AN1119: Using RAIL for Wireless M-Bus Applications with EFR32

This document describes using the Flex SDK for Wireless M-Bus development on EFR32™ Wireless Geckos. It includes features and limitations, using the radio configurator and supplied software components, and a description of the examples

Proprietary is supported on all EFR32FG devices. For others, check the device's data sheet under Ordering Information > Protocol Stack to see if Proprietary is supported. In Proprietary SDK version 2.7.n, Connect is not supported on EFR32xG22.

KEY POINTS

- Features and limitations
- Usage of the configurator in M-Bus mode
- Description of the component
- Description of the examples

Table of Contents

1. Overview

Wireless Gecko supports all Wireless M-Bus PHY configurations, according to EN13757-4 (both directions in all cases):

- Mode S
- Mode T
- Mode R2
- Mode C
- Mode N (all indexes, but 3/9/12 or submodes a, b, c, d, e, f, and g, or channels 1a/b, 2a/b, 3a/b, and 0)
- Mode F

For the RF perfomance of the above modes, see AN1076: EFR32FG1x Wireless M-Bus Performance Measurement Results.

In all modes, we support basic data link layer features for both frame format A and frame format B

- Segmenting frames into blocks, and reassembling them to frame
- CRC calculation and checking
- Frame length decoding
- Data encoding and decoding (Manchester and 3 out of 6)
- Postamble transmission

The software is built on top of RAIL, therefore it inherits RAIL features that can be useful for implementing Wireless M-Bus, such as timestamping or scheduled reception and transmission.

The supplied example includes an easily reusable module that provides some useful features, including mode 5 (AES-128 with dynamic initialization vector) security.

Simplicity Studio 5, used with the Proprietary Flex SDk v3.x, introduced an updated radio configurator graphical user interface (GUI). The illustrations in 3. Using the Radio Configurator have been updated. Places where the functionality diverges from that in Simplicity Studio 4, used with Flex SDK v2.x, are noted. This document applies to both variants.

2. Limitations

2.1 T Mode Meter to Other Device

While the hardware supports 3 out of 6 coding, it cannot generate the postamble required by this mode (this does not matter in receive mode). For T Mode Meter to Other transmission, we recommend using the supplied encoder function (`WMBUS_phy_software() function`), which calculates the CRC using the GPCRC peripheral, encodes the packet using a software based 3 out of 6 encoder, and adds the required postamble.

2.2 Select Frame Type Based on Sync Word

Modes C, N, and F support frame formats A and B, where the sync word selects the frame format. Receiving frame A and frame B with the same configuration is currently not supported. You can either:

- Set up the frameA configuration and receive only frame format A.
- Set up the frameB configuration and receive only frame format B.
- Manually create a multi-PHY configuration and switch between the configurations at runtime. However, you must decide which frame you expect before you switch to RX mode.
- Set up a custom C/N/F configuration with no frame coding option. In that case, you must decode the length and check CRC in software, but you can decide the frame type during reception as well.

Modes that support frame format B always have both sync words configured:

- FrameA and noFrame configurations have sync word 0 configured for frame type A and sync word 1 for type B.
- FrameB configurations have sync word 0 configured for frame type B and sync word 1 for frame type A.

You cannot receive frame type B with a frame A configuration or vice versa, but you can enable the second sync word in RAIL, and use the sync detect event to recognize that you are receiving something with the other frame type.

2.3 Decoder for Both C and T Mode Meter to Other

It is theoretically possible to receive mode C frames with a mode T configuration, because the first 10 bits of the sync word are the same: If the next 6 bits are a valid 3 out of 6 code, it is a T frame, and if the next 16 bits are the second part of mode C's sync word, it is a C frame. Since the second part of mode C's sync word is not a valid 3 out of 6 code, it is not possible that both conditions are true for a correctly formatted frame.

We do not provide a decoder that supports both C and T mode, but a software decoder can be written using the `WMbus T M2O (100k, no framing)` configuration, which is a mode T configuration without a frame decoder. You must set the length during reception when using this configuration in RX mode.

Note, however, that mode C and mode T have different RF performance requirements, so a combined receiver might not be able to meet all of them. See AN1076: EFR32FG1x Wireless M-Bus Performance Measurement Results for details

3. Using the Radio Configurator

For Wireless M-Bus configurations, you should always use the Mbus Profile:

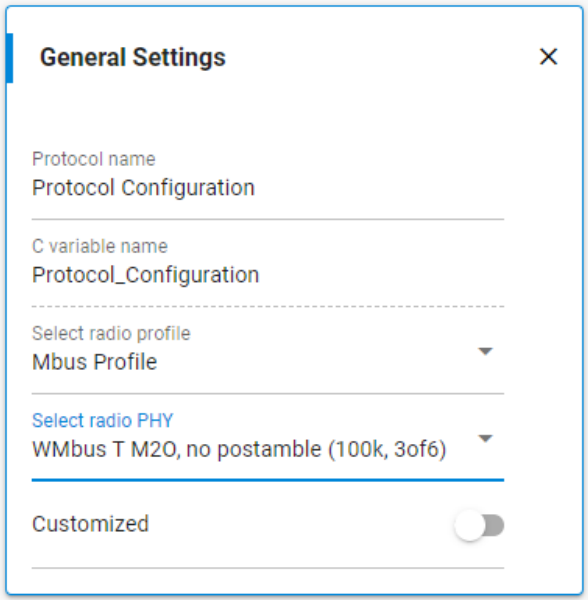


Figure 3.1. Mbus Profile

3.1 Multi-PHY Configuration

The following modes require multiple PHYs:

- Mode T2 (bidirectional only)
- Mode C2 (bidirectional only)
- Mode N has a separate PHY for each bitrate (index 1-5-7-10; 2-6-8-11; 4-18 and 3-9-13) - In the 2013 release of EN13757-4, the naming was different **see Table X**
- Modes C, N, and F have separate PHYs for frame formats A and B

The Radio Configurator can be used to set up virtual channels, for example channel 0 can be used for *Mode T meter to other* and channel 1 can be used for *Mode T other to meter*.

There are other possible uses, such as supporting all 868MHz based modes (S, T and C) with simple channel changes.

It is also possible to configure an application with multiple protocols as Wireless M-Bus modes, that is in order to change protocols RAIL_ConfigChannels must be used. This can be useful to select a mode during boot or configuration. For further details, if you are working with Proprietary Flex SDK v2.x see AN971: EFR32 Radio Configurator Guide for RAIL in Simplicity Studio v4. If you are working with Flex SDK v3.x see *AN1235: EFR32 Radio Configurator Guide for Simplicity Studio 5*.

The **Wireless M-bus Meter** example application is configured to support Mode T2 bidirectional mode using Multi-PHY configuration.

3.2 Recommended Configurations

3.2.1 For 868 MHz and 434 MHz bands

Mode	Submode	Single PHY Configuration Name (frame A) (1)	Single PHY Configuration Name (frame B) (1)	Freq. (MHz)
Mode S	N/A	WMbus S (32.768k, Manchester)		868.3
Mode T	Meter to Other, Rx	WMbus T M2O, no postamble (100k, 3of6) (2)		868.95
	Meter to Other, Tx	WMbus T M2O (100k, no framing) (3)		868.95
	Other to Meter	WMbus T O2M (32.768k, Manchester)		868.3
Mode R2	N/A	WMbus R2 (4.8k, Manchester)		868.33
Mode C	Meter to Other	WMbus C M2O frameA (100k) (4)	WMbus C M2O frameB (100k) (5)	868.95
	Other to Meter	WMbus C O2M frameA (50k) (4)	WMbus C O2M frameB (50k) (5)	869.525
Mode F	N/A	WMbus F, frameA (2.4k) (4)	WMbus F, frameB (2.4k) (5)	433.82

Note:

1. The names were changed since earlier releases to make it more uniform, but it always included the mode, the bitrate and the frame type (where applicable)
2. Could also work in Tx mode, but it does not generate the postamble required by the standard.
3. Should be used with the supplied software encoder (`WMBUS_phy_software() function`).
4. Sync word for frame format A is set up for sync word 0, sync word for frame format B set up as sync word 1, but not enabled by default. Receiving frame format B is not possible, but the sync detect event could be used.
5. Sync word for frame format B is set up for sync word 0, sync word for frame format A set up as sync word 1, but not enabled by default. Receiving frame format A is not possible, but the sync detect event could be used.

3.2.2 For the 169 MHz band (Mode N)

Index (according to EN13757-4-2019)	Single PHY configuration (Frame A) (1,2)	Single PHY configuration (Frame B) (1,3)	Freq. (MHz)	Channel Spacing (kHz)	Last channel number
1 (4)	WMbus N, index 1/5/7/10, frameA (2.4k)	WMbus N, index 1/5/7/10, frameB (2.4k)	169.40625	12.5	5
2 (5)	WMbus N, index 2/6/8/11, frameA (4.8k)	WMbus N, index 2/6/8/11, frameB (4.8k)	169.40625	12.5	5
4 (6)	WMbus N, index 4/13, frameA (19.2k)	WMbus N, index 4/13, frameB (19.2k)	169.4375	50	0
5	WMbus N, index 1/5/7/10, frameA (2.4k)	WMbus N, index 1/5/7/10, frameB (2.4k)	169.48125	12.5	0
6	WMbus N, index 2/6/8/11, frameA (4.8k)	WMbus N, index 2/6/8/11, frameB (4.8k)	169.48125	12.5	0
7	WMbus N, index 1/5/7/10, frameA (2.4k)	WMbus N, index 1/5/7/10, frameB (2.4k)	169.49375	12.5	7
8	WMbus N, index 2/6/8/11, frameA (4.8k)	WMbus N, index 2/6/8/11, frameB (4.8k)	169.49375	12.5	7
10	WMbus N, index 1/5/7/10, frameA (2.4k)	WMbus N, index 1/5/7/10, frameB (2.4k)	169.59375	12.5	17
11	WMbus N, index 2/6/8/11, frameA (4.8k)	WMbus N, index 2/6/8/11, frameB (4.8k)	169.59375	12.5	17
13	WMbus N, index 4/13, frameA (19.2k)	WMbus N, index 4/13, frameB (19.2k)	169.625	50	3

Note:

1. The names were changed since earlier releases to make it more uniform, but it always included the mode, the bitrate and the frame type (where applicable)
2. Sync word for frame format A is set up for sync word 0, sync word for frame format B set up as sync word 1, but not enabled by default. Receiving frame format B is not possible, but the sync detect event could be used.
3. Sync word for frame format B is set up for sync word 0, sync word for frame format A set up as sync word 1, but not enabled by default. Receiving frame format A is not possible, but the sync detect event could be used.
4. Submodes, according to EN13757-4-2013 available on this index: N1c, N2c (on ch1); N1d, N2d (on ch3).
5. Submodes, according to EN13757-4-2013 available on this index: N1a, N2a (on ch0); N1b, N2b (on ch1); N1e, N2e (on ch4); N1f, N2f (on ch5).
6. Same as submode N2g in EN13757-4-2013.

3.3 Using Custom Settings

All the above configurations (and more) can be set up using Custom settings. Currently, to use the multi-PHY features, you have to use custom settings for all but the first (virtual) channel.

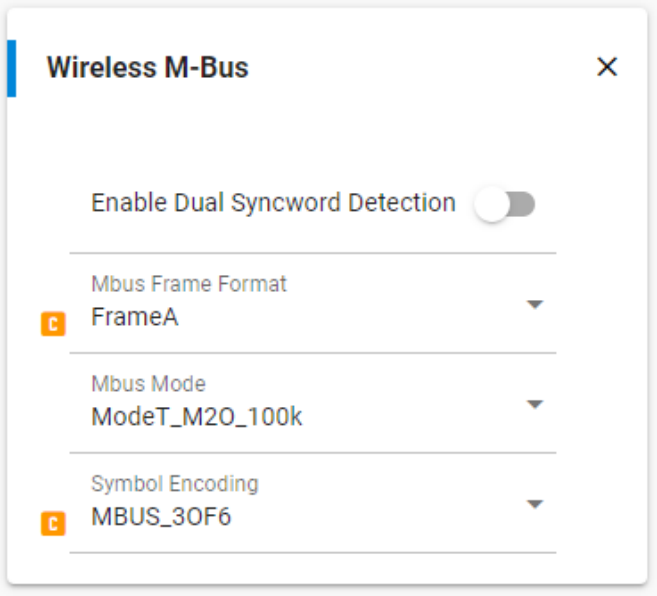


Figure 3.2. Custom Settings

Mbus Frame Format

Sets the frame format to use:

- *FrameA*: Frame format A (10B first block, 16B further blocks, CRC after each block, length does not include CRCs)
- *FrameB*: Frame format B (10B first block, max 115B second block, optional block. CRC after second and optional block, length includes CRCs)
- *NoFormat*: No frame decoder; fixed length mode. Length must be set with `RAIL_SetFixedLength()` before packet Tx and during packet Rx (e.g. using `RAIL_PeekRxPacket` to check the length field when it is received)

Mbus Mode

Sets the mode (modulation, deviation, bit rate, and so on). Possibilities are:

- ModeS_32p768k
- ModeT_M2O_100k
- ModeT_O2M_32p768k
- ModeR_4p8k
- ModeC_M2O_100k
- ModeC_O2M_50k
- ModeN1a_4p8K
- ModeN1c_2p4K
- ModeNg

Symbol Encoding

Sets the symbol coding:

- *NRZ* – No symbol coding
- *Manchester* – Manchester (zero is “10”). Also adds 2 bits of “10” postamble for Tx packets
- *3 of 6* – 3 out of 6 coding. Not recommended for Tx as it does not send postamble.

Enable Dual Syncword Detection

Enables second sync word for ModeC, ModeN, and ModeF

Reconfigure for BER Testing

Test mode for BER testing. For further details, see AN971: EFR32 Radio Configurator Guide for RAIL in Simplicity Studio v4 (Flex SDK v2.x) or *AN1235: EFR32 Radio Configurator Guide for Simplicity Studio 5* (Flex SDK v3.x).

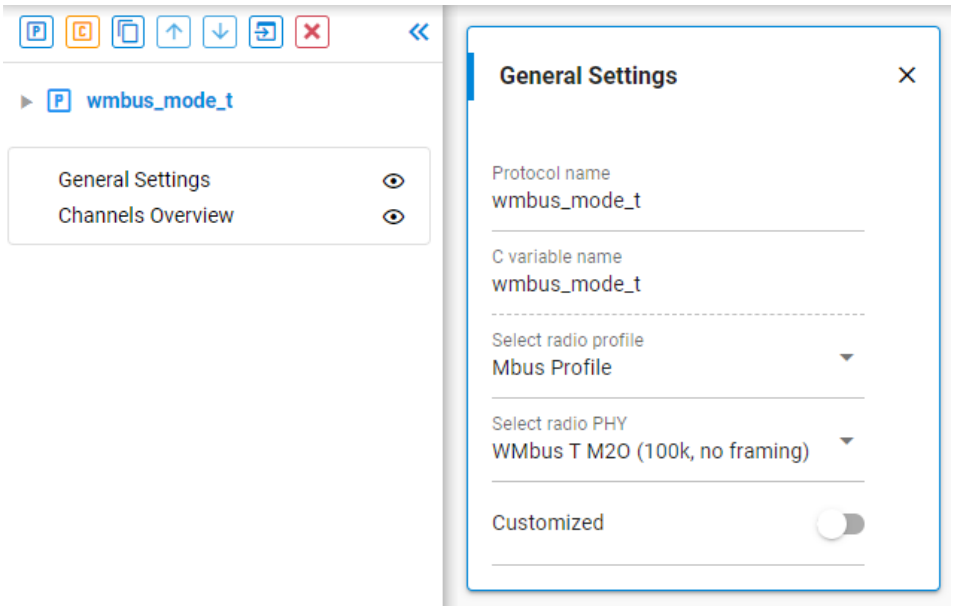
Names and channel group properties

These options are the same as in the general configurator. For further details, see AN971: EFR32 Radio Configurator Guide for RAIL in Simplicity Studio v4 (Flex SDK v2.x) or *AN1235: EFR32 Radio Configurator Guide for Simplicity Studio 5* (Flex SDK v3.x).

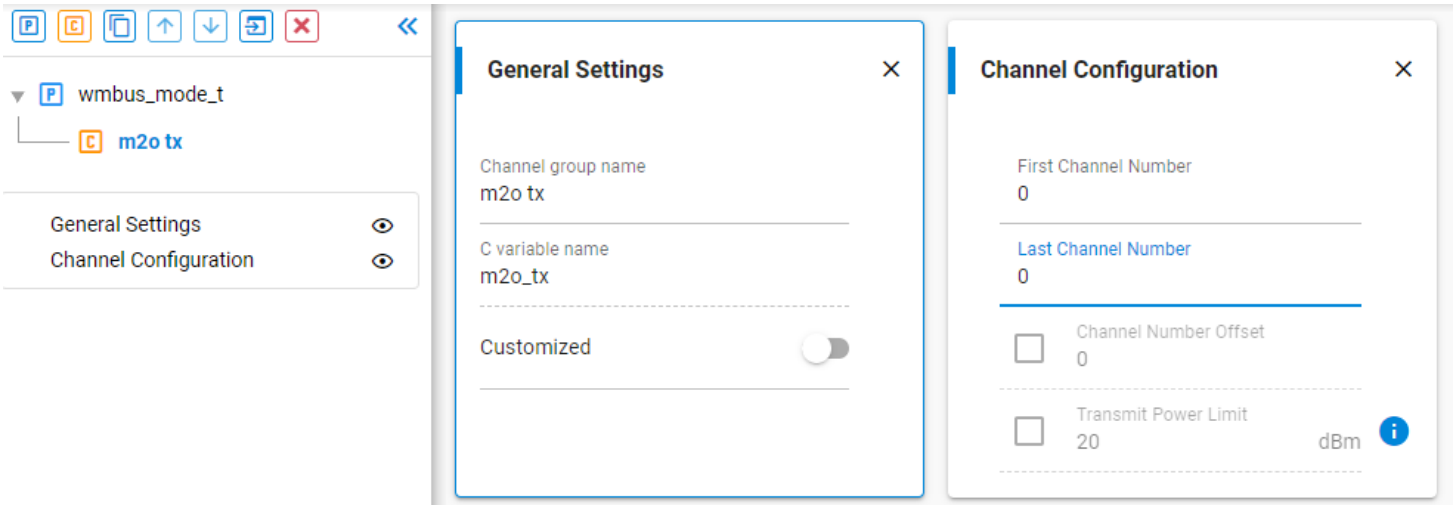
3.4 Using Multi-PHY Features

The following procedure shows how to set up a T2 meter/collector as an example, using Simplicity Studio 5 and Flex SDK v3.x. The settings to use are the same as in earlier configurator versions, but the Multi-PHY configurator workflow changed significantly between Simplicity Studio 4 and Simplicity Studio 5. To apply these changes in Simplicity Studio 4, see AN971: EFR32 Radio Configurator Guide for RAIL in Simplicity Studio v4 for details.

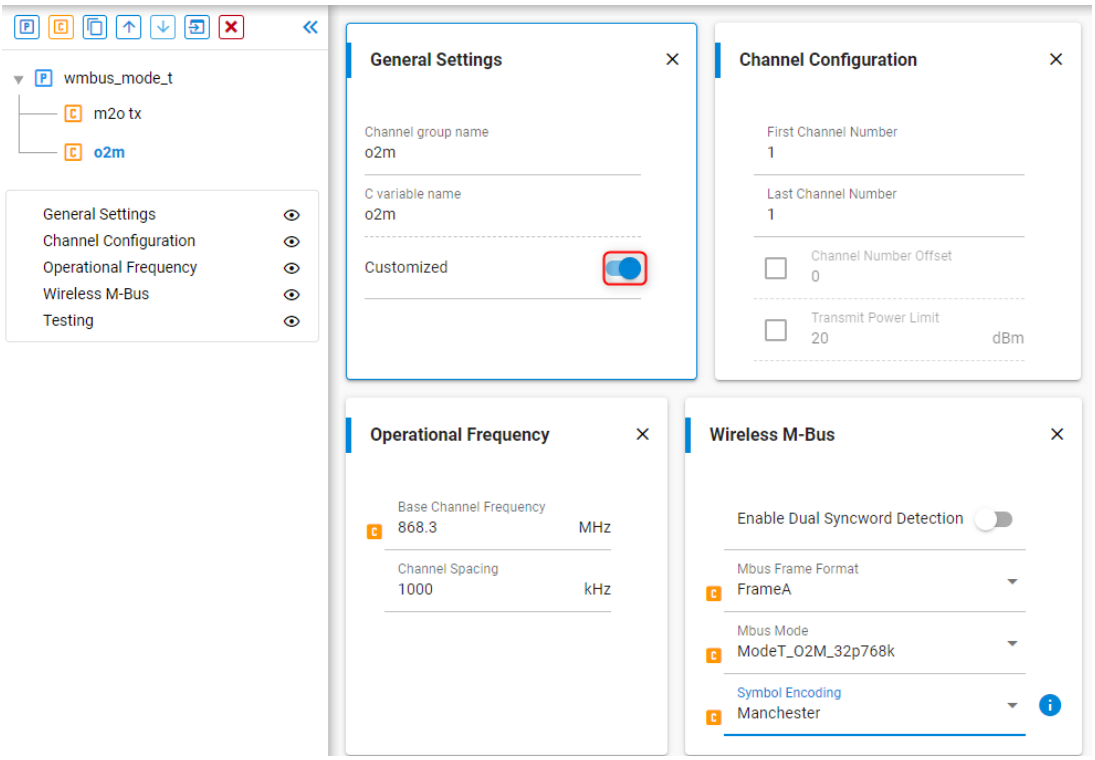
First, select the PHY you wish to use as channel 0 under the protocol setup. In the example, this would be M2O tx setup, which is *WMbus T M2O (100k, no framing)*.



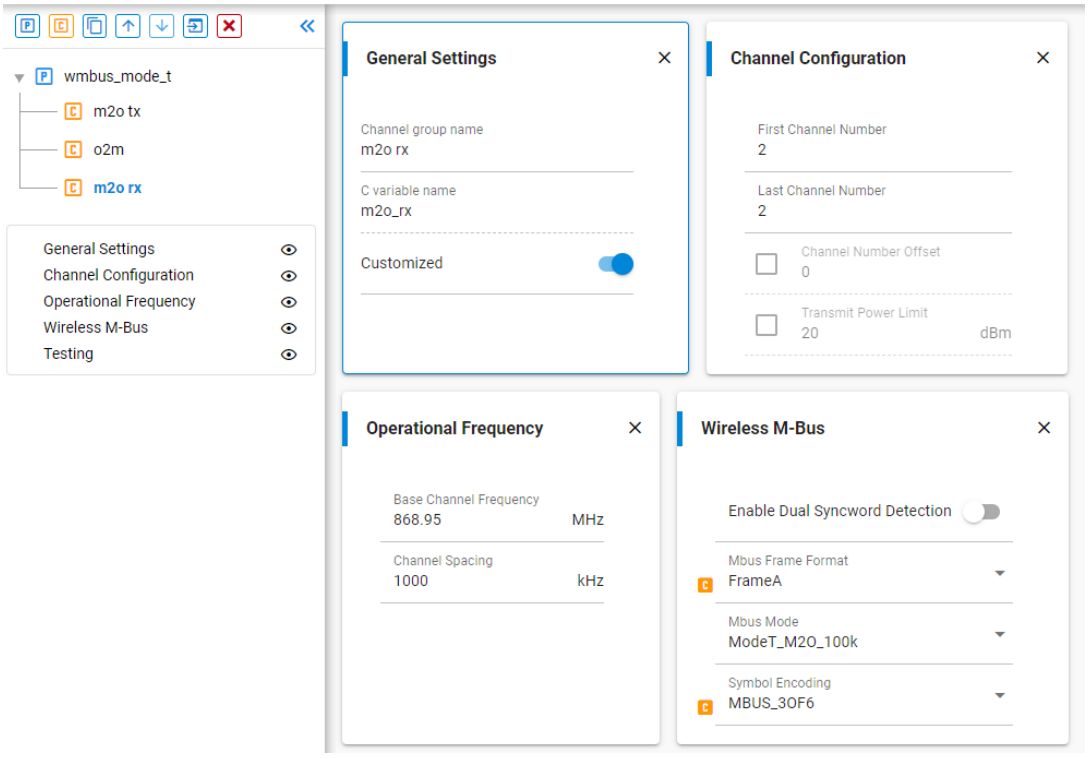
With this, the default channel group is already set for Mode T M2O tx. All that remains is to set the channel numbering and the name.



Create a new channel group by clicking the [C] control on the menu in the upper left. Enable customization on it, then configure it for mode T O2M:



Create and customize a third channel group the same way, and set it up for M2O rx mode:



With this setup, a meter can transmit on channel 0 and receive on channel 1, while a collector can transmit on channel 1 and receive on channel 2.

4. The Wireless M-Bus Support Component

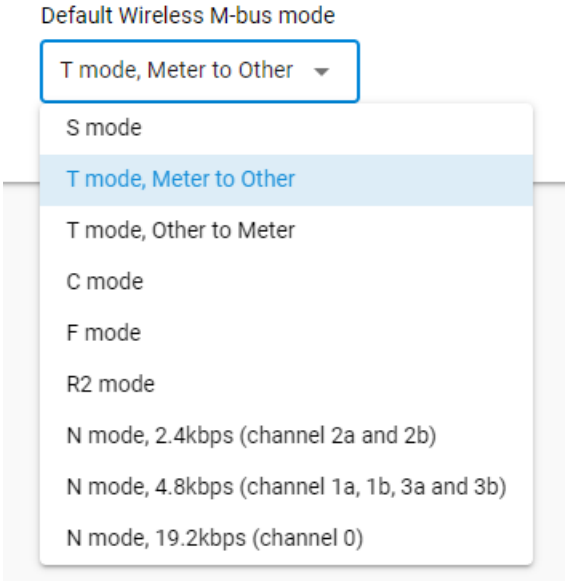
Note: In Flex SDK 3.x, the component can be found under Flex/RAIL/Utility. In Flex SDK v2.x, the wmbus plugin was equivalent to this component.

This component provides the following features:

- Processes the packet before sending it, if needed. Currently needed for mode T meter to other only.
- Helper functions to get the timing in limited access mode
- Helper functions to fill the payload with EN13757-3 compatible payload
- Helper functions to encode/decode manufacturer field
- Helper functions for cryptography using mbedTLS. Currently only supports mode5 (AES128-CBC with dynamic init vector)
- Helper types for data link and transport layer header

For detailed documentation, see `sl_wmbus_support.h` (or in Flex v2.x `wmbus.h`), documented via doxygen style.

In Flex 3.x, the component allows configuration of the wmbus mode used. This selects the timing provided to the application in limited access mode. If *T mode, Meter to Other* is selected and the part requires it, it also enables the workaround described in section 2.1 T Mode Meter to Other Device.



5. Example Application

The provided example application pair implements a very basic meter-collector interaction.

Meter

Note: In Flex SDK 2.x, both TX_CHANNEL and the mode variable can be found in main.c.

The meter periodically sends synchronous SND-NR messages with some hardcoded value with mode 5 security. While waiting, it sleeps in EM2 in idle or EM1 if the main oscillator is required for scheduling or Rx mode. By default, the meter is configured for limited access mode (short receive window after transmission), but it doesn't handle any received packets, it is just implemented to demonstrate the scheduling required for an application.

The application will always transmit on the channel defined by *TX_CHANNEL* in appinit.h (0 by default). On symmetric modes (S, R2, N, F) *TX_CHANNEL* is also used for reception, asymmetric modes (T and C), it will receive on *TX_CHANNEL+1*,

In Flex SDK 3.x, be sure to select the correct mode in the wireless M-bus support component configuration, as described in section 4. The Wireless M-Bus Support Component. In Flex SDK 2.x, the same configuration should be applied in the main.c file:

```
static const WMBUS_Mode_t mode = WMBUS_MODE_T_METER
```

The meter application also includes wmbus_sample_frame.c and .h, which can be used for guidance on how to assemble a simple wmbus frame.

Collector

The collector prints the received packet on serial terminal, with some information (like the first block) detailed. If the packet is EN13757-3 compatible with the short header, it also decodes mode5 security if required.

Once the meter and collector is running, the collector should print messages like this to the serial terminal:

```
RX:[Time:163580960]
Block-1:[L:30,C:0x44,M:SIL,ID:00000001,Version:0x01,devType:0x07]
AppHeader:[CI:0x7A,AccessNr:60,Status:0x00,encMode:5,Accessibility:02,encBlocks:1,sync:1]
[0x2F 0x2F 0x04 0x13 0x39 0x30 0x00 0x00 | 0x02 0x3B 0x7B 0x00 0x2F 0x2F 0x2F 0x2F]
```

A Wireless M-Bus sniffer can also be used. In that case, the default crypto key used by the application is:
`00112233445566778899AABBCCDDEEFF`.

For unencrypted packets, RAILTest also can be used as a sniffer with the right meter to another device to receive configuration.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com