



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

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

KEY POINTS

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

Table of Contents

- 1. Overview 3
- 2. Limitations 4
 - 2.1 T Mode Meter to Other Device. 4
 - 2.2 Select Frame Type Based on Sync Word 4
 - 2.3 Decoder for Both C and T Mode Meter to Other 4
- 3. Using the Configurator. 5
 - 3.1 Multi-PHY Configurator 5
 - 3.2 Recommended Configurations. 6
 - 3.3 Using Custom Settings 7
 - 3.4 Using Multi-PHY Features 8
- 4. The WMBus Plugin 10
- 5. Example Application 11

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 (submodes a, b, c, d, e, f, and g, or channels 1a/b, 2a/b, 3a/b, and 0)
- Mode F

In all modes, we provide support to 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, which provides some useful features, including mode 5 (AES-128 with dynamic initialization vector) security.

2. Limitations

2.1 T Mode Meter to Other Device

While the hardware supports 3 out of 6 coding, it can't generate the postamble required by this mode (this doesn't 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 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.

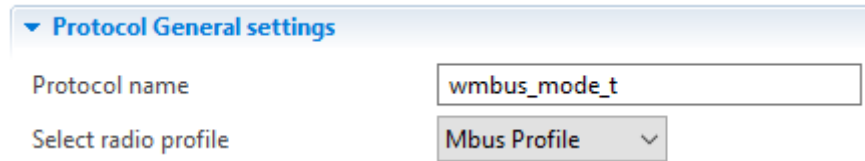
2.3 Decoder for Both C and T Mode Meter to Other

It is theoretically possible to receive mode C frames with a mode T configuration, because the first 10 bits of the sync word is the same: If the next 6 bits is 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 don't provide a decoder that supports both C and T mode, but it's possible to write a software decoder using the `PHY_WMBUS_ModeTC M20 100k noFrame` 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

3. Using the Configurator

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



▼ Protocol General settings

Protocol name

Select radio profile

Figure 3.1. Mbus Profile

3.1 Multi-PHY Configurator

The following modes require multiple PHYs:

- Mode T2 (bidirectional only)
- Mode C2 (bidirectional only)
- Mode N has a separate PHY for each bitrate (channel 1a-1b-3a-3b; 2a-2b and 0)
- Modes C, N, and F have separate PHYs for frame formats A and B

To provide support of these modes, the Wireless M-Bus applications are shipped with the multi-PHY configurator. This 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. However, the example applications do not support this Multi-PHY configuration.

3.2 Recommended Configurations

Mode	Submode	Single PHY Configuration Name	Freq. (MHz)	Ch
Mode S	N/A	PHY wMbus ModeS 32p768k frameA	868.3	0
Mode T	Meter to Other, Rx	PHY wMbus ModeT M2O 100k no postamble frameA ¹	868.95	0
	Meter to Other, Tx	PHY wMbus ModeTC M2O 100k noFrame ²	868.95	0
	Other to Meter	PHY wMbus ModeT O2M 32p768k frameA	868.3	0
Mode R2	N/A	PHY wMbus ModeR 4p8k frameA	868.33	0
Mode C	Meter to Other frame format A	PHY wMbus ModeC M2O 100k frameA ³	868.95	0
	Other to Meter frame format A	PHY wMbus ModeC O2M 50k frameA ³	869.525	0
	Meter to Other frame format B	PHY wMbus ModeC M2O 100k frameB ⁴	868.95	0
	Other to Meter frame format B	PHY wMbus ModeC O2M 50k frameB ⁴	869.525	0
Mode N	a (channel 1a), frame format A	PHY wMbus ModeNabef 4p8K frameA ³	169.406250	0
	b (channel 1b), frame format A	PHY wMbus ModeNabef 4p8K frameA ³	169.418750	1
	c (channel 2a), frame format A	PHY wMbus ModeNcd 2p4K frameA ³	169.431250	0
	d (channel 2b), frame format A	PHY wMbus ModeNcd 2p4K frameA ³	169.443750	1
	e (channel 3a), frame format A	PHY wMbus ModeNabef 4p8K frameA ³	169.456250	2
	f (channel 3b), frame format A	PHY wMbus ModeNabef 4p8K frameA ³	169.468750	3
	g (channel 3b), frame format A	PHY wMbus ModeN2g 19p2k frameA	169.437500	0
	a (channel 1a), frame format B	PHY wMbus ModeNabef 4p8K frameB ⁴	169.406250	0
	b (channel 1b), frame format B	PHY wMbus ModeNabef 4p8K frameB ⁴	169.418750	1
	c (channel 2a), frame type B	PHY wMbus ModeNcd 2p4K frameB ⁴	169.431250	0
	d (channel 2b), frame type B	PHY wMbus ModeNcd 2p4K frameB ⁴	169.443750	1
	e (channel 3a), frame type B	PHY wMbus ModeNabef 4p8K frameB ⁴	169.456250	2
	f (channel 3b), frame type B	PHY wMbus ModeNabef 4p8K frameB ⁴	169.468750	3
	g (channel 3b), frame type B	PHY wMbus ModeN2g 19p2k frameB ⁴	169.437500	0
Mode F	Frame type A	PHY wMbus ModeF 2p4k frameA ³	433.82	0
	Frame type B	PHY wMbus ModeF 2p4k frameB ⁴	433.82	0

Note:

1. Could also work in Tx mode, but it does not generate the postamble required by the standard.
2. Should be used with the supplied software encoder (`WMBUS_phy_software()` function)
3. 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.
4. 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.

3.3 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.

General settings

Mbus Frame Format: NoFormat ▼

Mbus Mode: ModeC_M2O_100k ▼

Symbol Encoding: NRZ ▼

☐ Enable Dual Syncword Detection

Testing

☐ Reconfigure for BER testing

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's received)

Mbus Mode

Sets the mode (modulation, deviation, bit rate, etc). 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. See application note *AN971: EFR32 Radio Configurator Guide* for details.

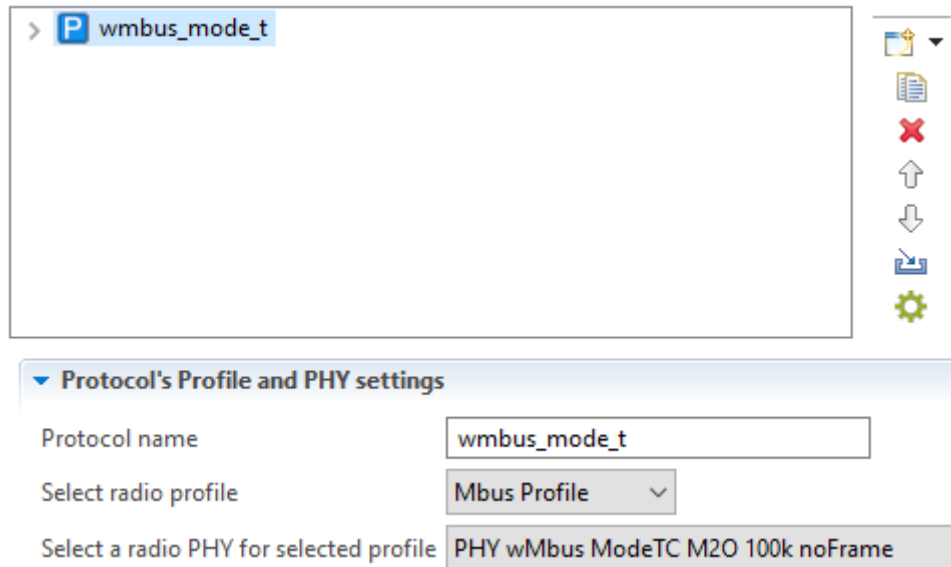
Names and channel group properties

These options are the same as in the general configurator; refer to *AN971: EFR32 Radio Configurator Guide* for details.

3.4 Using Multi-PHY Features

The following procedure shows how to set up a T2 meter/collector as an example.

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 *PHY wMbus ModeTC M2O 100k noFrame*.



> **P** wmbus_mode_t

Protocol's Profile and PHY settings

Protocol name: wmbus_mode_t

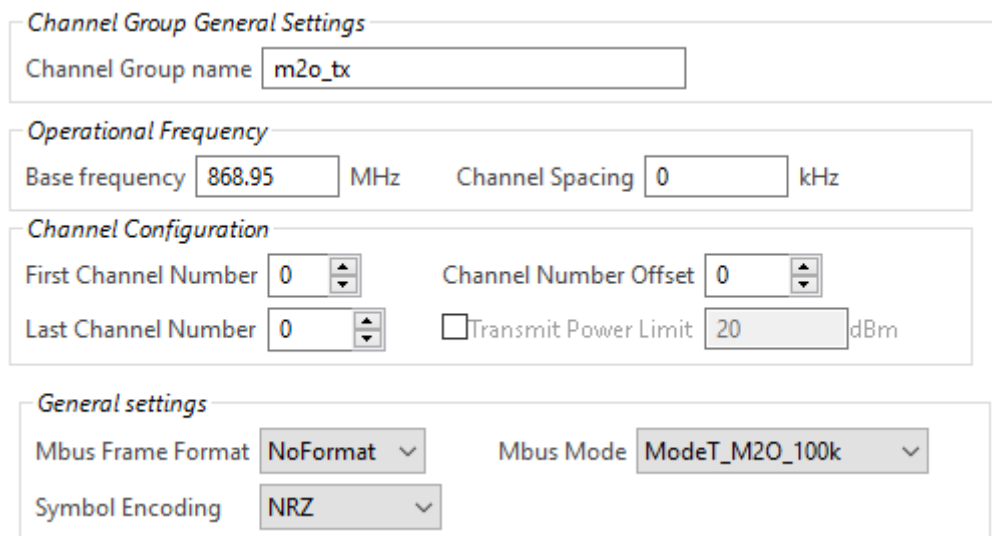
Select radio profile: Mbus Profile

Select a radio PHY for selected profile: PHY wMbus ModeTC M2O 100k noFrame

Next, configure the channel group-based properties using the cogwheel icon and enable the options that will be different between the channels. For mode T meter/collector, this means:

- Mbus Frame Format
- Mbus Mode
- Symbol Encoding

Set up the first channel group to M2O tx:



Channel Group General Settings

Channel Group name: m2o_tx

Operational Frequency

Base frequency: 868.95 MHz Channel Spacing: 0 kHz

Channel Configuration

First Channel Number: 0 Channel Number Offset: 0

Last Channel Number: 0 ☐ Transmit Power Limit: 20 dBm

General settings

Mbus Frame Format: NoFormat Mbus Mode: ModeT_M2O_100k

Symbol Encoding: NRZ

Set up the second channel group to O2M (tx and rx):

Channel Group General Settings

Channel Group name

Operational Frequency

Base frequency MHz Channel Spacing kHz

Channel Configuration

First Channel Number Channel Number Offset
Last Channel Number ☐ Transmit Power Limit dBm

General settings

Mbus Frame Format Mbus Mode
Symbol Encoding

And set up the last channel group as M2O (rx):

Channel Group General Settings

Channel Group name

Operational Frequency

Base frequency MHz Channel Spacing kHz

Channel Configuration

First Channel Number Channel Number Offset
Last Channel Number ☐ Transmit Power Limit dBm

General settings

Mbus Frame Format Mbus Mode
Symbol Encoding

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 WMBus Plugin

The plugin 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 `wmbus.h`, which is documented via doxygen style.

5. Example Application

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

Meter

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 *main.c* (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*,

The application selects timeouts and the rx channel for the mode, so make sure to update the following line for the mode you configured in the radio configurator:

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

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]
```

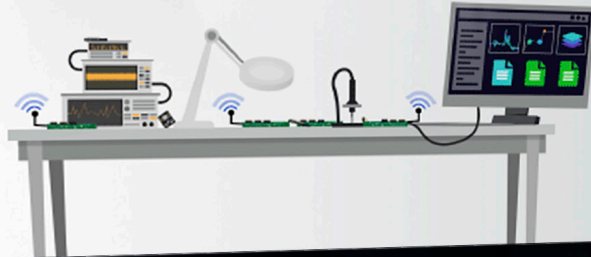
A Wireless M-Bus sniffer can also be used. In that case, the default crypto key used by the application is:

```
00112233445566778899AABBCCDDEEFF.
```

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

Silicon Labs

Simplicity Studio™4



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 and Community
community.silabs.com

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR®, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOModem®, Micrium, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri, Z-Wave, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.



SILICON LABS

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

<http://www.silabs.com>