# Manual AMB8425-M

Version 1.6

SW-V1.6

AMBER wireless GmbH

Albin-Köbis-Straße 18
51147 Köln
Tel. +49 (0) 2203-6991950
Fax +49 (0) 2203-459883
E-Mail info@amber-wireless.de
Internet http://www.amber-wireless.de

## Table of Contents

## Abbreviations

| | | |
|---|---|---|
| CS | Checksum | |
| PS | Pin select | Operating mode selection via pins |
| UI | User interface | Serial interface to the host |

## 1 Summary

The wireless M-bus radio module AMB8425-M was designed as a subcomponent group for wireless reading of meters. It complies with the requirements of EN13757-4:2005 (wireless M-bus; see [1]) and can be used wherever such radio communication is needed.

The following operating modes are supported:

- S1 / S1-m / S2
- T1 / T2 (meter/ other)
- R2

A serial interface (UART, optionally SPI) whose data rate and format can be adjusted flexibly is available for the communication with the host system.

All transfer parameters listed in the standard (coding methods, chip rate, carrier frequency, preamble length, etc.) can be adjusted flexibly via the user interface.

The module is suitable for use in the meter, but can also be used in a repeater or concentrator.

## 2 Communication with the Customer Application

Both the transparent, buffered data transfer (with and without addition and evaluation of block 1) as well as the device configuration take place via the serial user interface.

### 2.1 Transparent, Buffered Data Transfer

#### 2.1.1 Wireless Transmission of Data

In this mode, up to 128 bytes of payload data are transmitted to the radio module via the serial interface and initially buffered. A length byte must be placed in front of the actual payload data (see Figure 1).

| | |
|---|---|
| Length byte (n) | Payload data n bytes |

**Figure 1: Reception of data to be sent via radio transmission with check of block 1**

After all data have been received, the radio transmission takes place according to the operating mode configured according to [1]. Optionally, AMB8425-M can adopt the content of the first block (C-field, M-field, A-field) from the non-volatile memory and insert it in the radio telegram (see Figure 1); otherwise, the format according to Figure 2 is used (factory state; see parameter `MBUS_Bl_ADD_Disable`).

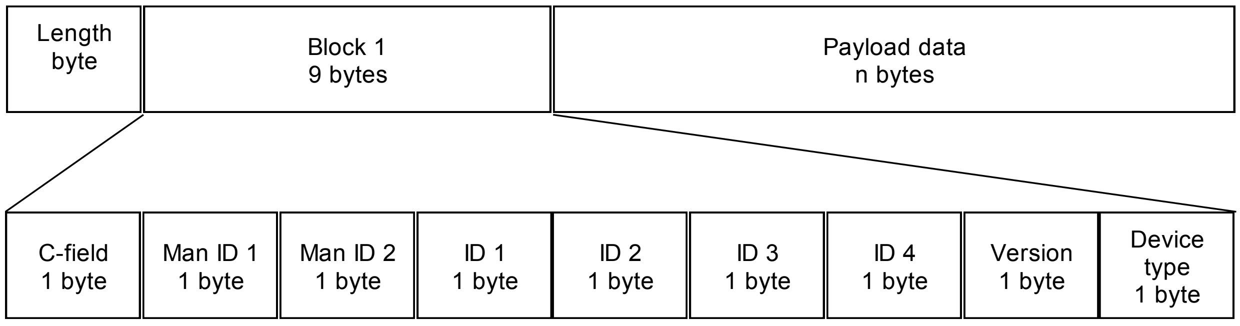


**Figure 2: Reception of data to be sent via radio transmission including block 1**

### 2.1.2 Wireless Reception of Data

The output of data also takes place with length information placed in front. Additionally, a measure for the connection quality can be made available in the form of a field strength value (see `RSSI_Enable`).

Optionally, the first block (M-field and A-field) can be compared with the parameters in the non-volatile memory (see `MBUS_Bl_ADD_Disable`).

If the check is enabled, the received data will only be issued as shown in Figure 3 if the M-field and the A-field (see sections `MBUS_Bl_ManID1`) correspond to the values stored in the radio module.

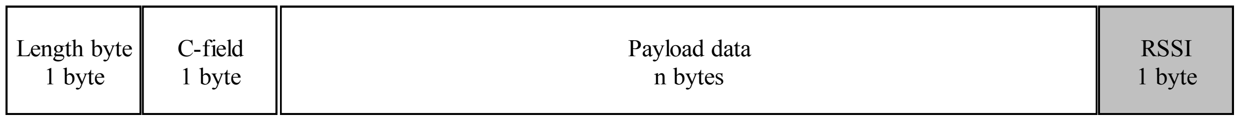


**Figure 3: Output of data received via radio transmission with check of block 1**

If the check is disabled, the received data will always be fully issued (see Figure 4).

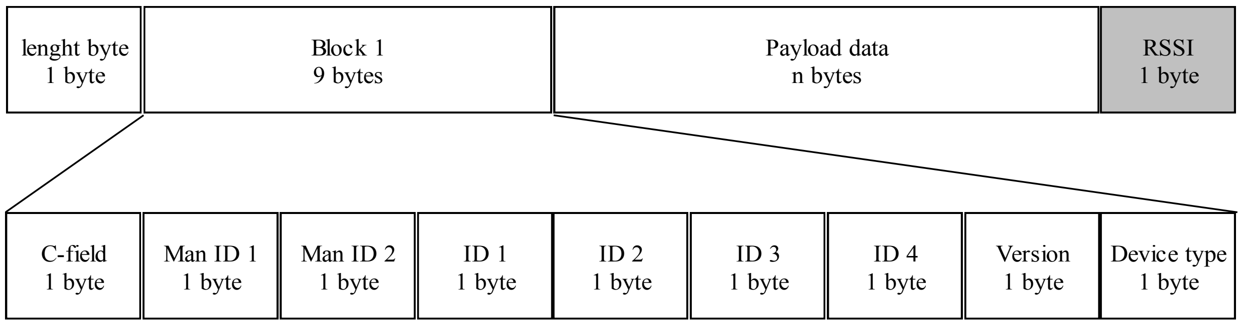


**Figure 4: Output of data received via radio transmission without check of block 1**

## 2.2 Command Mode

In this operating mode, the communication with the module is achieved by using predefined commands. For example, these commands can be used to set and read operating parameters and to execute special functions (e.g., software reset or reset of the operating parameters to the factory state).

The telegram consists of the start character (0xFFh) followed by a command byte and information about the length of the subsequent data. A checksum (XOR operation of all preceding characters) must be transmitted as the last byte.

| | | | | |
|---|---|---|---|---|
| Start character 0xFFh 1 byte | Command 1 byte | Length information(n) 1 byte | Data n bytes | Checksum 1 byte |

**Figure 5: Serial interface in the command mode**

Table 1 provides an overview of the valid commands.

| Command | Value [hex] | Description |
|---|---|---|
| `CMD_DATA_REQ` 6.1.1 | 0x00 | Send data |
| `CMD_DATARETRY_REQ` 6.1.2 | 0x02 | Resend data previously sent to the module |
| `CMD_DATA_IND` 6.1.3 | 0x03 | Received Data |
| `CMD_RESET_REQ` 6.1.4 | 0x05 | Software reset |
| `CMD_SET_CHANNEL_REQ` 6.1.5 | 0x06 | Select channel |
| `CMD_SET_REQ` 6.1.6 | 0x09 | Write parameters of the non-volatile memory |
| `CMD_GET_REQ` 6.1.7 | 0x0A | Read parameters of the non-volatile memory |
| `CMD_SERIALNO_REQ` 6.1.8 | 0x0B | Read serial number |
| `CMD_RSSI_REQ` 6.1.9 | 0x0D | Read current RSSI value |
| `CMD_SETUARTSPEED_REQ` 6.1.11 | 0x10 | Select transfer speed of the user interface |
| `CMD_FACTORYRESET_REQ` 6.1.12 | 0x11 | Reset module to factory settings |

**Table 1: Overview of the commands**

Every command of the type "request" is followed by an acknowledgement by the module in the form of a message of the type "confirm". For this purpose, the module adds the bit 0x80 to the respective command byte (e.g. "0x0D" becomes "0x8D", "0x11" becomes "0x91") and returns it along with the respective data or status information.

## 3 Radio Telegram

As mentioned in section 2.1.1, the content of the first block from the non-volatile memory is added to the payload data to be sent (this can be disabled). Moreover, the checksum—which is required by the standard—is computed for each block and added.

The fields marked in green are added or recomputed by the module.

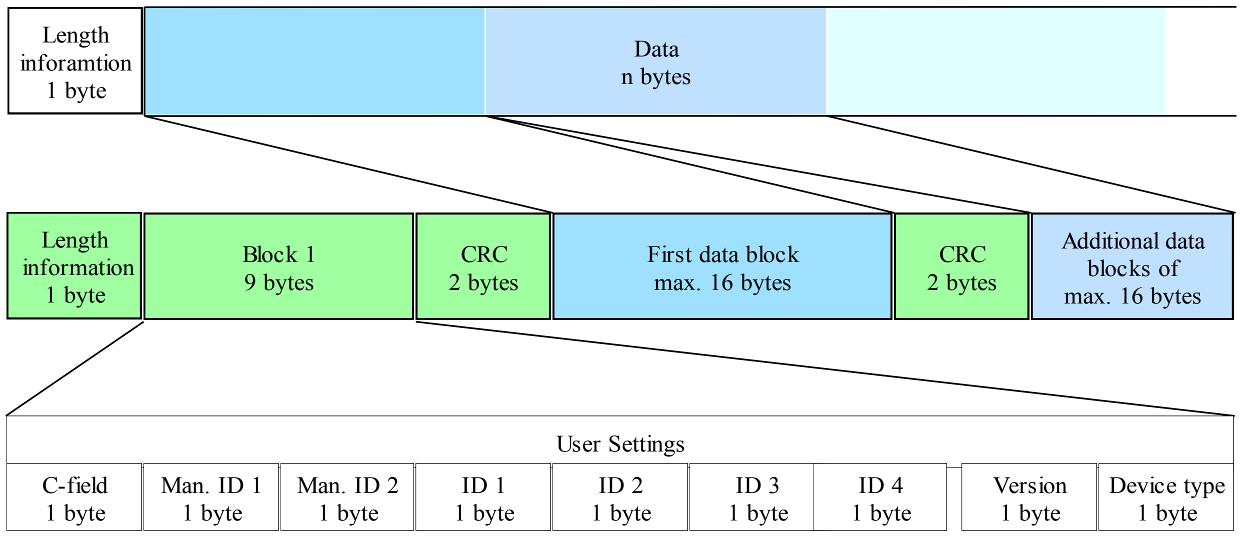


**Figure 6: Radio telegram with adoption of the content for block 1 from the non-volatile memory**

| Content of block 1 | Parameter in the non-volatile memory |
|---|---|
| C-field | `MBUS_B1_Control` 9.1.23 |
| Man ID 1 | `MBUS_B1_ManID1` 9.1.24 |
| Man ID 2 | `MBUS_B1_ManID2` 9.1.25 |
| ID 1 | `MBUS_B1_ID1` 9.1.26 |
| ID 2 | `MBUS_B1_ID2` 9.1.27 |
| ID 3 | `MBUS_B1_ID3` 9.1.28 |
| ID 4 | `MBUS_B1_ID4` 9.1.29 |
| Version | `MBUS_B1_Version` 9.1.30 |
| Device type | `MBUS_B1_DevType` 9.1.31 |

**Table 2: Settings radio telegram block 1**

# 4 Technical Parameters

## 4.1 Input Voltage

The input voltage of the module ranges from 2.2 to 3.6 V.

## 4.2 Power Consumption AMB8425-M

See data sheet [3].

## 4.3 Dimensions and Weight

See data sheet [3].

## 4.4 Pinout

See data sheet [3].

# 5 Serial User Interface

## 5.1 UART

### 5.1.1 Supported Data Rates

The data rate can be set by directly configuring the respective parameters in the module's non-volatile memory (see `UART_BR0`, `UART_BR1`, and `UART_MCTL` on page 19 ff) or using the command `CMD_SETUARTSPEED_REQ` (6.1.11).

As the UART speed is derived from the speed of the utilised clock quartz, there may be variations of up to 0.5%.

When using the PC program "ACC", the following data rates can be selected directly via drop-down menu:

110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 56000, 57600, and 115200 baud.

With this selection, the registers stated above are automatically set to the optimum value.

Moreover, the "ACC" program also provides a dialogue for calculating any needed baud rates.

The default baud rate of the module is 9600.

The output of characters on the serial interface takes place with secondary priority. For this reason, short interruptions may occur *between* the output of individual characters (e.g., in the event of an interrupt).

### 5.1.2 Supported Data Formats

The following data formats are supported:

- 8 bits
- No, even, or odd parity
- 1 or 2 stop bits

In ACC, the following data formats can be selected directly via drop-down menu:

8n1, 8o1, 8e1, 8n2, 8o2, 8e2.

The data format, too, can be set by directly configuring the respective microprocessor registers.

The default data format is 8 data bits, no parity, 1 stop bit ("8n1").

## 5.2 SPI Interface

Instead of the UART interface, the module also offers an SPI interface (separate firmware; in preparation).

# 6 The Command Interface

Below, the commands already mentioned in 2.2 are described in detail. All module settings specified in the document EN 13757-4:2005 can be adjusted via this interface.

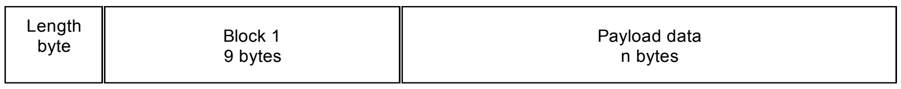
## 6.1 Data Transfer in the Command Mode

### 6.1.1 `CMD_DATA_REQ`

This command serves the simple data transfer. Transmission takes place on the configured channel.

Format:

0xFF ***0x00***



< CS >

Return:

0xFF ***0x80*** 0x01 < status > < CS >

Status:

0x00: data transmitted.

### 6.1.2 `CMD_DATARETRY_REQ`

This command relaunches the transmission of the data previously submitted to the module. Thus, the data do not need to be re-sent over the serial interface.

The buffered data are lost as soon as new data are sent via UART or data are received via wireless transmission.

Format:

0xFF ***0x02*** 0x00 0xFD

Return:

0xFF ***0x82*** 0x01 < status > < CS >

Status:

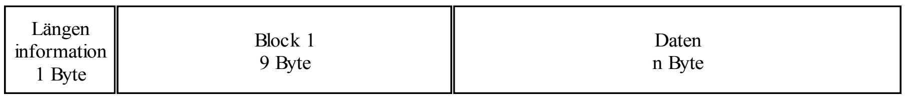
0x00: data sent

### 6.1.3 `CMD_DATA_IND`

This telegram indicates the reception of data bytes and represents the counterpart to the command `CMD_DATA_REQ.`

Format:

0xFF ***0x03***



< PS >

### 6.1.4 `CMD_RESET_REQ`

This command triggers a software reset of the module. The reset is performed after the acknowledgement is issued.

Format:

0xFF ***0x05*** 0x00 0xFA

Return:

0xFF ***0x85*** 0x01 < status > < CS >

Status:

0x00: success

### 6.1.5 `CMD_SET_CHANNEL_REQ`

This command is used to toggle the wireless channel. An overview of the channels is presented in section 0.

Format:

0xFF ***0x06*** 0x01 < 1-byte channel > < CS >

Example (selection of channel 11):

0xFF 0x06 0x01 0x0B 0xF3

Return:

0xFF ***0x86*** 0x01 < new channel > < CS >

Return for above example:

0xFF 0x86 0x01 0x0B 0x73

### 6.1.6 `CMD_SET_REQ`

This command enables the direct manipulation of the parameters in the non-volatile memory of the module. The respective parameters are accessed via the memory position described in Table 6.

Individual or multiple consecutive parameters in the memory can be modified concurrently.

**Caution: The validity of the specified parameters is not verified. Incorrect values can result in device malfunction!**

**Caution: To save the parameters in the flash memory of the module, the particular memory segment must first be flushed entirely and then restored from the RAM. If a reset occurs during this procedure (e.g., due to supply voltage fluctuations), the entire memory area may be destroyed. In this case, the module may no longer be operable, which means that the firmware must be re-installed via "ACC". Recommendation: First, verify the configuration of the module with CMD_GET_REQ; write only if necessary.**

Format:

0xFF ***0x09*** < number of bytes + 2 > < memory position > < number of bytes > < parameter > < CS >

Return:

0xFF ***0x89*** 0x01 < status > < CS >

Status:

0x00: success

0x01: verification failed

0x02: invalid memory position or invalid number of bytes to be written (write access to unauthorised area > 127 / 0x7F)

Example 1: Change adding of block 1 (`MBUS_Bl_ADD_Disable`; according to Table 6, memory position 0x2Dh):

0xFF ***0x09*** 0x03 0x2D 0x01 < parameter value > < CS >

Example 2: Setting the 3 registers for the baud rate configuration (UART_MCTL, UART_BR0, and UART_BR1). According to Table 6, UART_BR0 has the memory position 0x02h:

0xFF ***0x09*** 0x05 0x02 0x03 < UART_BR0 > < UART_BR1 > < UART_MCTL > < CS >

### 6.1.7 `CMD_GET_REQ`

This command can be used to query individual or multiple non-volatile parameters (see 9.1). The requested number of bytes from the specified memory position is returned.

Individual or multiple consecutive parameters in the memory can be queried concurrently.

Format:

0xFF ***0x0A*** 0x02 < memory position > < number of bytes > < CS >

Example (query of all parameters):

0xFF ***0x0A*** 0x02 0x00 0x80 0x77

Return:

0xFF ***0x8A*** < number of bytes + 2 [0x82]> < memory position[0x00] > < number of bytes[0x80] > < parameter > < CS >

The read access to the memory area behind the parameters documented in Table 6 is blocked. The memory position and the number of bytes are limited accordingly. Thus, the last memory position that can be read out is 127 (0x7F).

### 6.1.8 `CMD_SERIALNO_REQ`

This command can be used to query the individual serial number of the module.

Format:

0xFF ***0x0B*** 0x00 0xF4

Return:

0xFF ***0x8B*** 0x04 < 4-byte serial number > < CS >

The most significant byte is returned first (MSB first); this byte identifies the product ("product ID").

### 6.1.9 `CMD_RSSI_REQ`

This command delivers the current RX level determined by the transceiver IC in the form of a two's complement.

Format:

0xFF ***0x0D*** 0x00 0xF2

Return:

0xFF ***0x8D*** 0x01 < RX level > < CS >

The value obtained in this way delivers the RX level $RSSI_{dBm}$ in dBm as follows:

1. Conversion of the hexadecimal value to a decimal $RSSI_{dec}$
2. If $RSSI_{dec} \geq 128$: $RSSI_{dBm} = (RSSI_{dec} - 256) / 2 - RSSI_{Offset}$
3. Otherwise ($RSSI_{dec} < 128$): $RSSI_{dBm} = RSSI_{dec} / 2 - RSSI_{Offset}$

$RSSI_{Offset}$ is a data-rate-dependent correction factor according to Table 3 (AMB8425).

The relation between the calculated value and the physical RX level in dBm is not linear across the entire operating range and is displayed in Figure 7.

| Data rate | RSSI offset |
|---|---|
| 1.2 kbps | 74 |
| 38.4 kbps | 74 |
| 250 kbps | 78 |

**Table 3: Data-rate-dependent RSSI offset for AMB8425 (from [2])**

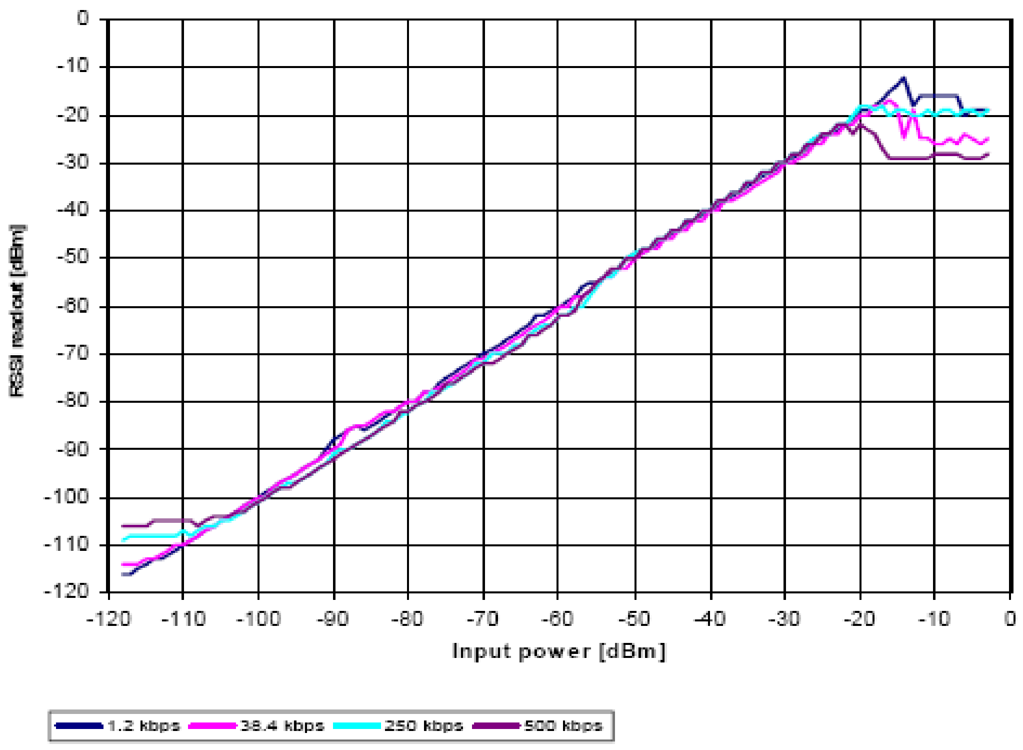


**Figure 7**

**Relation between the RX level and the RSSI value read out for AMB8425 (from [2])**

### 6.1.10 `CMD_ERRORFLAGS_REQ`

This command returns internal error states.

Format:

0xFF ***0x0E*** 0x00 0xF1

Return:

0xFF ***0x8E*** 0x02 < error flags MSB > < error flags LSB >< CS >

An error flag return value of "0" indicates that no error has occurred. The value is set back after the query and in the event of a reset.

The meaning of the error flags is not described in detail in this context.

### 6.1.11 `CMD_SETUARTSPEED_REQ`

This command changes the parameters of the serial user interface (`UART_CTL0`, `UART_CTL1`, `UART_BR0`, `UART_BR1`, `UART_MCTL`) to preset values for fixed transfer rates. The index of the transfer rate is indicated in Table 4.

| Transfer rate [baud] | Index [hex] |
|---|---|
| 1200 | 0x00 |
| 2400 | 0x01 |

| Transfer rate [baud] | Index [hex] |
|---|---|
| 4800 | 0x02 |
| 9600 | 0x03 |
| 19200 | 0x04 |
| 38400 | 0x05 |
| 56000 | 0x06 |
| 115200 | 0x07 |

**Table 4: Preconfigured transfer rate of the serial interface**

Format:

0xFF ***0x10*** 0x00 < index > < CS >

Return:

0xFF ***0x90*** 0x01 < status > < CS >

Status:

0x00: success

### 6.1.12 `CMD_FACTORYRESET_REQ`

This command resets all parameters to the factory settings:

Format:

0xFF ***0x11*** 0x00 0xEE

Return:

0xFF ***0x90*** 0x01 < status > < CS >

Status:

0x00: success

# 7 Factoryreset

One option to perform a factory reset ist the command `CMD_FACTORYRESET_REQ`. This commad can be used as long as the UART settings are known and the module can be configured by the serial port.

Another way to mak a factory reset is the Boot Up Factory Reset. To use this mechanism, Pin4.3 and Pin4.4 must be connected during a reset. Upon success Pin2.1 (RX_INDICATE) will toggle 3 times.

After a Factory Reset the speed of the serial interface is set back to 9600 8n1.

# 8 Toggling the Operating Mode via Pin

Moreover, the host can use a pin interface to toggle the preconfigured operating modes. If this interface is activated, the unit toggles between the preconfigured wireless M-bus operating modes. To activate this functionality, the parameter

Mode_Preselect is set to the value 0x0F, and the pins of the interface are connected according to the desired operating mode.

| Pinout | Selected operating mode |
|---|---|

| Pinout | | Selected operating mode |
|---|---|---|
| [1.5] | [1.4] | |
| 0 | 0 | None |
| 0 | 1 | S1 |
| 1 | 0 | S1-m |
| 1 | 1 | S2 |
| 0 | 0 | Reserved |
| 0 | 1 | T1 meter |
| 1 | 0 | T1 other |
| 1 | 1 | T2 meter |
| 0 | 0 | T2 other |
| 0 | 1 | Reserved |
| 1 | 0 | R2 meter |
| 1 | 1 | R2 other |
| 0 | 0 | Reserved |
| 0 | 1 | Reserved |
| 1 | 0 | Reserved |
| 1 | 1 | Not permitted |

**Table 5: Signals when toggling between the modes via pin interface**

**Caution: In the event of periodic toggling, the duration of the state for one operating type should be at least 2.5 ms, as otherwise the detection of a preamble cannot be guaranteed!**

# 9 Configuration Parameters

## 9.1 Non-volatile Configuration Parameters

The non-volatile parameters listed in the following table can be modified by means of specific commands in the configuration mode (`CMD_SET_REQ`, see 6.1.6) of the module or by using the Windows software "AC". These parameters are stored permanently in the module's flash memory.

**Caution: The validity of the specified parameters is not verified. Incorrect values can result in device malfunction!**

| Parameter | Description | Adr. [hex] | Permissible values [hex] | Default [hex] |
|---|---|---|---|---|
| `UART_CTL0` | Control register for UART data format | 0x00 | See Table 7 | |

| Parameter | Description | Adr. [hex] | Permissible values [hex] | Default [hex] |
|---|---|---|---|---|
| UART_CTL1 | Control register for UART | 0x01 | 0x80 | |
| UART_BR0 | Prescaler for setting the baud rate (LSB) | 0x02 | Compute with ACC | |
| UART_BR1 | Prescaler for setting the baud rate (MSB) | 0x03 | Compute with ACC | |
| UART_MCTL | Modulation control register UART | 0x04 | Compute with ACC | |
| UART_CMD_Out_Enable | Output from received frames in CMD format | 0x05 | 0x00, 0x01 | 0x00 |
| UART_DIDelay | Configurable output delay | 0x06 | 0x0000 – 0xFFFF | 0x0000 |
| UART_Reserved3 | Reserve | 0x08 | | 0xFF |
| UART_Reserved4 | Reserve | 0x09 | | 0xFF |
| APP_MAXPacketLength | Maximum number of bytes to send/receive | 0x0A | 0x0A – 0xFF | 0x80 |
| APP_AES_Enable (in Preparation) | Enable encryption | 0x0B | See Table 8 | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB | 0x0C | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 1 | 0x0D | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 2 | 0x0E | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 3 | 0x0F | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 4 | 0x10 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 5 | 0x11 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 6 | 0x12 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 7 | 0x13 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 8 | 0x14 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 9 | 0x15 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 10 | 0x16 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 11 | 0x17 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 12 | 0x18 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 13 | 0x19 | 0x00 – 0xFF | 0x00 |
| APP_AESX (in Preparation) | Key AES128 MSB – 14 | 0x1A | 0x00 – 0xFF | 0x00 |
| APP_AESX (in | Key AES128 MSB – 15 | 0x1B | 0x00 – 0xFF | 0x00 |

| Parameter | Description | Adr. [hex] | Permissible values [hex] | Default [hex] |
|---|---|---|---|---|
| Preparation) | or LSB | | | |
| APP_WOR_PeriodH | Period length in ms - high byte | 0x1C | 0x00 – 0xFF | 0x00 |
| APP_WOR_PeriodL | Period length in ms - low byte | 0x1D | 0x00 – 0xFF | 0x32 |
| APP_WOR_MultiplierH | Multiplier of the period length - high byte | 0x1E | 0x00 – 0xFF | 0x00 |
| APP_WOR_MultiplierL | Multiplier of the period length - low byte | 0x1F | 0x00 – 0xFF | 0x02 |
| APP_RX_Time | WOR time in ms during which the module remains in the RX mode | 0x20 | 0x00 – 0xFF | 0x14 |
| APP_RX_After_RX (in Preparation) | Time that the module remains in the RX mode after the reception | 0x21 | 0x00 – 0xFF | 0x00 |
| APP_Reserved0 | Reserve | 0x22 | | 0xFF |
| APP_Reserved1 | Reserve | 0x23 | | 0xFF |
| APP_Reserved2 | Reserve | 0x24 | | 0xFF |
| APP_Reserved3 | Reserve | 0x25 | | 0xFF |
| APP_Reserved4 | Reserve | 0x26 | | 0xFF |
| MBUS_Coding | Coding method | 0x27 | See Table 9 | 0x01 |
| MBUS_HeaderLengthH | Header length in bytes (high byte) | 0x28 | 0x00 – 0xFF | 0x00 |
| MBUS_HeaderLengthL | Header length in bytes (low byte) | 0x29 | 0x00 – 0xFF | 0x04 |
| MBUS_Synch | Synchronisation character | 0x2A | See Table 10 | 0x00 |
| MBUS_RXTimeout | Time in ms that the module remains in the RX mode after transmitting data | 0x2B | 0x00 – 0xFF | 0x00 |
| MBUS_Reserved1 | Reserve | 0x2C | | 0xFF |
| MBUS_Reserved2 | Reserve | 0x2D | | 0xFF |
| MBUS_Reserved3 | Reserve | 0x2E | | 0xFF |
| MBUS_Reserved4 | Reserve | 0x2F | | 0xFF |
| MBUS_Bl_ADD_Disable | Disable adding of block 1 | 0x30 | See Table 12 | 0x00 |
| MBUS_Bl_Control | C-field of block 1 | 0x31 | 0x00 – 0xFF | 0x44 |
| MBUS_Bl_ManID1 | M-field of block 1 | 0x32 | 0x00 – 0xFF | 0x00 |
| MBUS_Bl_ManID2 | M-field of block 1 | 0x33 | 0x00 – 0xFF | 0x00 |
| MBUS_Bl_IDNr1 | A-field of block 1 | 0x34 | 0x00 – 0xFF | 0x00 |
| MBUS_Bl_IDNr2 | A-field of block 1 | 0x35 | 0x00 – 0xFF | 0x00 |
| MBUS_Bl_IDNr3 | A-field of block 1 | 0x36 | 0x00 – 0xFF | 0x00 |
| MBUS_Bl_IDNr4 | A-field of block 1 | 0x37 | 0x00 – 0xFF | 0x00 |
| MBUS_Bl_Version | A-field of block 1 | 0x38 | 0x00 – 0xFF | 0x00 |
| MBUS_Bl_DevType | A-field of block 1 | 0x39 | 0x00 – 0xFF | 0x00 |

| Parameter | Description | Adr. [hex] | Permissible values [hex] | Default [hex] |
|---|---|---|---|---|
| `MBUS_Reserved5` | Reserve | 0x3A | | 0xFF |
| `MBUS_Reserved6` | Reserve | 0x3B | | 0xFF |
| `RF_Channel` | Radio channel | 0x3C | See Table 13 | 0x0B |
| `RF_Power` | Transmission power | 0x3D | See Table 14 | 0x05 |
| `RF_DataRate` | Chip rate | 0x3E | See Table 15 | 0x01 |
| `RF_AutoSleep` | Sleep mode | 0x3F | See Table 16 | 0x00 |
| `RF_Reserved0` | Reserve | 0x40 | | 0xFF |
| `RF_Reserved1` | Reserve | 0x41 | | 0xFF |
| `RF_Reserved2` | Reserve | 0x42 | | 0xFF |
| `RF_Reserved3` | Reserve | 0x43 | | 0xFF |
| `RF_Reserved4` | Reserve | 0x44 | | 0xFF |
| `RSSI_Enable` | Enable RSSI value output | 0x45 | See Table 17 | 0x00 |
| `Mode_Preselect` | Default settings for M-bus operating modes | 0x46 | See Table 18 | 0x03 |
| `Net_Mode` | Modem, concentrator, or repeater mode | 0x47 | In preparation | 0x00 |
| `Config_CRC_Disable` | Take checksum into consideration in the configuration mode or not | 0x48 | See Table 19 | 0x00 |
| `Reserved0` | Reserve | 0x49 | | 0xFF |
| `Reserved1` | Reserve | 0x4A | | 0xFF |
| `Reserved2` | Reserve | 0x4B | | 0xFF |
| `Reserved3` | Reserve | 0x4C | | 0xFF |
| `Reserved4` | Reserve | 0x4D | | 0xFF |
| `Reserved5` | Reserve | 0x4E | | 0xFF |
| `Reserved6` | Reserve | 0x4F | | 0xFF |
| `CFG-Flags` | Configuration flags | 0x50 | 0x0000 – 0xFFFF | 0x0000 |

**Table 6: Non-volatile parameters**

### 9.1.1 `UART_CTL0`

With the help of this register, the format of the serial interface can be adjusted.

| Bit no. | Description |
|---|---|
| 0 (0x01) | Toggle between UART (0x00h) and USART (0x01h). |
| 1 to 2 (0x02 – 0x04) | Reserved, must always be set to 0x00, |
| 3 (0x08) | This bit selects the number of stop bits. If this bit is set, 2 stop bits will be used, if not, 1 will be used. |
| 4 (0x10) | If this bit is set, the character length will be 7 bits, if not, it will be 8 bits. |
| 5 (0x20) | If the MSB of a byte is the first thing to be sent, this bit must be set. Otherwise, it remains 0x00. |

| | |
|---|---|
| 6 (0x40) | Even parity if the bit is set, otherwise odd parity |
| 7 (0x80) | This bit enables the use of parity (if set). |

**Table 7: Parameter UART_CTL0**

### 9.1.2 `UART_CTL1`

This register selects the source for generating the UART clock speed. Currently, the only permissible value is 0x80h.

### 9.1.3 `UART_BR0`

The ACC program should be used to compute this parameter.

### 9.1.4 `UART_BR1`

The ACC program should be used to compute this parameter.

### 9.1.5 `UART_MCTL`

The ACC program should be used to compute this parameter.

### 9.1.6 `UART_CMD_Out_Enable`

If this parameter contains the value 0x01h, received frames will send in cmd format over the UART. If the parameter contains 0x00h, the output format is transparent.

### 9.1.7 `UART_DIDelay`

This parameter determines the duration in milliseconds between the incoming radio signal data on the pin / DATA_INDICATION and the output of data via the UART. This delay can be used to prepare eg a "sleeping" host system to receive the data. Activated from SW version 1.6 with CFG flags

### 9.1.8 `APP_MAXPacketLength`

The maximum packet length can be set by means of this parameter. This limits the number of bytes issued via the serial interface in case the device after the module only has limited buffer space.

### 9.1.9 `APP_AES_Enable (in Preparation)`

If this parameter contains the value 0x00h, the AES128 encryption is disabled. If the value is 0x01h, the AES128 encryption key entered in the following 16 bytes is used.

| Value | Description |
|---|---|
| 0x00 | Encryption disabled |
| 0x01 | Encryption enabled |

**Table 8: Parameter APP_AES_ENABLE**

### 9.1.10 `APP_AESX (in Preparation)`

The 128-bit key for AES128 encryption is entered in these parameters. APP_AES0 contains the MS byte and APP_AESF the LS byte.

### 9.1.11 `APP_WOR_PeriodH`

This parameter contains the high byte for the length of the wake period in WOR operation. Together with the parameter

### 9.1.12 `APP_WOR_PeriodL`

the result is a parameter that determines the length of the wake period in ms. The parameters APP_WOR_MultiplierH and APP_WOR_MultiplierL can be used to enable WOR times of more than 65535 ms.

### 9.1.13 `APP_WOR_MultiplierH`

and

### 9.1.14 `APP_WOR_MultiplierL`

are multipliers for the length of the period of the module's WOR operation. Thus, the total interval between the wake times is:

*Wake time*[ms]
= ((APP_WOR_PeriodH x 255) + APP_WOR_PeriodL) x ((APP_WOR_MultiplierH x 255) + APP_WOR_MultiplierL)

i.e., after every wake-up timeout, the module wakes up and "listens" to see whether a preamble is being received. If a preamble is being received, the module will look for a synchronisation character.

N.B.: Do not set this parameter to 0.

### 9.1.15 `APP_RX_Time`

By means of this parameter, the user can set the time period during which the module is to be in RX mode in WOR operation. Up to 255 ms can be set. The minimum time period should be 3 ms in order to make sure that the radio chip is ready. If a preamble is received during this time period, the module will remain in RX mode until a data set is received or the preamble aborts.

### 9.1.16 `APP_RX_After_RX (in Preparation)`

If the module is to remain in RX mode for a certain time period after receiving a data set in WOR mode, this parameter must contain this period in ms.

### 9.1.17 `MBUS_Coding`

The coding can be set by means of this parameter.

| Value [hex] | Description |
|---|---|
| 0x00 | 3-of-6 coding |
| 0x01 | Manchester coding |

**Table 9: Parameter MBUS_Coding**

### 9.1.18 `MBUS_HeaderLengthH`

The high byte of the entire number of bytes before the synchronisation character is set in this parameter. Together with the parameter

### 9.1.19 `MBUS_HeaderLengthL`

this results in a 16-bit number that contains the number of header bytes. The number of (01) is computed as follows:

*Number*(01)*sequences* = ((*MBUS_HeaderLengthH* x 255) + *MBUS_HeaderLengthL*) x 4

Example: If the value contained in the high byte is 0x00h and the value contained in the low byte is 0x46, the total number of bytes is 0x0046h (70). Each byte consists of 4 (01) sequences. According to the formula, this means that there are 280 (01) sequences. Thus, it is sufficient to set the high byte to 0x00h and the low byte to 0x46h in order to send a valid header length in the operating mode S1 (long message header).

The duration of the header can be determined by means of the following formula:

$$\textit{Header duration}[s] = \frac{8 \times ((\textit{HeaderLengthH} \times 255) + \textit{HeaderLengthL})}{\textit{chip rate}[Hz]}$$

### 9.1.20 `MBUS_Synch`

The value in this parameter is the index for the selection of the synchronisation character.

| Value | Synchronisation character |
|---|---|
| 0x00 | 000111011010010110 |
| 0x01 | 0000111101 |

**Table 10: Parameter MBUS_Synch**

The selection of the synchronisation character is effective for the transmission and for the reception.

### 9.1.21 `MBUS_RXTimeout`

By means of this parameter, it is possible to configure the time in ms for which the module is to remain in the RX mode after transmitting data before it enters the sleep mode. This parameter is effective in all operating modes. If the parameter contains the value 0x00h, no timeout is activated.

| Value | Description |
|---|---|
| 0x00 | RX timeout disabled |
| > 0x00 | RX timeout in ms |

**Table 11: Parameter MBUS_RXTimeout**

### 9.1.22 `MBUS_Bl_ADD_Disable`

The adding of the first block to be transferred can be disabled by means of this parameter. Thus, the host must transfer the content for the C-field, M-field, and A-field over the serial interface (Figure 8).

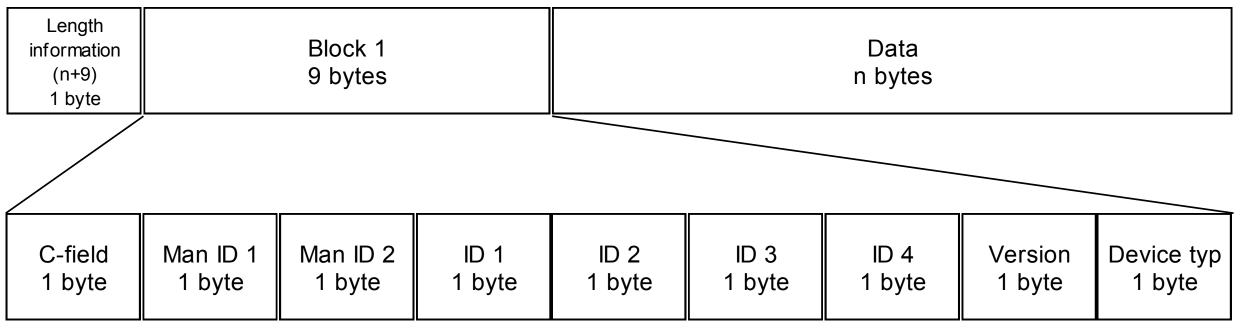


**Figure 8: Adding of block 1 disabled**

In this case, the values stored in the non-volatile memory will not be compared with the received values of the M-field and the A-field in block 1. The data are issued with optional field strength value (RSSI) (see  Figure 9).

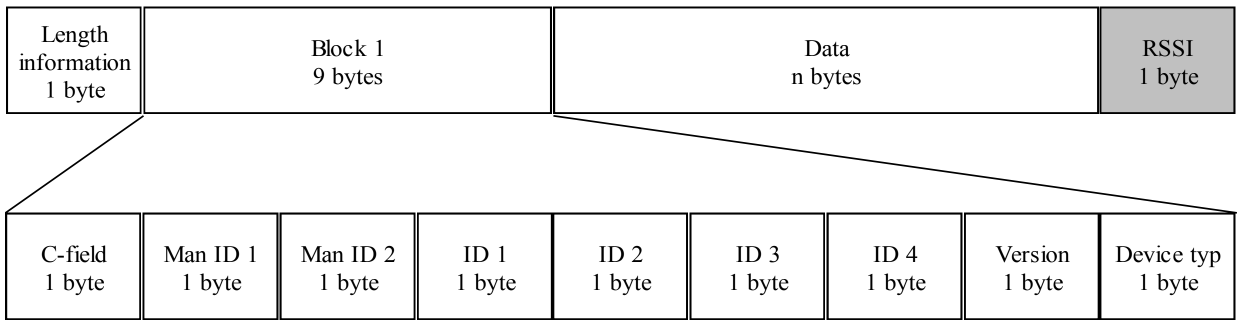


**Figure 9: Output of received data if adding of block 1 is disabled**

| Value | Description |
|---|---|
| 0x00 | The address is evaluated and block 1 is added in the module. |
| 0x01 | Block 1 is not evaluated and added in the module. |

**Table 12: Parameter MBUS_BI_ADD_Disable**

#### 9.1.23 `MBUS_Bl_Control`
This byte contains the value of the C-field in the first block of the data to be sent.

#### 9.1.24 `MBUS_Bl_ManID1`
This byte contains the value of the first byte of the M-field in the first block of the data to be sent.

#### 9.1.25 `MBUS_Bl_ManID2`
This byte contains the value of the second byte of the M-field in the first block of the data to be sent.

#### 9.1.26 `MBUS_Bl_IDNr1`
This byte contains the value of the first byte of the A-field in the first block of the data to be sent.

#### 9.1.27 `MBUS_Bl_IDNr2`
This byte contains the value of the second byte of the A-field in the first block of the data to be sent.

#### 9.1.28 `MBUS_Bl_IDNr3`
This byte contains the value of the third byte of the A-field in the first block of the data to be sent.

#### 9.1.29 `MBUS_Bl_IDNr4`
This byte contains the value of the fourth byte of the A-field in the first block of the data to be sent.

#### 9.1.30 `MBUS_Bl_Version`
This byte contains the value of the fifth byte of the A-field in the first block of the data to be sent. (Version)

#### 9.1.31 `MBUS_Bl_DevType`
This byte contains the value of the sixth byte of the A-field in the first block of the data to be sent. (Device type)

#### 9.1.32 `RF_Channel`
The desired TX/RX channel can be selected by means of this parameter.

| Channel | Value | Frequency [MHz] | Operating mode |
|---|---|---|---|
| 1 | 0x01 | 868.03 | R2 |
| 2 | 0x02 | 868.09 | R2 |
| 3 | 0x03 | 868.15 | R2 |
| 4 | 0x04 | 868.21 | R2 |
| 5 | 0x05 | 868.27 | R2 |
| 6 | 0x06 | 868.33 | R2 |
| 7 | 0x07 | 868.39 | R2 |
| 8 | 0x08 | 868.45 | R2 |
| 9 | 0x09 | 868.51 | R2 |
| 10 | 0x0A | 868.57 | R2 |

| | | | |
|---|---|---|---|
| 11 | 0x0B | 868.30 | S1, S1-m, S2, T2 |
| 12 | 0x0C | 868.95 | T1, T2 |

**Table 13: Parameter RF_Channel**

All changes to these parameters are adopted *without* reset.

### 9.1.33 `RF_Power`

Enables the adjustment of the transmission power.

| Power [dBm] | Value |
|---|---|
| -5 | 0x01 |
| 0 | 0x02 |
| +5 | 0x03 |
| +7 | 0x04 |
| +10 | 0x05 |

**Table 14: Parameter RF_Power**

### 9.1.34 `RF_DataRate`

The various chip rates can be selected by means of this parameter.

| Chip rate [kcps] | Value | Operating mode |
|---|---|---|
| 4.8 | 0x00 | R2 |
| 32.768 | 0x01 | S1, S1-m, S2, T2 |
| 100.0 | 0x02 | T1, T2 |

**Table 15: Parameter RF_DataRate**

### 9.1.35 `RF_AutoSleep`

The parameter RF_AutoSleep supports 3 different settings. First, AutoSleep can be disabled. In this case, the module will be in the RX mode permanently. If the parameter is set to WOR, the radio chip will wake up at the configured intervals and enter the RX mode. If the parameter is set to sleep, the entire module will go to sleep. In this case, it can only be waked up via the user interface (data or command mode).

| Value [hex] | Description |
|---|---|
| 0x00 | Between the transmissions, the module is permanently in RX mode |
| 0x01 | The module wakes up according to the settings of the parameters `APP_WOR_PeriodH`, `APP_WOR_PeriodL`, `APP_WOR_MultiplierH`, and `APP_WOR_MultiplierL` and listens for the time set in the parameter `APP_RX_Time` in order to see whether it receives a preamble. |

| | |
|---|---|
| 0x02 | Between the transmissions, the module is always in the sleep mode (and cannot receive anything). |

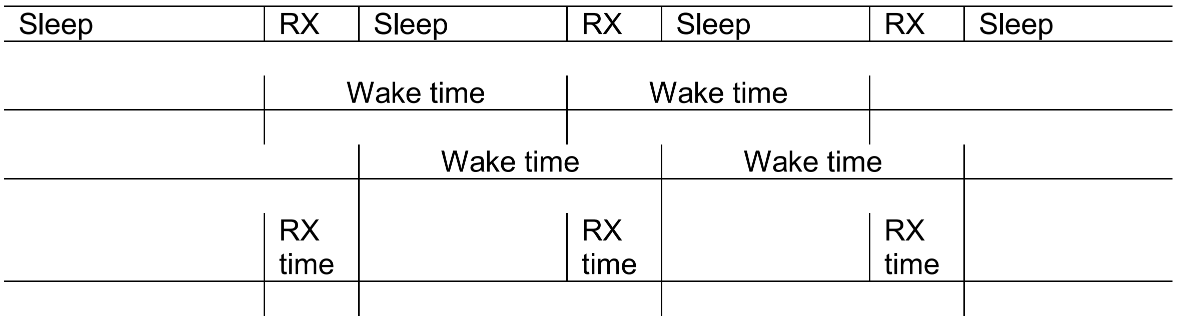
**Table 16: Parameter RF_AutoSleep**

- RX:

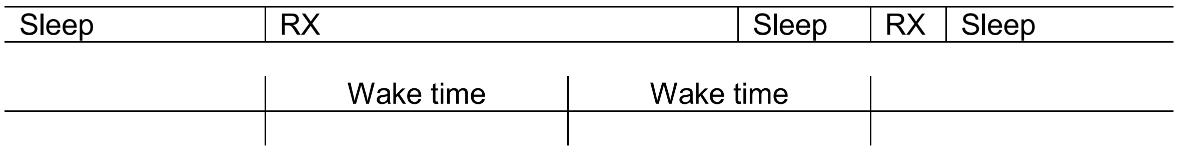
  The module is permanently in RX mode. Data can be received at all times.

- WOR :

  If the module is set to WOR operation (parameter RF_AutoSleep = 0x01), a timer that counts the time defined by means of the parameters `APP_WOR_PeriodH`, `APP_WOR_PeriodL`, `APP_WOR_MultiplierH`, and `APP_WOR_MultiplierL` is started in order to wake up the module.



The above graph demonstrates the periodic switch between RX and sleep if the module does not receive any preamble during the RX period. If the module receives a preamble and a data set, the wake time will continue periodically.



- Sleep:

  In this mode, the module has the lowest power consumption, as the radio chip and µC are in the low-power state. In this mode, the power consumption is <1 µA. By way of the UART, the module can be waked up at all times for transmitting data or for configuration purposes.

### 9.1.36 `RSSI_Enable`

If this parameter contains the value 1, the RSSI value will not be appended to the payload data as shown in Figure 3/Figure 4.

| RSSI | Value |
|---|---|
| Disabled | 0x00 |
| Enabled | 0x01 |

**Table 17: Parameter RSSI_Enable**

### 9.1.37 `Mode_Preselect`

Default settings of the operating modes specified in [1] can be selected by means of this parameter.

| Operating mode | Value | Direction(s) | Header length [(01) toggle] | Synchro-nisation character [Index] | Channel [Index] | Chip rate [Index] | Coding | Auto-sleep [Index] |
|---|---|---|---|---|---|---|---|---|
| UI | 0x00 | 2 | UI | UI | UI | UI | UI | UI |
| S1 | 0x01 | TX | 280 | 0 | 11 | 1 | 1 | 2 |
| S1-m | 0x02 | TX | 16 | 0 | 11 | 1 | 1 | 2 |
| S2 | 0x03 | 2 | UI | 0 | 11 | 1 | 1 | UI |
| Reserved | 0x04 | | | | | | | |
| T1 meter | 0x05 | TX | 20 | 1 | 12 | 2 | 0 | 2 |
| T1 other | 0x06 | TX | 16 | 0 | 11 | 1 | 1 | 2 |
| T2 meter | 0x07 | TX | 20 | 1 | 12 | 2 | 0 | UI |
| | | RX | | 0 | 11 | 1 | 1 | UI |
| T2 other | 0x08 | TX | 16 | 0 | 11 | 1 | 1 | UI |
| | | RX | | 1 | 12 | 2 | 0 | UI |
| Reserved | 0x09 | | | | | | | |
| R2 meter | 0x0A | TX | 40 | 0 | UI | 0 | 1 | UI |
| | | RX | | 0 | 6 | 0 | 1 | UI |

| Operating mode | Value | Direction(s) | Header length [(01) toggle] | Synchro-nisation character [Index] | Channel [Index] | Chip rate [Index] | Coding | Auto-sleep [Index] |
|---|---|---|---|---|---|---|---|---|
| R2 other | 0x0B | TX | 40 | 0 | 6 | 0 | 1 | UI |
| | | RX | | 0 | UI | 0 | 1 | UI |
| Reserved | 0x0C | | | | | | | |
| Reserved | 0x0D | | | | | | | |
| Reserved | 0x0E | | | | | | | |
| Toggle mode | 0x0F | PS | PS | PS | PS | PS | PS | UI |

**Table 18: Parameter Mode_Preselect**

The transmission power is always determined by the value of the parameter `RF_Power`. Therefore, it must also be configured for the preconfigured modes.

- If the parameter contains the value 0x00h (no default settings), all parameters will be adopted from the non-volatile memory.
- If the module is connected to a meter that is supposed to transmit the meter level to a stationary concentrator at certain intervals and that does not expect any feedback, the operating mode S1 can be selected, thereby setting the parameter Mode_Preselect to the value 0x01h.
- If the concentrator is not battery-operated and therefore does not need an extended preamble, you can also select the operating mode S1-m. For this, the parameter Mode_Preselect can be set to the value 0x02h.
- In the previous two cases, the concentrator that collects the meter data should run in the operating mode S2 (0x03h), as this mode also comprises the reception of data. The preamble length must be set by means of the parameters MBUS_HeaderLengthH and MBUS_HeaderLengthL. Just as the behaviour of the module in the stand-by mode (AutoSleep).
- If a meter is to send its data in the form of short data blocks—e.g., in order to enable drive-by reading of these data—the module can be run in the mode T1 meter (Mode_Preselect = 0x05h).
- The setting 0x06h, which selects the operating mode T1 other, sets the same parameter as the operating mode S1-m.
- If the meter is also to be able to receive data (drive-by reading and wireless configuration of the meter), the module should run in the operating mode T2 meter (0x07h).
- In the receiving device, which can also configure the meter, the module would have to run in the operating mode T2 other (0x08h) in order to be able to communicate with the meter in the operating mode T1 meter or T2 meter.
- If data are to be transmitted frequently and this is to take place over a long radio distance, the parameter Mode_Preselect can be set to the value 0x0Ah (meter) or 0x0Bh (other), and the module operates in the wireless M-bus [1] operating mode R2. Here too, the stand-by behaviour corresponds to the configured parameter RF_AutoSleep.

- For quick toggling between the operating modes, the parameter should be set to the value 0x0Fh. In this way, the toggling between the preconfigured wireless M-bus [1] operating modes via pins (see 8) is activated.

### 9.1.38 `Net_Mode`

Reserved, must be set to 0x00h.

### 9.1.39 `Config_CRC_Disable`

By means of this parameter, the checksum check can be disabled when transmitting commands in the configuration mode. If the value of the parameter is 0x00h, the checksum will be taken into consideration. Otherwise, the checksum will not be checked and will be processed without being checked for correct transfer.

| Value [hex] | Description |
|---|---|
| 0x00 | CRC required |
| 0x01 | CRC not required |

**Table 19: Parameter Config_CRC_Disable**

### 9.1.40 `CFG-Flags`

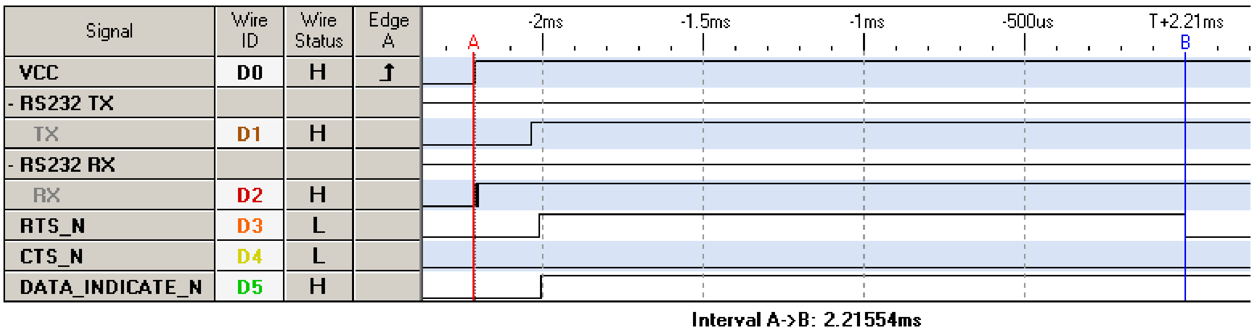
This 2 byte large parameter provides the ability to carry out different configurations. It serves e.g. to activate various pin functions. All pin functions are disabled in the factory setting, so that the normal pin configuration is compatible with previous modules. A description of the flags, see Table 20. During the configuration of this parameter it is important to ensure that the LSB is transmitted always first. All settings will take effect only after a hardware reset of the module.

| Bit Nr. | Description |
|---|---|
| 0 (0x0001) | If set, this bit activates the function of the pin /**RTS**. After a reset of the module /RTS function is then available on pin P4.3. This pin will always symbolize whether the internal buffer is currently occupied (assigned = high). |
| 1 (0x0002) | If set, this bit activates the function of the pin **/CTS**. After the reset of the module, the /CTS function is then available on pin P4.4. This pin will be queried before each byte to be output via the UART (low = data will be send). |
| 2 (0x0004) | If set, this bit activates the function of the pin **/DataIndication**. After the reset of the module /DataIndication function is available on pin P4.5. This pin will indicate whether data is ready for output via UART. It can be used, for example, to awaken a host processor. A delay between the display at the pin and output on the UART can be done using the parameter UART_DIDelay. |
| 3 bis 15 (0xFFF8) | Reserved |

**Table 20 Configurationflags**

Examples:

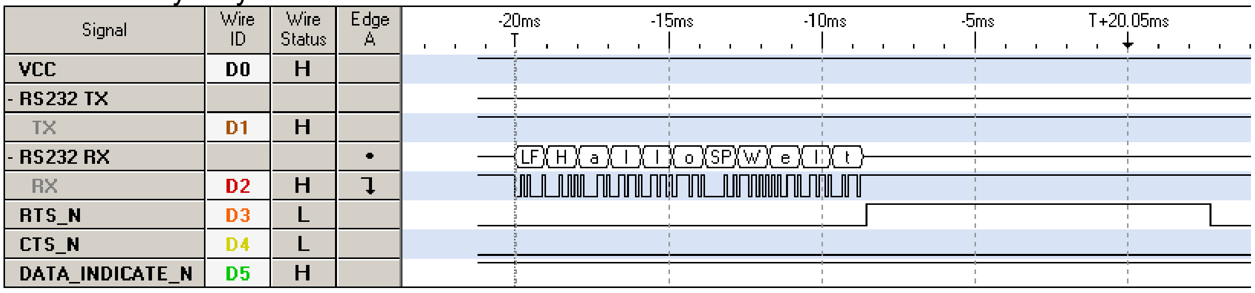
Activated Pins: / RTS + / CTS + /DataIndication



| Signal | Wire ID | Wire Status | Edge A |
|---|---|---|---|
| VCC | D0 | H | ↥ |
| - RS232 TX | | | |
| TX | D1 | H | |
| - RS232 RX | | | |
| RX | D2 | H | |
| RTS_N | D3 | L | |
| CTS_N | D4 | L | |
| DATA_INDICATE_N | D5 | H | |

**Figure 9-10 Turn-on**

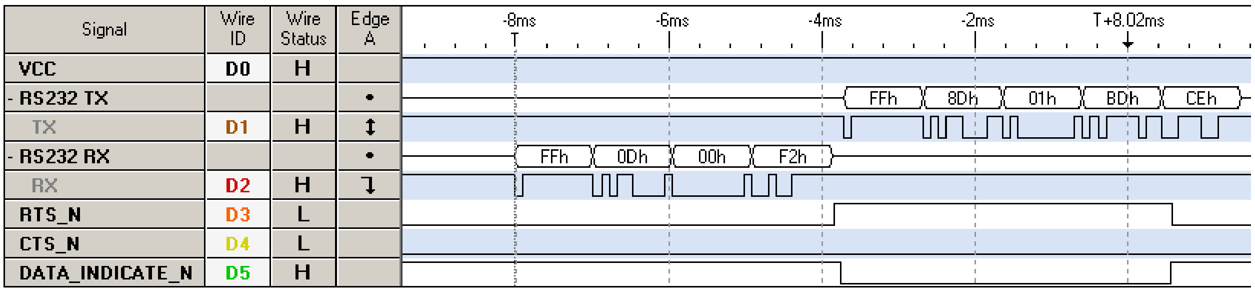
On turn-On the /RTS pin, signalize when the module can receive data via the UART (B signals). This time may vary.



| Signal | Wire ID | Wire Status | Edge A |
|---|---|---|---|
| VCC | D0 | H | |
| - RS232 TX | | | |
| TX | D1 | H | |
| - RS232 RX | | | • |
| RX | D2 | H | ↴ |
| RTS_N | D3 | L | |
| CTS_N | D4 | L | |
| DATA_INDICATE_N | D5 | H | |

**Figure 9-11 Send data (transparent mode))**

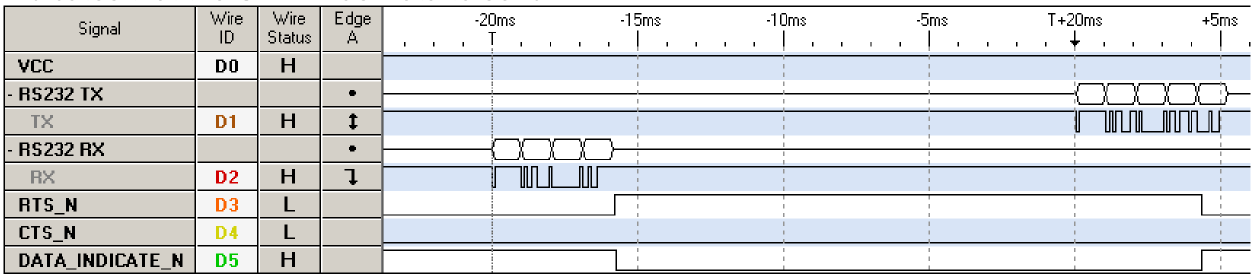
When sending data, the / RTS pin indicates the date from the buffer is occupied for the transmission process (rising edge / RTS) and when the UART is receiving again (falling edge / RTS).



| Signal | Wire ID | Wire Status | Edge A |
|---|---|---|---|
| VCC | D0 | H | |
| - RS232 TX | | | • |
| TX | D1 | H | ↕ |
| - RS232 RX | | | • |
| RX | D2 | H | ↴ |
| RTS_N | D3 | L | |
| CTS_N | D4 | L | |
| DATA_INDICATE_N | D5 | H | |

**Figure 9-12 RSSI Command Request (UART_DIDelay = 0x0000)**

At a command, the / RTS pin indicates the dates when the buffer is occupied. /DataIndication Indicates that the UART has Data to send.



| Signal | Wire ID | Wire Status | Edge A |
|---|---|---|---|
| VCC | D0 | H | |
| - RS232 TX | | | • |
| TX | D1 | H | ↕ |
| - RS232 RX | | | • |
| RX | D2 | H | ↴ |
| RTS_N | D3 | L | |
| CTS_N | D4 | L | |
| DATA_INDICATE_N | D5 | H | |

**Figure 9-13 RSSI Command Request (UART_DIDelay = 0x0010)**

# 10 Using the Module

## 10.1 Minimal Configuration

In the factory state, the modules are immediately ready for operation; the following pins are required in the minimal configuration: VCC, GND, UTXD, and URXD.

If the module is to be connected to a PC, a level converter (TTL to RS232) must be used.

## 10.2 Transfer of Large Amounts of Data

When transmitting larger amounts of data, the limited buffer size in the module must be taken into consideration. The data can only be transmitted packet by packet.

## 10.3 Use of the Low-Power Functionality

In this case, we recommend using the parameter `RF_AutoSleep`. If this parameter is set to sleep (0x02h), the module will be in the sleep mode except during transmission. During this time, it is not possible to receive data. If the reception/listening is controlled from the host, the module can be set to the RX mode by reconfiguring this parameter to RX (0x00h). If a message is received during this time, the module will forward it to the host and can subsequently go back to sleep.

If the module is to operate autonomously in such a WOR (wake on radio) mode, this can be activated by means of the parameter RF_AutoSleep = 0x01h and configured via the parameters `APP_WOR_PeriodH`, `APP_WOR_PeriodL`, `APP_WOR_MultiplierH`, `APP_WOR_MultiplierL`, and `APP_RX_Time`.

# 11 Firmware Update

The firmware of the module can be updated with the PC utility "ACC" from version 2.5 over the serial interface. If the module is not connected to a PC, the UART of the module should be made accessible, e.g. by means of suitable connectors. Only the UTDX and URXD signals are needed for this procedure.

A level converter (TTL to RS232) is required for the PC connection.

## 11.1 Firmware Version History

Version 1.0

- First release

Version 1.1

- Error flags

Version 1.2

- Bugfix

Version 1.3

- Addition command CMD_DataReq

Version 1.4

- Addition command CMD_Data_IND
- Bugfix
- Boot Up Factory Reset

Version 1.5

- Bugfix

Version 1.6

- Added support for hardware handshake

## 12 References

[1] Specification wireless M-Bus EN 13757-4:2005

[2] "CC1101 Single-Chip Low-Cost Low-Power RF Transceiver (Rev. B)", Texas Instruments

[3] "AMB8425-M Data Sheet", AMBER wireless GmbH

# 13 Regulatory compliance information

## 13.1 Important notice

The use of RF frequencies is limited by national regulations. The AMB8425-M has been designed to comply with the R&TTE directive 1999/5/EC of the European Union (EU).

The AMB8425-M can be operated without notification and free of charge in the area of the European Union. However, according to the R&TTE directive, restrictions (e.g. in terms of duty cycle or maximum allowed RF power) may apply.

### Conformity assessment of the final product

The AMB8425-M is a subassembly. It is designed to be embedded into other products (products incorporating the AMB8425-M are henceforward referred to as "final products").

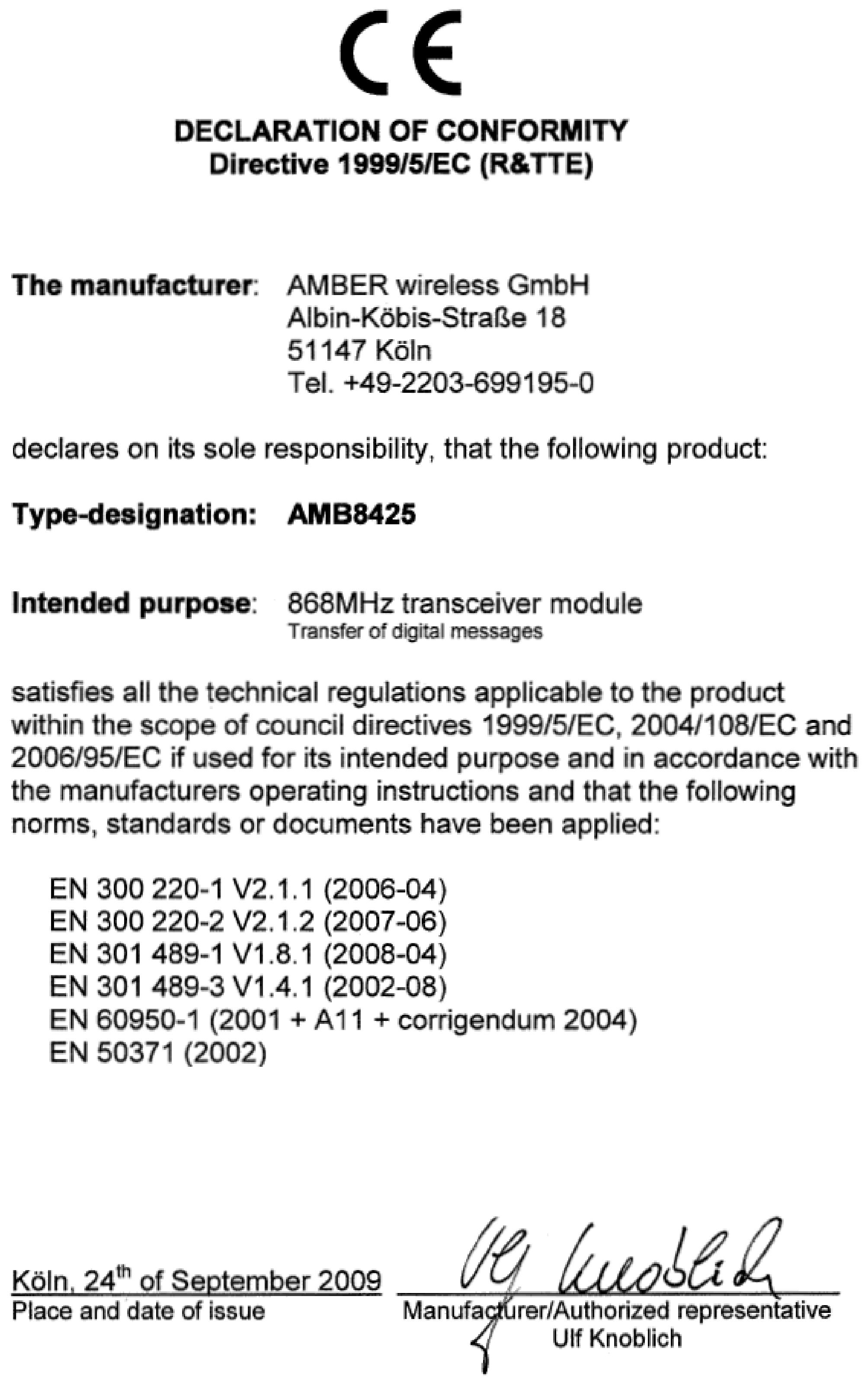
It is the responsibility of the manufacturer of the final product to ensure that the final product is in compliance with the essential requirements of the European Union's Radio & Telecommunications Terminal Equipment (R&TTE) directive.

The conformity assessment of the subassembly AMB8425-M carried out by AMBER wireless GmbH does not replace the required conformity assessment of the final product in accordance to the R&TTE directive!

### Exemption clause

Relevant regulation requirements are subject to change. AMBER wireless GmbH does not guarantee the accuracy of the before mentioned information. Directives, technical standards, procedural descriptions and the like may be interpreted differently by the national authorities. Equally, the national laws and restrictions may vary with the country. In case of doubt or uncertainty, we recommend that you consult with the authorities or official certification organizations of the relevant countries. AMBER wireless GmbH is exempt from any responsibilities or liabilities related to regulatory compliance.

## 13.2 Declaration of Confomity



CE

**DECLARATION OF CONFORMITY**
**Directive 1999/5/EC (R&TTE)**

|  |  |
|---|---|
| **The manufacturer**: | AMBER wireless GmbH<br>Albin-Köbis-Straße 18<br>51147 Köln<br>Tel. +49-2203-699195-0 |

declares on its sole responsibility, that the following product:

**Type-designation: AMB8425**

**Intended purpose**: 868MHz transceiver module
Transfer of digital messages

satisfies all the technical regulations applicable to the product within the scope of council directives 1999/5/EC, 2004/108/EC and 2006/95/EC if used for its intended purpose and in accordance with the manufacturers operating instructions and that the following norms, standards or documents have been applied:

EN 300 220-1 V2.1.1 (2006-04)
EN 300 220-2 V2.1.2 (2007-06)
EN 301 489-1 V1.8.1 (2008-04)
EN 301 489-3 V1.4.1 (2002-08)
EN 60950-1 (2001 + A11 + corrigendum 2004)
EN 50371 (2002)

Köln, 24th of September 2009 [illegible]
Place and date of issue — Manufacturer/Authorized representative
Ulf Knoblich

# 14 Important notes

## 14.1 Disclaimer of liability

AMBER wireless GmbH believes the information contained herein is correct and accurate at the time of this printing. However, AMBER wireless GmbH reserves the right to change the technical specifications or functions of its products, or to discontinue the manufacture of any of its products or to discontinue the support of any of its products, without any written announcement and urges its customers to ensure, that the information at their disposal is valid. AMBER wireless GmbH does not assume any responsibility for the use of the described products, neither does it convey any license under its patent rights, or its other intellectual property rights, or any third party rights. It is the customer's responsibility to ensure that his system or his device, in which AMBER wireless products are integrated, complies with all applicable regulations.

## 14.2 Trademarks

AMBER wireless® is a registered trademark owned by AMBER wireless GmbH. Windows is a registered trademark of the Microsoft Corporation. All other trademarks, registered trademarks and product names are the sole property of their respective owners.

## 14.3 Limitation of use

AMBER wireless products are not authorised for use in life support appliances, devices, or other systems where malfunction can reasonably be expected to result in significant personal injury to the user, or as a critical component in any life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness. AMBER wireless GmbH customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify AMBER wireless GmbH for any damages resulting from any improper use or sale.

Use of AMBER wireless products commits the user to the terms and conditions set out herein.

AMBER wireless GmbH

Albin-Köbis-Straße 18
51147 Köln
Tel. +49 (0) 2203-6991950
Fax +49 (0) 2203-459883
E-Mail info@amber-wireless.de
Internet http://www.amber-wireless.de