from the master RTU8 to the slave RTU8 (output words transmitted). This value should normally not be changed.

n+11
Defines the number of Modbus output data holding registers to transfer.

n+12
Defines if Modbus output data holding registers is read back, when connecting to a slave. The slaves Modbus output and input holding registers are read (two Modbus requests) when enabled.
Only input registers are read (one Modbus request) when disabled, and this could speed up multi drop system scan time. However write to slave output holding registers are possible (without read back) if needed.

n+13
Defines the number of retries which should be made to the same node before giving up. If dial-up is enabled, the driver will wait the 'redial delay' time before trying again.

n+14
Defines the number of successful communication request before automatic termination of the connection. The value 256 will disable the automatic termination. BM33 contain the actual communication counter value.

n+15
Defines the number of seconds to wait, if a dial up fails, before trying again. This is only used if dial-up mode is enabled.

n+16
Modem initialisation string. Experience shows that many modem manufactures, are making minor changes to the Hayes standard, so it is recommended to consult the actual modems manual to be sure to use the right initialisation string. Up to 60 characters could be entered into the string. The default string is as follows:

AT &C1 &D0 S0=1 E0 V0

n+17
Defines how long time the Modbus master driver waits for a response from the slave, before the master try again.

n+18
Defines additional delay added between repeatedly scan of the same slave (typical point to point solution), if the slave for some reason can't cope with fast scanning (some old third-party equipment have this problem).

SOFTWARE / PROGRAMMING DESCRIPTION

Modbus data buffers
All data read from and written to the substation, is mapped into BM area. The B-CON application program has control of all data, and are able to process and/or log data as needed. All Modbus input data received from the master port are placed by the driver in BM area 600..799 (100 words). All Modbus output data transmitted on the master port are read by the driver in BM area 800..999 (100 words). Up to a maximum of 100 words input and 100 words output could be transmitted between master and slave. At PowerOn reset are BM600..BM999 cleared to zero (if driver is enabled, otherwise not).

There is synchronisation between the driver and the B-CON application. New inputs and outputs are always updated at the end of a B-CON scan, to be sure the data is consistent.

However, to be sure output data is transmitted in a consistent matter, all outputs the user want to change, must be changed before a connection is initiated.

Communication sequence
A RTU8 master to RTU8 slave Modbus communication are typical as follows:

Chose the number to dial in BM31 (if dialup mode, otherwise don't care).
Set Modbus node address in BM34 (1 should be used when connecting to a RTU8 slave in dialup mode)
Set Modbus data buffer offset in BM35.
Initiate connection by setting BM30 to 1 for at least one scan.
Wait until one of the following occur:
1. The com counter >= 1 (output is read back, and input is read), and communication state bits in BM32 are changed to 01 (active outgoing).
2. The BM32.7 is set to '1' (connection suspended).

In case 1 input data (read back output, and input) is transferred successfully.

If no output data should be changed, the connection is terminated by setting BM30 to 2. Wait until BM32.4..5 and BM32.7 = '0', indicating disconnection is done. Set BM30 to 0.

If output data should be changed, new data is written in the output buffer (BM800 and up). Set the transmit bit (BM30.4) to '1'. The driver transmits the outputs, and when successful, bit BM30.4 is set to '0'. The connection is terminated by setting BM30 to 2. Wait until BM32.4..5 and BM32.7='0', indicating disconnection is done. Set BM30 to 0.

The Modbus data buffers are now consistent, and input data could be processed.

In case 2 connection has failed and no successfully data transfer has taken place. In dialup mode this could take some time before the bit is set because of the redial delay (90 sec default) between retry. The suspend bit is automatically set to '0' if a new connection is initiated, or an incoming call is received.

The connection cycle is now finished, and a new one could be started.

The connection / disconnection procedures are the same no matter if dial-up is enabled or not. The only difference is a telephone number is needed in dialup mode.
An incoming call from a slave station is automatically detected and answered by the firmware.
The application program detects the communication by changes in communication state bits (BM32.4..5 = 10 for incoming call) or com counter. Input data is read by the driver, and placed in buffer area from

BM600 and up. When all inputs are updated (com counter >= 1) the master terminates the connection.
To determinate which substation the call came from, the application program must check the station number, which normally are located in the first Modbus input word of the received data.

Note! Inputs read during an incoming call, are always placed from address BM600 and up, no mater BM35 (Modbus offset) value. No output values could be send as well. If the master wants to changes some outputs, it must initiate a call to the slave.

Note!
When the master connect to a slave, current value of the slave output holding registers are automatically read back by the master, and stored in driver output BM area 800..999, for further use. When connecting to equipment where no output should be send, the number of Modbus output data holding registers to transfer could be set to zero in the RTU configuration, to disable the automatic output holding register read back.

Error handling
When the driver querying I/O data from the slave station, it will default wait up to 1.5 sec for the response. If it doesn't get any response the driver will try again after the response timeout. After 3 requests without any response, the BM32.7 bit is set to '1', and connection is terminated, when in dialup mode. In non dialup mode connection is maintained, and the driver continue to poll the slave station.
If the slave station don't respond to a request, the masters current input (BM600..BM799) is frozen until communication is re-established.

B-CONW code example
Below is listed a simple B-CONW code examples for a point-to-point system with a RTU8 Master and a RTU8 Slave. The code define I/O transfer in Null-modem mode (e.g. connected direct via cable or radio modems).

Code in RTU8 Master

```
/           RTU8 with Master Port
/           Radio Communication between 1 RTU Master and 1 RTU Slave

/           Master:     UCR-28IO/RTU120.D1
/           Slave1:     UCR-28IO/RTU21.D1
/           Radio:      UCW-10.912


#Define AI_CNT 4          /Set Number of Analog Input Channels (1 = 1AI)
#Define DI_CNT 1          /Set Number of Digital Input Channels (1 = 16DI)
#Define YI_CNT 5          /Set Number of Transfer Registers (AI_cnt + Di_cnt + 2)


#Define     Init m38.0
#Define     ModbusCmdReg bm30     /Modbus Command Register
#Define     CommStateOut m32.4    /Output Holding Registers are Updated

#Define     Slave_DI0-15 wm604    /Digital Input from Slave data
#Define     Slave_AI0 wm606                 /First Analogue Input
#Define     Slave_AI1 wm608                 /Second Analogue Input
#Define     Slave_AI2 wm610                 /Third Analogue Input
#Define     Slave_AI3 wm612                 /Fourth Analogue Input
#Define     SendOutput m30.4
#Define     SlaveAdr bm34                   /Adress of
#Define     Connected m38.1

/************** IO Mapping ************************************************************ /
/ Slave DI are read on      wm604     (DI 0-15)
/
/ If more Slave DI´s these will be available from
/                                     wm606     (DI 16-31)
/                                     wm608     (DI 32-47)
/
/ AI are Located on First available address following the DI´s
/
/                                     wm606     (AI0)
/                                     wm608     (AI1)
/                                     wm610     (AI2)
/                                     wm612     (AI3)
/
```

```
/*************************************************************************

Main:
        ld Init
        jmpc Start
        mov bc2 Slaveadr                    / Connect to slave adr. 2
        mov bc0 bm35
        mov c1 Init
        jmp Slave2Master

Start:
        ld connected            /If Connected with Slave
        jmpc online             / Wait for All I/O Updated before
        mov bc1 ModbusCmdReg    /Connect to selected slave
        mov c1 Connected
        jmp Slave2Master

Online:
        ld CommStateOut                     / Is Connection established and all I/O
Updated
        jmpc Master2Slave                   / If so, New Output can be send
        jmp Slave2Master

Master2Slave:
        mov wi0     wm800                   / Copy DI0-15 on Master to Slave
        mov wi2000 wm802                    / Copy AI0 on Master to Slave
        mov wi2002 wm804                    / Copy AI1 on Master to Slave
        mov wi2004 wm806                    / Copy AI2 on Master to Slave
        mov wi2006 wm808                    / Copy AI3 on Master to Slave

        mov c1 SendOutput

Slave2Master:
        Mov Slave_DI0-15 wo0    / Copy DI0-15 from slave to Master Output
        Mov Slave_AI0 wo2000    / Copy AI0 from slave to Master Output
        Mov Slave_AI1 wo2002    / Copy AI1 from slave to Master Output
        Mov Slave_AI2 wo2004    / Copy AI2 from slave to Master Output
        Mov Slave_AI3 wo2006    / Copy AI3 from slave to Master Output

ep
```

Code in RTU8 Slave

```
/           P2P Radio Slave
/           Radio Communication between 1 RTU Master and 1 RTU Slave

/           Master:     UCR-28IO/RTU120.D1
/           Slave1:     UCR-28IO/RTU21.D1
/           Radio:      UCW-10.912


#Define AI_CNT 4          /Set Number of Analog Input Channels (1 = 1AI)
#Define DI_CNT 1          /Set Number of Digital Input Channels (1 = 16DI)
#Define YI_CNT 5          /Set Number of Transfer Registers (AI_cnt + Di_cnt)
#Define YO_CNT 5          /Set Number of Transfer Registers (AI_cnt + Di_cnt)

/************************************************************************* / Master DI are read on
wi6000      (DI 0-15)
/
/ If more Slave DI´s these will be available from
/                                     wm6002    (DI 16-31)
/                                     wm6004    (DI 32-47)
/
/
/ AI are Located on First available address following the DI´s
/
/                                     wi6002    (AI0)
/                                     wi6004    (AI1)
/                                     wi6006    (AI2)
/                                     wi6008    (AI3)
/*************************************************************************
main:
 mov wi0 wo6000          / Copy DI0-15 on Slave to Master
 mov wi6000 wo0          / Copy DI0-15 on Master to Slave output
 mov wi2000 wo6002       / Copy AI0 to Master
 mov wi2002 wo6004       / Copy AI1 to Master
 mov wi2004 wo6006       / Copy AI2 to Master
 mov wi2006 wo6008       / Copy AI3 to Master

 mov wi6002 wo2000       / Copy AI from Master to Slave Analogue output
 mov wi6004 wo2002       / Copy AI from Master to Slave Analogue output
 mov wi6006 wo2004       / Copy AI from Master to Slave Analogue output
 mov wi6008 wo2006       / Copy AI from Master to Slave Analogue output

 log m17.1 bc31 wi0 bc1
 ep
```

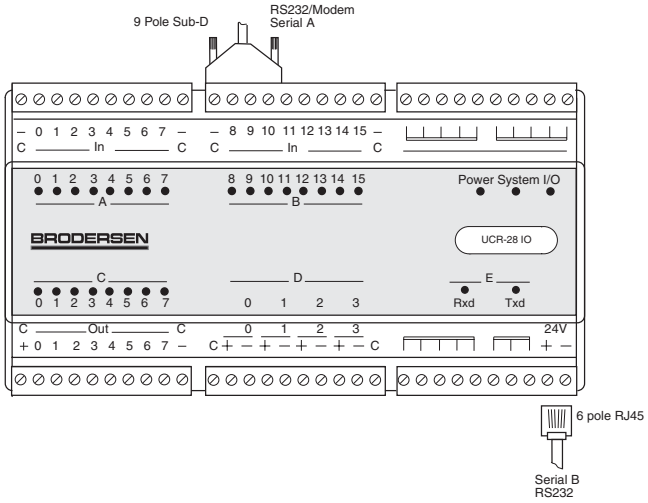
INTRODUCTION

The RTU8 with a mirrored Modbus slave protocol on the 2nd serial port B, 6 pole RJ45 connector. Provide all the functions as on the primary serial port. It means that the 2nd serial port provide access to communicate with the RTU8 data base, access I/O and upload log via IOTOOL32Pro. Also setting up the RTU8 configuration table and down load B-CONW application program is possible equal the primary port.

The additional serial port does not support dial-up features, i.e it can not be connected to the PSTN network through a modem.
The port is configurable from 300 to 9600 baud, 8 data bits, none/even/odd parity.

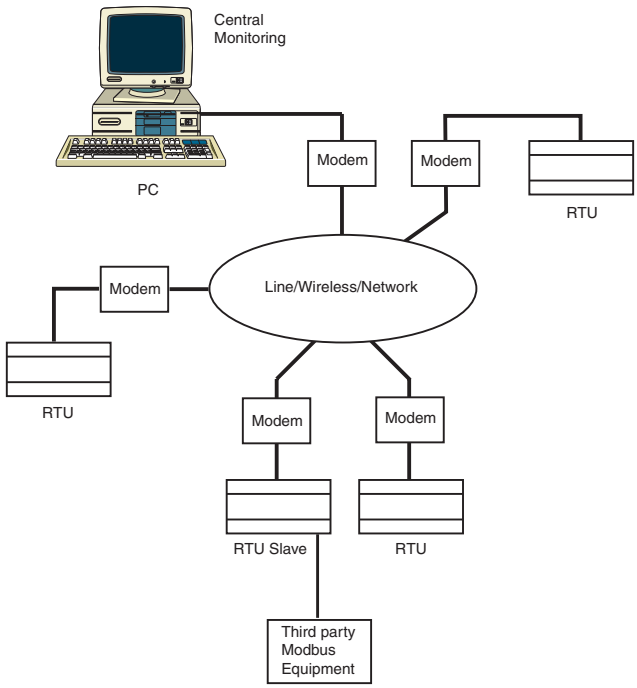
For additional RTU8 data, please see the general RTU8 data sheet.

RTU8 with two serial ports

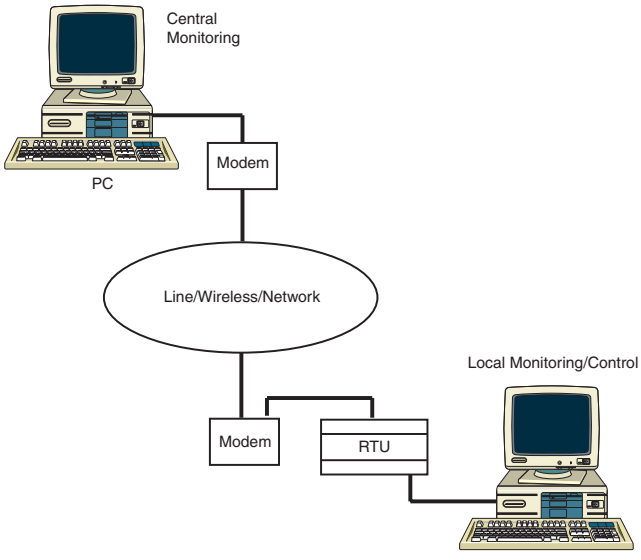


TYPICAL APPLICATIONS

Multi point application



RTU8 with local terminal application



VERSIONS/ORDERING CODE

The code for indicating the presence of the second serial port on the RTU8 module is added to the normal RTU8 ordering/type code.

UCR-xxx/RTU2xx.Dx

Type

UCR	UCR-xxx/RTU

Serial port options

No port	blank
Port with Mirrored Modbus Slave	2

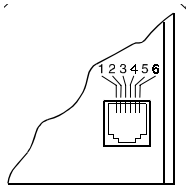
Additional Serial Port; Mirrored Modbus Slave Interface
RTU8

TECHNICAL DESCRIPTION

General
The Modbus Slave driver is implemented as a separate task in the RTU8 firmware, and is a complete mirror of the primary Modbus Slave interface. The driver will operate at the second RS232 port of the Dallas 80C320 controller in the RTU8. The driver occupies a number of BM registers. BM 30..35 are reserved for driver control.
The Modbus Slave protocol on the RTU8 second serial port is tolerant to odd characters added to the Modbus frame e.g. generated when passing through a radio link connection.

Hardware/serial interface
The second serial RS232 port is based on a 6 pole RJ45 connector, and therefore not full blown. Look in the configuration table for adjusting hardware handshake options The data signals are connected to the additional UART on the Dallas 80C320 micro controller.

The RS232 port provide the following signals:



Pin 1	SG	Signal Ground	Electronic GND
Pin 2	RTS	Request To Send	Output
Pin 3	RX	Receive data	Input
Pin 4	TX	Transmit data	Output
Pin 5	CTS	Clear To Send	Input
Pin 6		Shield	Module housing

Communication mode
The Mirrored Modbus Slave port do only support NullModem mode. The terms 'Null Modem Mode' or 'non dial-up' defines systems where no dialling is necessary to establish a connection between two stations. However some kind of modem could still be involved (e.g. radio modems).
In this mode are all communication initiated by the master station. The slave station is not able to initiate a connection. The RTS signal is active when a Modbus reply is transmitted to a slave, and inactive in receive mode. This is used in multi-drop systems (e.g. radio modems).

Command registers
BM registers 30...35 are reserved as interface between B-CON application program and the driver for the additional serial port.
Only BM33 are used in this driver implementation.

Communication counter register (BM33 input / output)
Used to monitor the I/O data transfer. The counter is incremented whenever a valid Modbus request is received.

CONFIGURATION

A number of configuration fields are provided. The field values are changed and downloaded into the RTU8 flash memory, using the IOTOOL 32 configuration menu. The fields are used for values which are programmed once when setting up the module (e.g. parity and baud rate settings).
The following fields are provided to control the additional Modbus slave port driver in addition to the already existing fields in the configuration menu.

Configuration table

Product		**UCR-28IO/RTU**		**Software version 1.00**	
Field	**Type**	**Description**	**(Min..Max)**	**Current value**	**Unit**
n	S	Port B cfg	1..1	1	Modbus
n+1	S	Baud Rate	1..6	6 9600	Port B
n+2	S	Parity	1..3	1 None	Port B
n+3	S	Handshake	1..4	3 RTS On/off	PortB
n+4	W	RTS Leading	0..500	1	of 10msec
n+5	W	RTS Trailing	0..50	0	of 10msec

n
Header-Non configurable.

n+1
Defines the port baud rate. 300, 600, 1200, 2400, 4800, 9600 baud is possible.

n+2
Defines port parity. None, Even, Odd is possible. The caracter length is fixed at 8 data bits.

n+3
Second serial port handshake. Select RTS, CTS functions.

n+4
Defines the delay from the RTU is activating RTS to transmission of the first character.

n+5
Defines the delay from transmission of the last character to deactivating RTS.

INTRODUCTION

The second serial port (port B, 6 pole RJ45 connector) provide a Modbus Slave protocol functionality. Only a subset of the Modicon Asynchronous Link Protocol is implemented. The Modbus RTU framing mode is used. The following two Modbus commands are used: Read Holding Registers (command 3), Preset Single Register (command 6) and Preset Multiple Registers (command 16).

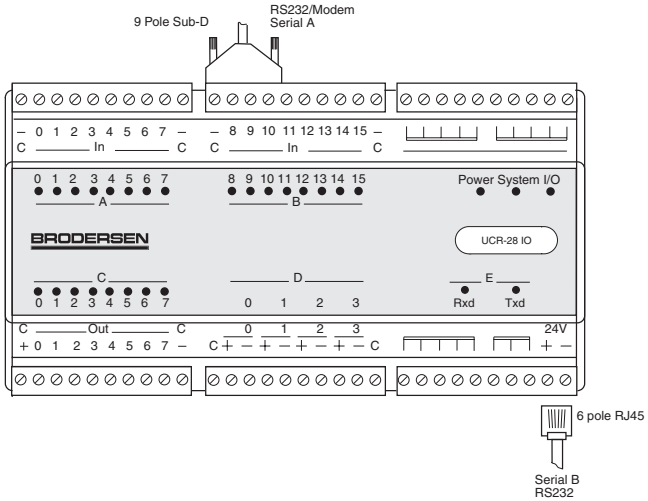
The Modbus Slave protocol on the RTU8 is tolerant to odd characters added to the Modbus frame e.g. generated when passing through a radio link connection.

The Modbus Slave port support direct linked connections with or without hardware handshake. All access from the application to the Modbus registers are controlled with a B-CONW application. Log is not accessible through this port.

The port is configurable from 300 to 9600 baud, 8 data bit, non/even/odd parity.

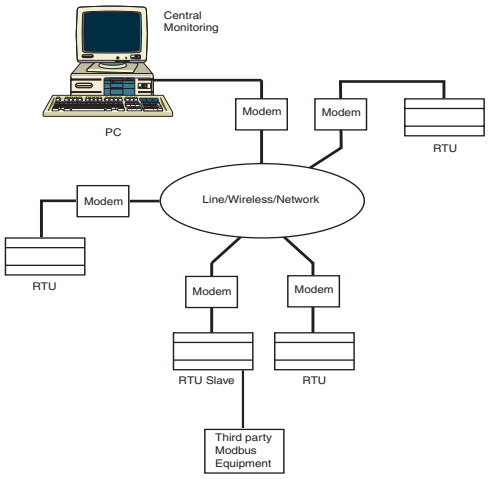
For additional RTU8 data, please see the general RTU8 data sheet.

RTU8 with two serial ports



TYPICAL APPLICATION

RTU connected to 3rd party equipment



VERSIONS/ORDERING CODES

The code for indicating the presence of the second serial port on the RTU8 module is added to the normal RTU8 ordering/type code.

UCR-xxx/RTU3XX.Dx

Type

UCR	UCR-xxx/RTU

Serial port options

No port	blank
Port with Modbus Slave protocol	3

TECHNICAL DESCRIPTION

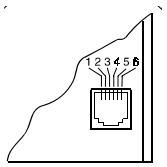
General
The Modbus slave driver is implemented as a separate task in the RTU8 firmware. The driver will operate at the second RS232 port of the Dallas 80C320 controller in the RTU8. If the driver is enabled in the configuration menu, it occupies a number of BM registers. BM 30..33 are used for driver control, and BM1000..1499 are used for Modbus data buffers. If the driver is disabled all the BM registers are free for application use.

Hardware/serial interface
The second serial RS232 port is based on a 6 pole RJ45 connector, and therefore not full blown. One input, and one output handshake signals are provided. The data signals are connected to the additional UART on the Dallas 80C320 micro controller.

The RS232 port provide the following signals:

Pin 1	SG	Signal Ground	Electronic GND
Pin 2	RTS	Request To Send	Output
Pin 3	RX	Receive data	Input
Pin 4	TX	Transmit data	Output
Pin 5	CTS	Clear to send	Input
Pin 6		Shield	Module housing

Baudrate and handshake mode is setup in the configuration menu.



Functionality
This port/driver can be used in applications where point to point, multidrop or radio modem are used. All communication must be initiated by the master station. The RTU slave station only reacts on incoming requests. Only Modbus requests with the correct substation addressing are responded to by the RTU slave. However, broadcast messages, i.e. address 0 (zero) is also recognized.
Different modes of handshake operation is available and can be selected according to the installation (RTS always Off, RTS always On, RTS On/Off - i.e. On in transmit mode and Off in receive mode or RTS/CTS flow control for multidrop systems).
It is also possible to adjust the leading- and trailing time of the RTS signal (may be required in radio or wired modem systems).
Any further hardware handshake signals required by the connected equipment (Modbus master or multi drop modem system) must be dealt with externally (e.g. connect DSR and DTR locally).

Command registers
A number of BM registers are used by the B-CONW application program, in conjunction with some configuration registers, to control the Modbus slave driver. The next will describe the registers in table format, followed by a description in text.

The BM registers are defined as follows:

Register	Description
BM30	Modbus Command register m30.0 0 = Disable access to Holding Registers 1 = enable access to Holding Registers
BM31	Not used
BM32	Input m32.0..2 Not used m32.3 Not used m32.4 Set when "Read Holding Register" query is received. m32.5 Set when "Write Holding Register" query is received.
BM33	Communication counter (Input)

Please see the description for each register below:

Command register (BM30 output)
Used to enable and disable access to Holding Registers.
When access is disabled, any Modbus Request will be responded with the Exception response “Busy, rejected message”.

Communication state register (BM32.4..5 input)
Used to monitor the state of a connection.

00:	The system is idle. No Modbus Requests executed
BM32.4:	Set by driver whenever a “Read Holding Registers” requests is executed. Can be cleared by B-CONW application program.
BM32.5:	Set by driver whenever a “Write Holding Register” request is executed. Can be cleared by B-CONW application program.

Communication counter register (BM33 input / output)
Used to monitor the I/O data transfer. The counter is incremented in every Modbus cycle. A cycle is defined as a reception of a valid Modbus Request plus transmission of the response.
Can be cleared by B-CONW application program.

CONFIGURATION

A number of configuration fields are provided. The field values are changed and downloaded into the RTU8 flash memory, using the IOTOOL 32 configuration menu. The fields are used for values which are programmed once when setting up the module (e.g. protocol parameters and baud rate).
The following configuration table fields are provided to control the Modbus Slave port driver in addition to the already existing fields in the configuration menu.

Configuration table

Field	Type	Description	Min	Max	Default value	Unit
n	S	Port B cfg				Modbus
n+1	S	Use Driver	1..2		No	Modbus
n+2	S	Slave addr.	1..247		1	
n+3	W	PLC Dreg	40001..49999		40001	1.st H-Reg.
n+4	W	Dreg size	1..250		10	no of H-Reg.
n+5	S	Baud rate	1..6		9600	Port B
n+6	S	Parity	1..3		None	Port B
n+7	S	Handshake	1..4		RTS off	PortB
n+8	W	RTS Leading	0..500		1	of 10msec
n+9	W	RTS Trailing	0..50		0	of 10msec

n
Header - Non configurable.

n+1
Define if the Modbus Slave driver port is enabled or not. If disabled, the occupied BM registers could be used freely by the application program.

n+2
Define the Modbus Slave device address that the RTU8 shall respond to. Valid range: 1-247. NB: Broadcast messages (address=0) is recognized.

n+3
Define the first Modbus data holding registers to be used for data exchange between the slave RTU8 and the master RTU, Valid range: 40001 - 49999.

n+4
Define the number of Modbus Holding registers reserved for data exchange (inputs + outputs). The last accessible Modbus Holding register will be the value of field n+5 plus the value of this field minus Valid range: 0 - 250.

n+5
Define the port baud rate. 300, 600, 1200, 2400, 4800, 9600 baud is possible.

n+6
Define port parity. None, even, odd is possible.
Number of data bit is fixed to 8.

n+7
Select handshake/flow control mode.
1: RTS off 3: RTS on/off
2: RTS on 4: RTS/CTS

n+8
Defines the delay from the RTU is activating RTS to transmission of the first character. Used in multi drop communication and radio links. Valid range: 0-5000ms.

n+9
Defines the delay from transmission of the last character to deactivating RTS. Used in multi drop communication and radio links. Valid range: 0-500ms.

SOFTWARE / PROGRAMMING DESCRIPTION

Overview
The Modbus slave driver is implemented as a separate task in the RTU8 firmware. The driver will operate at the second RS232 port of the Dallas 80C320 controller in the RTU8. If the driver is enabled in the configuration menu and it will occupy a number of BM registers.
BM 30..33 are used for driver control, and depending on the value of the number of Holding Registers defined in field n+5 (see Table 1 Configuration fields) BM 1000 and up to 1499 are reserved as Modbus Holdings registers. If the driver is disabled all the BM registers are free for application use.

The driver will use RX, TX, CTS and RTS (and GND) signals to communicate with Modbus master, either in a Point–To-Point connection or by use of some kind of multi drop connection.
If a multi drop connection is being used; the RTS/CTS handshaking signals must be used to control access to the line.

Two Modbus commands are implemented: Read Holding Registers (command 3), and Preset Multiple Registers (command 16).

Modbus Holding Registers
All received and transmitted data to the master station are mapped into the BM registers starting with BM1000. The B-CONW application program has access to of all the data, and are able to process and/or log data as needed. It is the B-CONW programmers' job to interpret the data and make the B-CONW application program consistent with the Modbus master application.

The Modbus master can read any "Holding Register" and write to any "Holding Register" as long as it is within the defined area. The only limitation is the Modbus Protocol. Eventually two or more request must be issued to access the whole area. At Power On Reset the "Holding Register" area are cleared to zero (if driver is enabled, otherwise not).

There is synchronisation between the driver and the B-CONW application. New inputs and outputs are always updated at the end of a B-CONW scan, to be sure the data is consistent.
To be sure that output data is transmitted in a consistent matter, all inter related outputs must be defined within one B-CONW scan.
Note! Requests are processed only in the intervals when B-CONW is idle, i.e. waiting for the next scan.

Communication sequence
In the typical application, the RTU slave is connected to a Modbus master with a point-to-point connection or a multi drop line (radio or wired). It waits for a Modbus Request with the correct slave device address to be received. In the interval between two B-CONW scan the request will be serviced and a response is built in the transmit buffer. After having serviced the requests the RTU slave sends a response.

Error handling
Operation errors are those involving illegal data in a message or difficulty in communication with the master. One of the following possible events can occur from the master's query:

1. If the slave device receives the query without a communication error, and can handle the query normally, it returns a normal response.
2. If the slave does not receive the query due to a communication error, no response is returned.

3. If the slave receives the query but detects a communication error (parity or CRC), no response is returned.

4. If the slave receives the query without a communication error, but cannot handle it (e.g. the request is to read a non-existing register) the RTU slave sends an exception response to the master, consisting of the slave address, the function code and error check fields.

5. If a Modbus query addresses Holding Registers both inside - and outside the area defined by field n+5 plus field n+6 (see Configuration fields) the hole query is discarded and a exception response “ILLEGAL DATA ADDR” is send.

6. If a Modbus request contains a function code which is not supported, the exception response “ILLEGAL FUNCTION” is send.

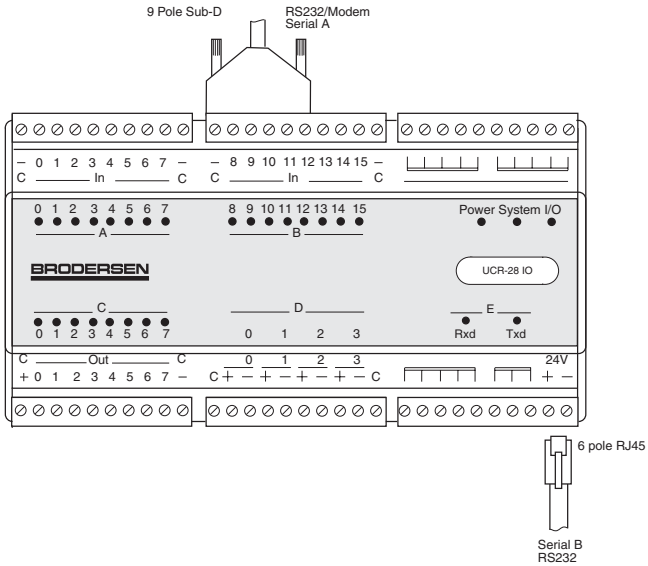
INTRODUCTION

The implemented gateway driver is a transparent driver totally controlled by the internal B-CONW program. It can handle simple serial communication protocols for reading and writing data.

The port is configurable from 300 to 9600 baud, 8 data bit, non/even/odd parity.

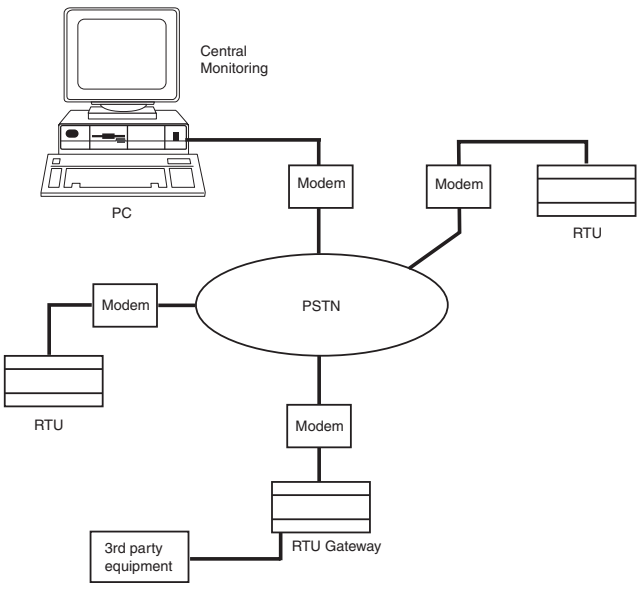
For additional RTU8 data, please see the general RTU8 data sheet.

RTU8 with two serial ports



TYPICAL APPLICATION

Typical RTU8 Gateway Application



VERSIONS/ORDERING CODES

The code for indicating the presence of the second serial port on the RTU8 module is added to the normal RTU8 ordering/type code.

UCR-xxx/RTUX21.Dx

Type
UCR — UCR-xxx/RTU

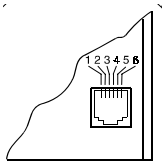
Serial port options
No port — blank
Transparent to B-CON — 4

TECHNICAL DESCRIPTION

General
The transparent driver is implemented as a separate task in the RTU8 firmware. The driver will operate at the second RS232 port of the Dallas 80C320 controller in the RTU8. If the driver is enabled in the configuration menu, it occupies a number of BM registers. BM600..999 are used for data buffers. If the driver is disabled all the BM registers are free for application use.

Hardware/serial interface
The second serial RS232 port is based on a 6 pole RJ45 connector, and therefore not full blown. Only one input, and one output handshake signals are provided. The data signals are connected to the additional UART on the Dallas 80C320 micro controller.

The RS232 port provide the following signals:



Pin 1	SG	Signal Ground	Electronic GND
Pin 2	RTS	Request To Send	Output
Pin 3	RX	Receive data	Input
Pin 4	TX	Transmit data	Output
Pin 5	CTS	Data Carrier Detect	Input
Pin 6		Shield Module housing	

The driver will use RX, TX, CTS and RTS (and GND) signals to communicate.
Handshake is setup in configuration table.

Additional Serial Port; Transparent Gateway
RTU8

Gateway driver general
The RTU8 gateway driver interface can be used for simple protocols that can be implemented in a B-CONW program. The serial driver is a simple general purpose driver that can handle exchange of commands, requests and responses. The driver can be configured through a number of parameter settings at the physical layer- and data link layer level. The application specific protocol is to be programmed in B-CONW.

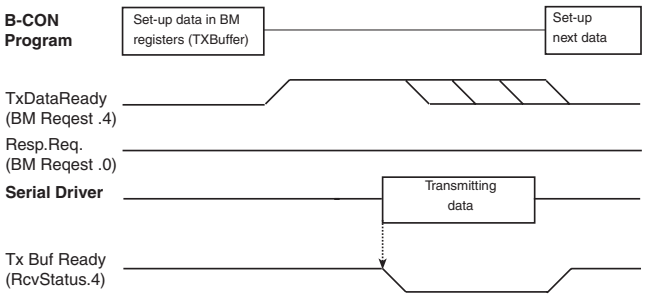
Serial driver – BCONW interface
The BM registers used for the serial gateway driver are fixed.

All input data from the serial gateway driver is handled in BM600 to BM799. BM600 are used for receive status flags as data ready, errors, buffer overrun etc., and BM601 defining the number of received data bytes. Max. receive data buffer is 126 bytes. BM602 to BM7xx will contain the received data bytes.

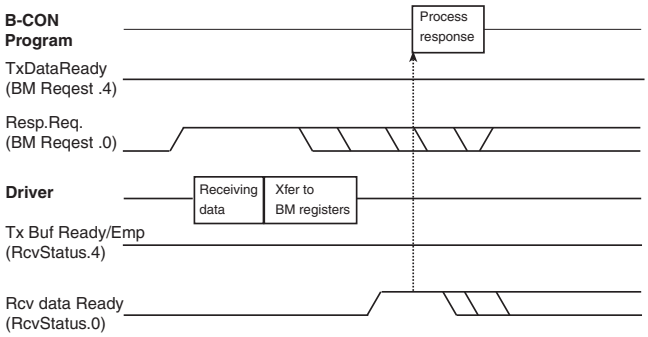
Request/data output from BCONW to the serial driver is located in BM800 to BM999. BM800 and BM801 are used for defining response/ recive requests, data transmit ready and also defining the number of bytes of data to transmit from the transmit buffer. Again the max. transmit data buffer is 126 bytes. BM802 and upwards is the data bytes to transmit placed.

Serial Driver / BCONW data exchange diagram

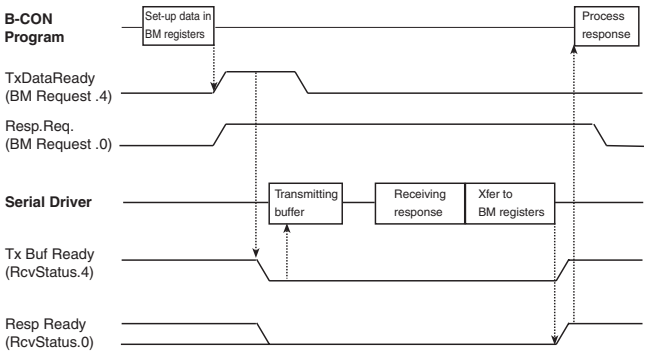
Figur 1 Sending only



Figur 2 Receive Only



Figur 3 Send data – Get response



Notes.
Requests from the BCONW program is detected immidiatly after execution of a BCONW scan .
If the serial driver is idle at this point , a new serial driver session is initiated and status flags in BCONW (RcvStatus) are cleared. While the driver is busy (transmitting or receiving , new requests from the application program is not registered

T3 gateway driver communication registers
The BM registers used for the serial driver are fixed.
BM RcvStatus (input data to B-CONW from serial driver) is located at BM 600. BM Request (output data from B-CONW to serial driver) is located at BM800

The serial driver has a 126 byte receive buffer and a 126 byte transmit buffer.

Register	Description
BM600	RcvStatus Receive Status Flags, (from serial driver to appl. Program)
m600.0	Receive/Resp data ready
m600.1	Receive Error (Frame, Parity, Timeout)
m600.2	Receive Response Timeout
m600.3	Receive Buffer overrun
m600.4	Transmit buffer ready
m600.5	reserved
m600.6	CTS timeout (RTS/CTS handshake)
m600.7	reserved
BM601	RcvCnt - Defines the number of received data bytes / number of bytes in receive buffer
BM602 to BM728	RcvBuffer - Receive data (bufferlength = 126 bytes)
BM800	Request - B-CONW serial I/O request flag (from application program to serial driver)
m800.0	Rsp/Rcv message request-
m800.1	not used
m800.2	not used
m800.3	not used
m800.4	transmit data ready
m800.5	not used
m800.6	not used
m800.7	reserved
BM801	TxCnt - Number of bytes to transmit / number of bytes in transmit buffer
BM802 to BM928	TxBuffer - Transmit data (bufferlength = 126 bytes)

CONFIGURATION

A number of configuration fields are provided. The field values are changed and downloaded into the RTU8 flash memory, using the IOTOOL 32 configuration menu. The fields are used for values which are programmed once when setting up the module (e.g. telephone numbers, and baud rate).
The following fields are provided to control the Modbus master port driver in addition to the already existing fields in the configuration menu.

Configuration table

Field	Type	Description	Min Max	Default value	Unit
n+1.	S	Use Driver	(1..2)	: Yes	T3 driver
n+2.	S	Serial Port	(1...2)	RS232	RS232, RS485
n+3.	S	Baudrate	(1...7)	9600	300...19K2
n+4.	S	Data Bits	(5...8)	8	5, 6, 7 or 8
n+5.	S	Parity	(1...3)	None	None,even,Odd
n+6.		Handshake	(1...4)	RTS off	Hdw. flow
n+7.		RTS Leading	(0..500)	: 3	of 10 msec
n+8.		RTS Trailing	(0..50)	: 1	of 10 msec
n+9.		Resp. Timeout	(1...255)	: 20	x100ms(=255forever)
n+10.		Inter Char timeout	(1...99)	: 2	x10ms

Field	Description
n+1	Enable/Disable T3 driver on serial gateway port
n+2	Specifies the physical interface for T3 driver. Valid range: only RS232
n+3	Defines baudrate for serial gateway port. Valid range: 300 – 19k2 bps
n+4	Defines bit length of characters. Valid Range: 5 – 8 bit
n+5	Defines parity bit generation and detection. Valid range : None, Even, Odd
n+6	Defines handshake protocol. Valid range: RTS Off, RTS On, RTS On/Off, RTS/CTS
n+7	Defines the delay from the driver activates RTS to transmission of first character. Used in multidrop - and radio links. Valid range: 0 – 5000 ms
n+8	Defines the delay from transmission of the last character to deactivation RTS. Valid range: 0 – 500 ms
n+9	Defines the maximum time the driver will wait for a response after detection of “Receive Request”. Valid range : 1 – 255 x100ms (255 equals “Forever”)
n+10	Defines this maximum allowed gap between two characters in the received message. A gap is used as an end-of-message indication. Valid range: 20-990ms.