AN1253: EFR32 Radio Configurator Guide for Simplicity Studio 5

This document describes the Radio Configurator tool provided as part of Simplicity Studio® 5 (SSv5) for Proprietary applications. With the help of the Radio Configurator, users can create standard or custom radio configurations for their RAIL-based radio applications. This document explains the role of each item in the configuration.

If you are working with Proprietary SDK 2.7.n in Simplicity Studio 4, see *AN971: EFR32 Radio Configurator Guide for RAIL in Simplicity Studio 4*.

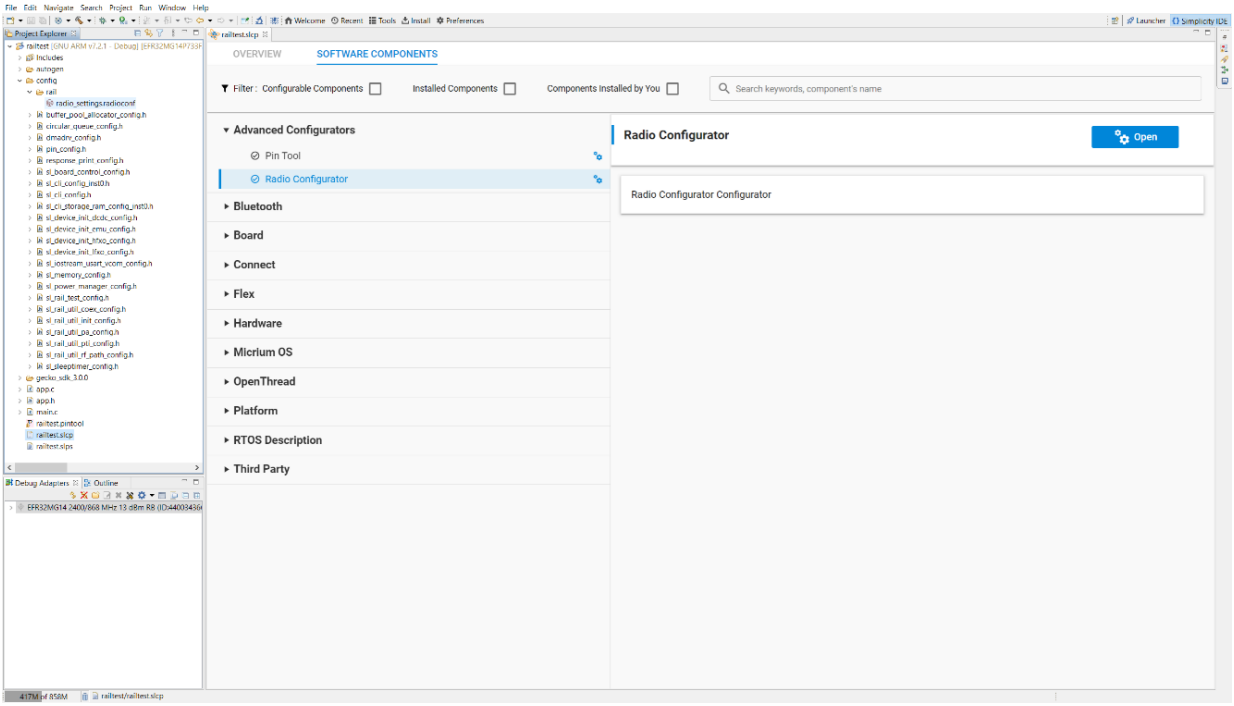
Proprietary is supported on all EFR32FG devices. For others, check the device's data sheet under Ordering Information > Protocol Stack to see if Proprietary is supported. In Proprietary SDK version 2.7.n, Connect is not supported on EFR32xG22.

KEY POINTS

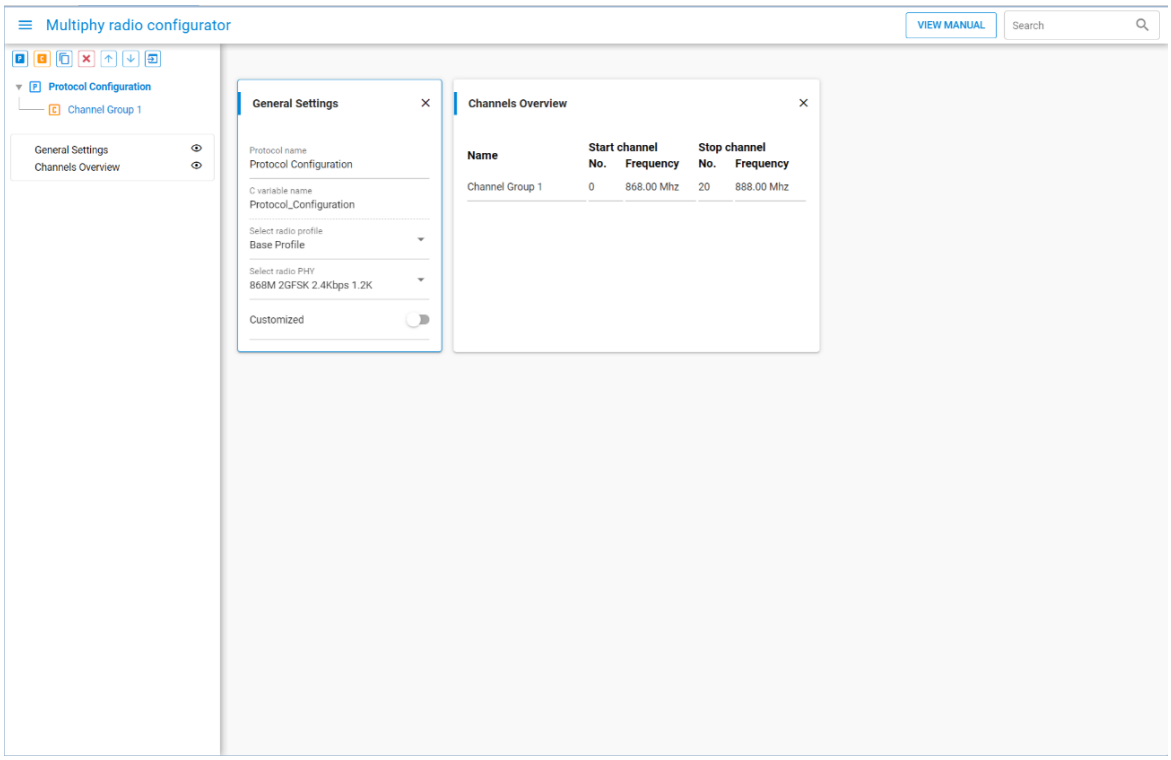
- Radio Configurator for Proprietary applications is described
- User can create custom radio configurations

1. Radio Configuration Flow

Once an EFR32-based project that uses Proprietary protocol (either a project in Flex SDK, or a DMP project) has been created in Simplicity Studio (as described in *QSG168: Silicon Labs Flex SDK v3.x Getting Started Guide*) an .slcp project file is created and an *Overview* tab is opened. Next to it, in the *Software Components* tab, the Radio Configurator can be accessed under the *Advanced Configurators* group. (For some examples, the Radio Configurator might open on project creation). All the radio configurator settings are stored at config/rail/ in the *radio_settings.radioconf* file.



All the parameters in the Radio Configurator are arranged in cards, some of which are grouped together. Each card contains entries that logically go together. Different radio profiles (see section 1.1 Protocols) offer different views and parameter sets as a profile is a high-level view of the parameter set valid for and describing a given radio link.



A radio configuration has two hierarchical levels: Protocol level and Channel Group level. A radio configuration can contain multiple protocols and a protocol can have multiple channel groups defined.

1.1 Protocols

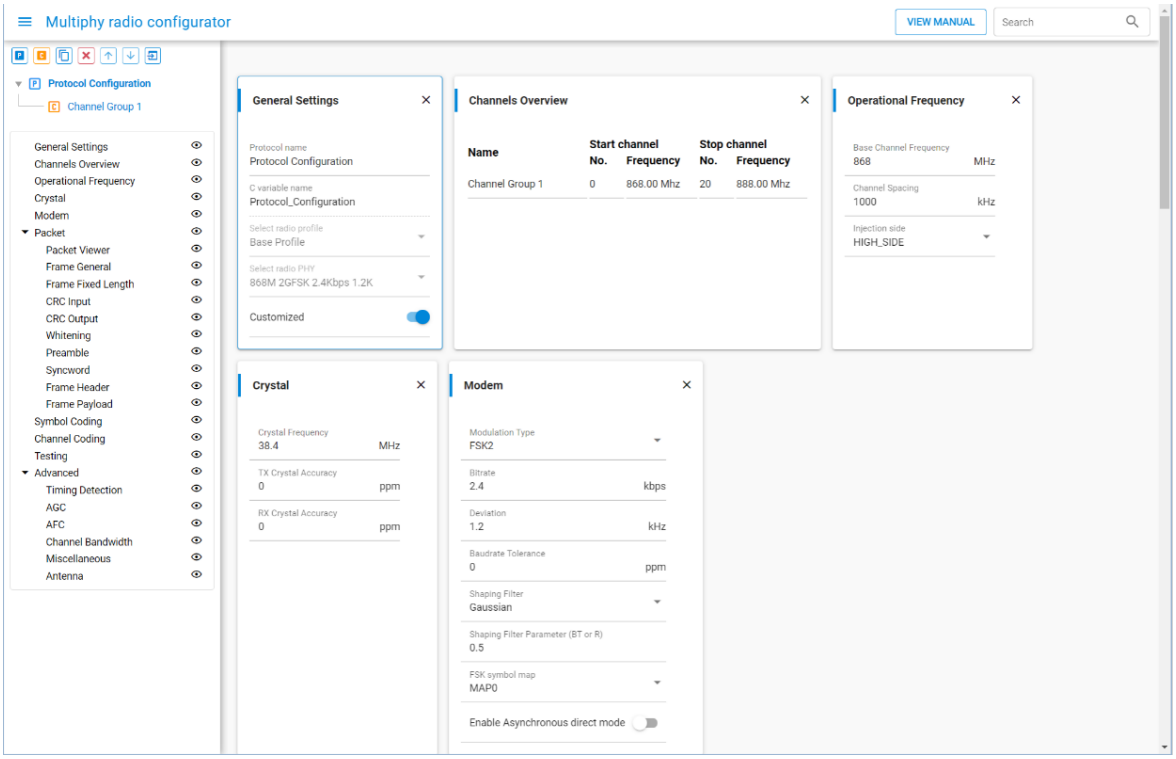
Protocols are complete radio configurations that can be switched using the `RAIL_ConfigChannels()` API, or can be used in Dynamic Multiprotocol applications. For Channel Group definitions see section 1.2 Channel Groups.

To configure a protocol, first **select a predefined PHY** configuration, then **customize** it to meet your needs.

First, look at the **General Settings** card. Select a radio profile in the **Select radio profile** drop-down menu. A radio profile may be any supported radio link technology. These technologies can be bound by standards (for example the Sigfox or WMBus protocols) or can be fully customized. The fully customizable profile is called the "**Base Profile**".

Once the radio profile has been selected, the next step is to select a radio PHY (radio configuration) in the **Select a radio PHY** drop-down list. Each profile has "built-in" configurations ready to use.

Once the radio profile and radio PHY have been selected, users can review the **profile options**. By default, no changes are allowed, fields are grayed out. To enable customize, use the **Customized** switch on the **General Settings** card. This allows access to all the parameters defined by the profile.



Important notes:

1. If you select a "built-in" PHY, and then switch to "Customized ", the Radio Configurator retains the property values of the "built-in" PHY. You can edit the values, but can also revert to the defaults.
2. If you switch to “Customized” mode, we **recommend unchecking all “Advanced”** properties, as those are fine-tuned for the “built-in” PHY, and may not be the optimal choice for the modified PHY. This way those parameters get auto-calculated, and users can experiment with the fine tuning starting up from the calculated values. To keep the improved performance achieved by the original optimization, only minor changes should be made. For example, <100 MHz change in carrier frequency, or a different frame length configuration.
3. If you switch customization off, your **modifications will be reverted** to the property values of the “built-in” PHY.
4. Each menu item in the Navigation pane (on the left) is represented by a card in the main editor panel (on the right). **Cards can be hidden** by clicking the corresponding “eye” icon on the Navigation panel.

Based on the selected radio profile, customizable options may be restricted. For example, if a radio profile is selected that is bound by a standard, the profile options only allow users to set the base frequency. All other options are preset according to the standard.

1.2 Channel Groups

Each protocol configuration includes one or more channel group configurations. Channel groups define one or more (sequential) channels, with a constant channel spacing between them. Channel groups can differ in the radio configuration both from each other and from the parent protocol. By default, a channel group configuration includes only the **General Settings** and **Channel Configuration** cards. Additional parameters defined by the Protocol can be accessed for customization on a channel group basis by sliding the **Customized** switch on the corresponding card.

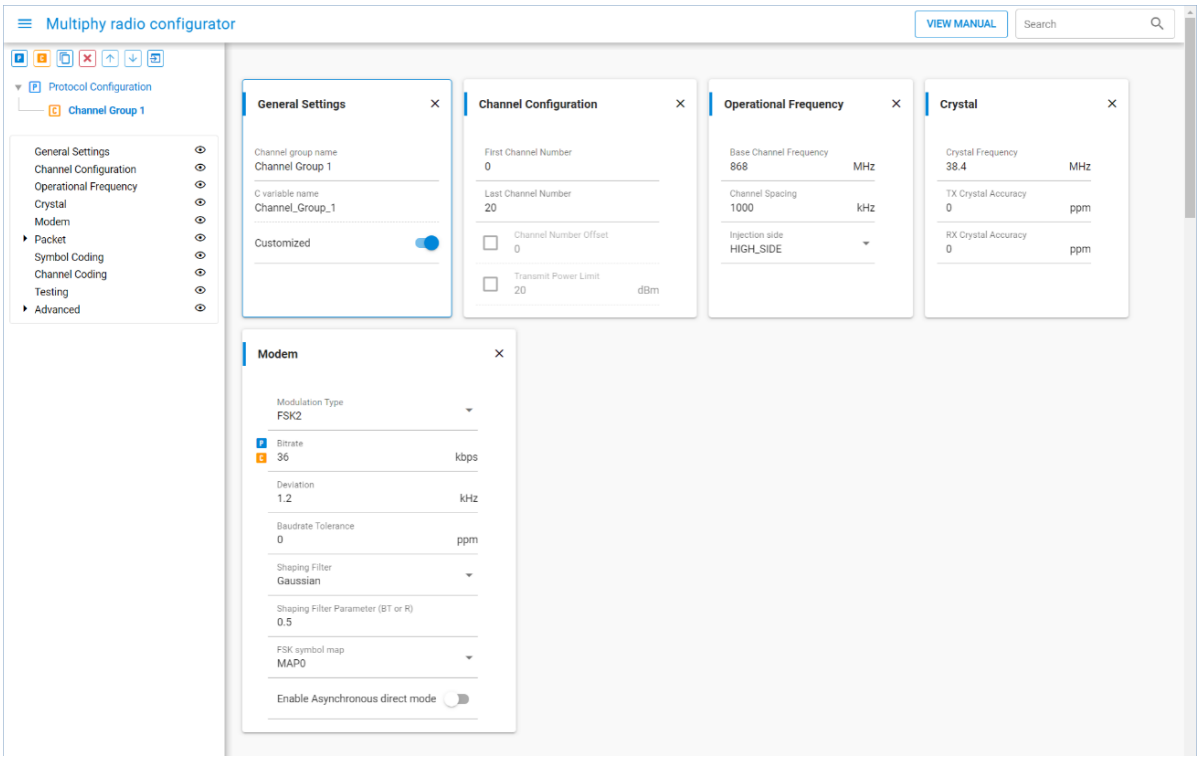
RAIL automatically detects when hopping to a new channel requires hopping between channel groups. The configured property values defined by the channel group will be applied automatically for the new channel. This enables users to define virtual channels to the same physical frequency, but with different configuration settings.

The order of the channel groups will be the same in the `RAIL_ChannelConfigEntry_t` array as shown in the Radio Configurator. Channel group order can be used to override channels in channel groups, as RAIL will always load the channel from the first channel group in which it is defined.

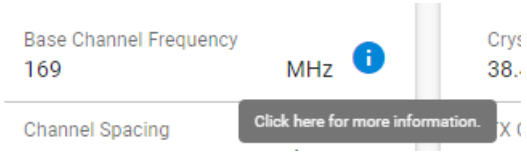
For more on Multi-PHY configuration, see the example in 3. Multi-PHY Configuration Example.

1.3 Finalizing a Configuration

When a parameter is modified from its pre-loaded value, **small pictograms show up** next to the property field on the card. These pictograms symbolize the differences against the originally selected PHY configurations. A “C” means a difference to the original Channel Group property, a “P” means difference to the original Protocol Configuration.



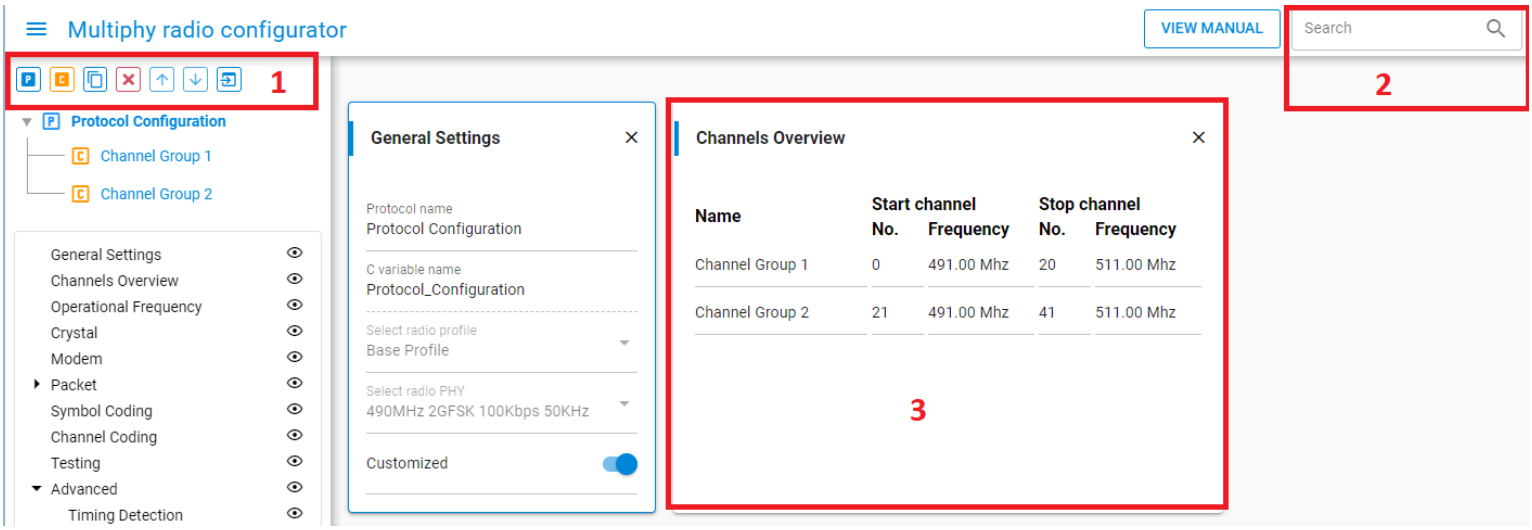
Each input field has an **Information** icon next to it, which opens the embedded version of this user guide at the relevant section. You can also reach the documentation using the [**View Manual**] control at the top right corner of the perspective.



Finally, the Radio Configurator **generates an output upon saving the file** . The generated files are called *rail_config.c* and *rail_config.h* and are located in the *autogen/* folder of your project. These files get also generated upon the creation of a project.

1.4 Other Radio Configurator Controls

The Radio Configurator hosts several new features, to simplify the work with multiple protocols and PHY configurations. This chapter discusses the tools highlighted by red rectangles on the following figure.



1. The tool bar in the top left corner enables the following self-explanatory functions:
   - Add another Protocol Configuration
   - Add another Channel Group
   - Duplicate a selected entry (Protocol Configuration or Channel Group)
   - Delete a selected entry (Protocol Configuration or Channel Group)
   - Move selected item up or down
   - Load a legacy .isc file (from Simplicity Studio 4)
2. The Search window enables real-time filtering of input parameters. Example: typing AGC will hide all the input fields, except those having AGC in the name.
3. The Channels Overview card summarizes the physical frequency assignments for the Channel Groups added under the selected Protocol Configuration.

2. Radio Configurator Cards and Configurable Parameters

2.1 General Settings

Protocol Name (Protocol only)

- Description: Configures the name of the *RAIL_ChannelConfig_t* struct for the protocol. The Radio Configurator will convert this into a C-compatible variable name.
- Unit: String

Channel Group Name (Channel Group only)

- Description: Configures the name of the *RAIL_ChannelConfigEntry_t* struct for the given channel group, and will be used for its *phyConfigDelatAdd* and attr if necessary and/or possible for the protocol. The configurator will convert this into a C-compatible variable name.
- Unit: String

Select radio profile (Protocol only)

- Description: Selects the radio profile for the given protocol. A profile is a high-level view and parameter set valid for and describing a given radio link, for example the WMBUS or Sigfox standards.

Select radio PHY (Protocol only)

- Description: Selects the radio PHY (low-level radio configuration) for the given protocol. Each radio profile has "built-in" configurations ready to use. Even for a custom PHY configuration, starting with a built-in config best matching the user's application and then customizing it is recommended (see next field).

Customized

- Description: Unlocks all the customization features, parameter sets to create custom PHY configurations.
- Note: All the customizations revert to default state determined by the selected base configuration (radio PHY) once this field gets toggled off.

2.2 Channel Configuration (Channel Group only)

First Channel Number

- Description: Sets the first channel number in the channel group. The frequency of this channel will be Base_frequency + (First Channel Number – Channel Number Offset) * Channel spacing.
- Min value: 0
- Max value: 65535

Last Channel Number

- Description: Sets the last channel number in the channel group. The frequency of this channel will be Base_frequency + (Last Channel Number – Channel Number Offset) * Channel spacing.
- Min value: 0
- Max value: 65535

Channel Number Offset

- Description:
  - Sets the channel number offset in the channel group. The frequency of any channel will be Base_frequency + (Channel Number – Channel Number Offset) * Channel spacing. Generally, this should be the same as the First Channel Number, so the first channel will be on the base frequency.
  - When disabled, the Channel Number Offset is equal to the First Channel Number.
- Min value: 0
- Max value: 65535

Transmit Power Limit

- Description:
  - Limit the transmit power of the given channel group. RAIL will automatically lower the transmit power when changing to these channels (if needed), and will not allow setting higher transmit power.
  - If a channel is available with various power limits, RAIL will choose based on the order and the configured TX power. For example,. if you have a channel limited to 10 dBm and one unlimited, and the limited channel is defined first, RAIL will select the limited channel up to 10 dBm, and the unlimited above that. This is useful if you need to add some limitation (e.g. different shaping filter) at high power to pass compliance testing.
  - PA conversion should be set up to use this, see *AN1127: Power Amplifier Power Conversion Functions in RAIL 2.x* for details.
  - When disabled, transmit power is not limited on the channel group.
- Unit: 0.1 dBm
- Min value: -3276.8 dBm (further limited by the part's minimum Tx power)
- Max value: 3276.7 dBm (further limited by the part's maximum Tx power), or unlimited when disabled

2.3 Operational Frequency

The operational frequency card contains entries for setting the operational frequency, also referred to as the base channel frequency, as well as the channel spacing value. The channel spacing value is used for relative frequency configuration, where a frequency can be configured to be so many channel spacing away from the base channel frequency. The channel number can be passed to the RAIL_StartTx and RAIL_StartRx API functions.

Base Channel Frequency

- Description: Sets the base channel frequency/operational frequency
- Unit: MHz
- Min value: 100
- Max value: 2480
- Applicability: Tx and Rx

Channel Spacing

- Description: Sets the channel spacing frequency
- Unit: kHz
- Min value: 0
- Max value: 10 000
- Applicability: Tx and Rx

2.4 Crystal

The Crystal card parameters configure the frequency and the accuracy of the reference clock source (XO/ TCXO). Note that crystal accuracy has two entries, one for Rx and one for Tx. Both numbers are needed to calculate the worst-case frequency offset between the nodes and to select a channel filter in the Rx side accordingly. The suggested channel filter BW is calculated based on these inputs, except for OOK modulation (for more information see section 2.10.4 Channel Bandwidth).

An on-chip capacitor bank presents the desired load to the HF crystal, to oscillate at the desired nominal frequency. It can be configured through the `RAIL_GetTune()` and `RAIL_SetTune()` APIs after RAIL is initialized, or through the CMU API before that. See further notes on this topic in *AN0016.1 Oscillator Design Considerations*.

RX Crystal Accuracy in ppm

- Description: Sets the Rx node crystal accuracy. Crystal accuracy depends on the crystal oscillator used on the receiver side. Rx Crystal Accuracy and Tx Crystal Accuracy attribute has impact on the receiver's acquisition channel bandwidth calculation.
- Unit: ppm
- Min value: 0
- Max value: 200
- Applicability: Rx

TX Crystal Accuracy in ppm

- Description: Sets the Tx node crystal accuracy. Crystal accuracy depends on the crystal oscillator used on the receiver side. Rx Crystal Accuracy and Tx Crystal Accuracy attribute has impact on the receiver's acquisition channel bandwidth calculation.
- Unit: ppm
- Min value: 0
- Max value: 200
- Applicability: Tx

Crystal Frequency

- Description: Sets the crystal frequency. Used by the register calculator and RAIL.
- Unit: MHz
- Min value: 38
- Max value: 40
- Applicability: Tx and Rx

2.5 Modem

The Modem card contains the basic modem configuration entry fields including the modulation format, bit rate, and symbol filtering.

Modulation Type

- Description: This is an enumerated list that contains all the available modulation formats. Note that if a modulation format is not supported on a given platform it is not shown in the list.
- Units:
  - **FSK2**: Frequency Shift Keying on two frequencies. Symbol shaping is also available on this modulation format. See the 'Shaping Filter' entry description later in this section for details. For GFSK2 (Gaussian Frequency Shift Keying) also select this item and pair it with a Gaussian shaping filter.
  - **FSK4**: Frequency Shift Keying on four frequencies. For GFSK4 (Gaussian Frequency Shift Keying) also select this item and pair it with a Gaussian symbol shaping filter.
  - **OOK**: On Off Keying. OOK is an amplitude (or pulse) modulation where one symbol is represented by the presence of an RF carrier while the other symbol is represented by the absence of an RF carrier. Symbol shaping as selected by the shaping filter entry is applied in the amplitude domain to prevent spectral splatter induced by abrupt power level changes.
  - **MSK**: Minimum Shift Keying. MSK is a phase modulation where one symbol is represented by a positive 90 degree phase shift on the carrier (with regards to the preceding symbol) and the other symbols is represented by a negative 90 degree phase shift (with regards to the preceding symbol). This scheme is implemented as FSK2 whose frequency deviation is 1/4th of the data rate. Please note that even though the deviation is fixed for MSK implementation it still needs to be set in the deviation entry. Symbol shaping is performed in the frequency domain.
  - **OQPSK**: Only the half-sine shaped Offset Quadrature Phase Shift Keying is supported. QPSK is a phase modulation format where symbols are represented by 0, 90, 180 and 270 degree phase rotations with regards to the carrier. OQPSK is a modified version of QPSK where only 90 degree phase shifts are allowed at any one time. This is achieved by offsetting the symbol transitions on the I (In-phase) and Q (Quadrature-phase) components by one half of the symbol time. The resulted modulation format is without any amplitude modulation content, which would otherwise be present at 180 degree phase transitions. Finally, the half-sine symbol shaping in the IQ amplitude domain is achieved by default without any shaping filter. Additional shaping can be enabled to improve spectral (for example. roll-off) properties of the signal if needed, but that would introduce at least inter symbol interference due to signal distortion.
- Applicability: Tx and Rx

Bitrate

- Description:
  - This is the bit rate after channel coding and before symbol coding. Channel coding is done in the FRC (Frame Controller) so the modulator will get the already-coded bit stream. Most of the channel coding mechanisms will add additional bits to the data stream. Therefore if the net (un-coded) DR (data rate) is to be kept at a given level the bitrate must be increased in this entry scaled by the bit increase level at channel coding. For example if block coding is configured whereby every block of 4 bits gets 3 parity bits appended to it, the Bitrate entry must be calculated as DR_net* 7/4.
  - Whenever DSSS symbol coding is applied each bit is replaced by a longer bit sequence (chip) in the modulator. It is important that in such cases the Bitrate entry expects the bit rate as opposed to the chip rate. Chip rate up (down) conversion is done automatically in the modulator (demodulator).
  - Another note of caution here is that some modulation formats (4FSK, OQPSK) carry two bits in each symbol, therefore the symbol rate will be half as much as the bitrate. In such cases, it is still the bitrate that has to go in this field.
- Unit: kbps
- Min value: 0
- Max value: 2000
- Applicability: Tx and Rx

Deviation

- Description: This is the deviation parameter for FSK modulation formats. It is the single sided deviation measured from the carrier. **At 4FSK modulation formats the deviation entry expects the inner deviation measured from the carrier.** The outer deviation will be 3 times the configured inner deviation. At MSK modulation format the deviation must be set to 1/4th of the data rate.
- Unit: kHz
- Min value: 0
- Max value: 1000
- Applicability: Tx and Rx

Baudrate Tolerance

- Description: The demodulator's timing synchronization circuitry allows for compensation for baud rate errors with regard to the nominal configuration. This entry expects the baud rate inaccuracy of the signal transmitted to the receiver. Note that when high offsets are to be compensated for (typically > 2.5 %), the receiver measures the baud rate on the Preamble card and updates its nominal configuration to the measured value at preamble detection. In such a scenario it is recommended that the preamble length be longer by at least 8 bauds compared to low offset cases.
- Unit: ppm
- Min value: 0
- Max value: 100 000 (10%)
- Applicability: Rx

Shaping Filter

- Description: The shaping filter entry describes the filtering type that is applied to each symbol or chip before being modulated onto the carrier. Note that the selected modulation format may restrict the selection of symbol shaping filters. Also note that, for different modulation formats, filtering may get applied in different domains such as frequency or amplitude.
- Units:
  - **None**: No symbol shaping filter is applied. Can be used to obtain true FSK signals in 2FSK and 4FSK modulation formats or to obtain half sine IQ-shaped OQPSK signals. In all other cases some symbol shaping is recommended.
  - **Gaussian**: This filter implements Gaussian pulse symbol/chip shaping. Use this filter to obtain GFSK modulation formats. The BT (Bandwidth Time product) factor of the filter can be set in the Shaping Filter Parameter (BT or R) entry.
  - **Raised_Cosine**: This filter implements raised cosine symbol/chip shaping. The R (roll-off or excess bandwidth) factor of the filter can be set in the Shaping Filter Parameter (BT or R) entry.
  - **Custom_OQPSK**: This filter is specific to the 802.15.4 250 kbps DSSS OQPSK PHY to provide additional frequency domain shaping for better spectral properties of the output signal.
  - **Custom_PSK**: This filter implements a legacy third party MSK scheme where the phase rotation between symbols is 2.2 radian (126 degree) and the peak frequency deviation is close to the DR.
- Applicability: Tx

Shaping Filter Parameters (BT or R)

- Description: This entry either takes the BT (Bandwidth Time product) factor for Gaussian or the R (roll-off) factor for Raised Cosine shaping filters. The meaning of the entry field thus changes with filter selection. Note that the entry has no effect if any of the other shaping filters are selected.
- Unit: N/A
- Min value: 0 (For BT a value of 0 is not allowed, practical values range from 0.25 to 1)
- Max value: 1
- Applicability: Tx

FSK Symbol Map

- Description: This entry defines the symbol mapping at FSK modulation formats. With 2FSK it is simply stating which frequency carries which bit; and there are only two choices. At 4FSK modulation, however, where one frequency symbol carries two bits of information, the number of choices increase to 24 out of which 8 are implemented on the device. This configuration is shared between the modulator and demodulator.
- Units:
  - **2FSK mode**

Table 2.1. 2FSK Mapping Options

2FSK	-dev	+dev
MAP0	0	1
MAP1	1	0

  - **4FSK mode**

Table 2.2. 4FSK Mapping Options

4FSK	-3dev	-dev	+dev	+3dev
MAP0	01	00	10	11
MAP1	11	10	00	01
MAP2	00	01	11	10
MAP3	10	11	01	00
MAP4	10	00	01	11
MAP5	11	01	00	10
MAP6	00	10	11	01
MAP7	01	11	10	00

- Applicability: Tx and Rx

Enable Asynchronous Direct Mode

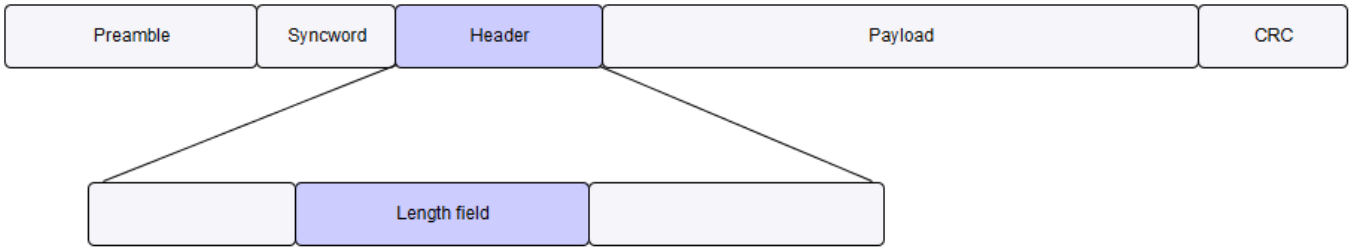
- Description: This entry selects whether the synchronous or asynchronous RX data is feeding the **configured direct mode output pin** when Rx direct mode is enabled from RAIL (see *UG409: RAILtest User's Guide* for more on this mode. Synchronous RX data is only available in packet mode. It starts outputting once sync word is detected and stops at the end of the packet. Synchronous Rx data is synchronized to the recovered bit clock that can also be output. Asynchronous Rx data is always outputting when direct mode is enabled. It is not synchronized to a bit clock and typically has an oversampling rate (with regard to the bit clock) in the range of 4-10.
- Unit: N/A
- Applicability: Rx

2.6 Packet Configuration

2.6.1 Length Configuration

Configurator term definitions are:

- **Header** is an optional field in the packet that starts immediately after the Syncword.
- **Payload** starts after the Header, and ends before the CRC field.
- **Length field** is a 1-12 bit long field in the Header, which stores the length of the Payload in bytes, used for variable length configuration.



The configurator supports three types of payload length configurations:

1. Fixed length
2. Variable length
3. Frame type

Note: The amount of data you load into the transmit buffer **does not** change the configured length. If it is less than the configured length, you will receive a RAIL_EVENT_TX_UNDERFLOW event during transmission. Length configuration applies to both receiver and transmitter.

Available options on the Frame General card, Frame Length Algorithm dropdown:

1. FIXED_LENGTH

All packets have the same payload length, which is configured by the *Fixed Payload Size* field on the **Frame Fixed Length** card. This card is automatically hidden when a different length algorithm is selected.

In fixed length mode, the length can be changed during runtime with `RAIL_SetFixedLength()`. However, RAIL does not handle the header and payload separately, so this API sets the sum of these two.

2. VARIABLE_LENGTH

Each packet can have different payload length, defined by the Length field, described above. The **Frame Variable Length** card is automatically enabled upon selecting this algorithm.

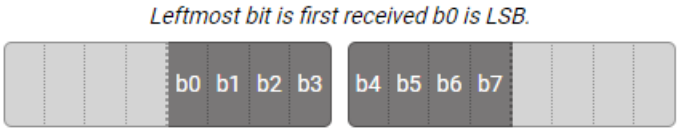
Length Decoding Process

To correctly configure variable length mode, it is useful to understand the length decoding process. The radio first receives the Header, and extracts the last 2 bytes of it for length decoding. If the received header was only 1 byte, the decoder prefixes it with 0x00.

1. If Variable Length Byte Endian is set to LSB first, it swaps the bytes (it does not do anything on a single byte length field).
2. It reverses the bit endianness of both bytes, if Variable Length Bit Endian is not the same as Frame Bit Endian.
3. Shifts right by Variable Length Bit Location number of bits.
4. Masks with an all-one mask with the length of Variable Length Bit Size.
5. Adds Frame Variable Length Adjust to the result.
6. If the resulting length is less than Minimum Length or more than Maximum Length, it aborts the reception, and generates a RAIL_EVENT_RX_FRAME_ERROR, otherwise it continuous reception with the length.

Because *Variable Length Bit Location* is maximum 7 bits, **the last bit of the length field must be in the last byte** of the header.

A graphic on the Frame Variable Length card shows the 2 bytes used by the length decoder and how the length field is located in it.



3. FRAME_TYPE

The details of this method are configured on the **Frame Type Length** card, automatically enabled on selecting this algorithm.

This is an option whereby the frame length information is available in the frame itself in a coded fashion. In other words, the frame information does not explicitly appear as a number but rather as a code that has to be further resolved to a physical length parameter. The code itself can be between 0-7, called Frame Type. This mode is handled slightly differently from RAIL, compared to Fixed and Variable length modes. While in fixed and variable length mode, the length decoder works the same in both receive and transmit. FRAME_TYPE mode is only active in receive. When transmitting, the radio will ignore the length bits and instead transmit the full FIFO.

The following options can be configured on the Frame Type field:

- **Number of Frame Type Bits**: Sets the length of the Frame Type.
- **Frame Type Location**: Sets the location (in Bytes) of the Frame Type.
- **Frame Type Bit Location**: Sets the location (in bits) of the Frame Type.

For each Frame Type, the following can be configured:

- **Accept Frame Type x**: Enables or disables the given frame type .
- **Apply Address Filter for Frame Type x**: Uncheck it to disable the address filter on these types (that is, the address filter is enabled from RAIL, but these frame types would not be filtered).
- **Frame Type x Length**: Sets the length of the frame coded by the given frame type.

2.6.2 Frame General

Header Enable

- Description: This control enables or disables the HEADER field in the packet.
- Applicability: Tx and Rx

Frame Coding Method

- Description: This control enables or disables coding methods applied to the packet (that is, after the sync word, including the header, payload and CRC).

  Note: Beginning with the EFR32xG23, the coding method can be set in the Symbol Encoding Mode field on Symbol Coding card in the Radio Configurator.

- Units:
  - **NONE**: No coding/decoding is applied.
  - **UART_NO_VAL**: Start (0) and a single stop (1) bits are inserted before and after each word to be transmitted, respectively. This feature is useful for emulating direct UART data transfer over the air. Data whitening and/or FEC are executed after coding in the Tx chain, and before decoding in the Rx chain. There is no validation in the Rx chain; the start/stop bits are removed from the frame, but not checked.
  - **UART_VAL**: Start (0) and a single stop (1) bits are inserted before and after each word to be transmitted, respectively. This feature is useful for emulating direct UART data transfer over the air. FEC is executed after coding in the Tx chain, and before decoding in the Rx chain. The Rx chain uses validation, an error will result RAIL_EVENT_RX_PACKET_ABORTED. The coding is handled by the same engine that handles whitening, so whitening cannot be used with UART_VAL frame coding.
  - **MBUS_3OF6:** Implements 3 out of 6 coding/decoding, described in EN13757-4 for Wireless M-Bus mode T (replaces every 4 bit with 6 bits, where each 6 bit code word has 3 “1” bits). The Rx chain uses validation, an error will result RAIL_EVENT_RX_PACKET_ABORTED. The coding is handled by the same engine that handles whitening, so whitening cannot be used with MBUS_3OF6 frame coding.
- Applicability: Tx and Rx

Frame Length Algorithm

- Description: This control sets how the frame length information is determined. Note that the frame length does not include the preamble, sync word, and header sections. See section 2.6.1 Length Configuration for more details.
- Units:
  - **FIXED_LENGTH**: Configured via the Frame Fixed Length card.
  - **VARIABLE_LENGTH**: Configured via the Frame Variable Length card.
  - **FRAME_TYPE**: Configured via the Frame Type Length card.

Frame Bit Endian

- Description: This control sets the bit endianness in the whole frame. Remember this has no effect on the preamble and sync word sections. The enumerated options are self-evident.
- Applicability: Tx and Rx

2.6.3 Frame Fixed Length

This card is enabled when FIXED_LENGTH is selected as the Frame Length Algorithm on the Frame General card. See section 2.6.1 Length Configuration for more details.

Fixed Payload Size

- Description: This controls sets the payload length in bytes when FIXED_LENGTH option is selected. Note that while this controls the length of the payload only, the limits are given for the Header+Payload length.
- Unit: byte
- Min value: 1 (including the header field)
- Max value: 4096 (including the header field)
- Applicability: Tx and Rx

2.6.4 Frame Variable Length

This card configures the various properties of the variable length coding scheme (such as location length, endianness of the length word) if the VARIABLE_LENGTH option is selected as the Frame Length Algorithm on the Frame General card. See section 2.6.1 Length Configuration for more details.

Variable Length Bit Endian

- Description: This control sets the bit endianness of the length word. The enumerated options are self-evident.
- Applicability: Tx and Rx

Variable Length Byte Endian

- Description: This control sets the byte endianness of the length word. The enumerated options are self-evident.
- Applicability: Tx and Rx

Length Includes CRC Bytes

- Description: This control decides whether the indicated length includes or excludes CRC bytes.
- Applicability: Tx and Rx

Maximum Length

- Description: If the decoded length in bytes is longer than this value, the packet is discarded.
- Unit: byte
- Min value: 0
- Max value: 4096 - header size
- Applicability: Tx and Rx

Minimum Length

- Description: If the decoded length in bytes is smaller than this value, the packet is discarded.
- Unit: byte
- Min value: 0
- Max value: 4096 - header size
- Applicability: Tx and Rx

Variable Frame Length Adjust

- Description: The decoded length is adjusted by this value (for example, if set to +2, the radio will always download 2 extra bytes).
- Unit: byte
- Min value: Header length - 8
- Max value: 7 - header length
- Applicability: Tx and Rx

Variable Length Bit Size

- Description: This control sets the variable length word's size in bits.
- Unit: bit
- Min value: 1
- Max value: Header size in bits, but less than 12 bits
- Applicability: Tx and Rx

Variable Length Bit Location

- Description: This controls sets the distance (in bits) between the end of the header and the length field. With the default, 0 setting, the end of length field should be at the end of the header. Bigger values mean the length field ends before the end of the header.
- Unit: bit
- Min value:0
- Max value: 7
- Applicability: Tx and Rx

2.6.5 Frame Type Length

This card configures the various properties of the variable length coding scheme if the FRAME_TYPE option is selected as the Frame Length Algorithm on the Frame General card. See section 2.6.1 Length Configuration for more details.

For each frame type the following controls can be set:

Frame TypeX Length

- Description: This control sets the frame length in bytes for type X (X = 0–7).
- Unit: byte
- Min value: 1 (including header field)
- Max value: 4096 (including the header field)
- Applicability: Tx and Rx

Frame TypeX Valid

- Description: This control enables/disables type X frame (X = 0–7).
- Applicability: Tx and Rx

Frame TypeX Address Filter

- Description: This control enables/disables an address filter on type X frame reception (X = 0–7).
- Applicability: Rx

A further three controls describe the location of the type information in the frame.

Number of Frame Type Bits

- Description: This control sets the length of the frame TYPE information in bits.
- Unit: bit
- Min value: 0
- Max value: 7
- Applicability: Tx and Rx

Frame Type Location

- Description: This control sets the location of the byte that contains the TYPE information in the frame. The location is counted from the beginning of the FRAME. Note that the 1st byte in the frame has a location number of 0 (as opposed to 1).
- Unit: byte
- Min value: 0
- Max value: 255
- Applicability: Tx and Rx

Frame Type Bit0 Location

- Description: This control defines the bit location of the LSB bit of the TYPE field within the byte specified by Frame Type Location.
- Unit: bit
- Min value: 0
- Max value: 7
- Applicability: Tx and Rx

2.6.6 CRC Input

This section contains the input configuration options of CRC. CRC is calculated at the TX side on the not yet coded (neither FEC nor symbol coded nor whitened) byte stream. At the receive side the CRC is calculated again on the fully decoded byte stream, and if the result matches the received CRC value, packet reception is deemed successful.

CRC Input Bit Endian

- Description: This control configures in which order the bits in any one byte are fed to the CRC calculator engine.
- Applicability: Tx and Rx

CRC Input Padding

- Description: In some standards the CRC value must be calculated on at least as long a byte stream as the CRC result itself. If this feature is enabled and the byte stream is shorter than this value, additional zero bytes will be appended to it to reach the minimum length. That is, if the CRC calculation has a result of 4 bytes and it is to be calculated on a 3-byte long sequence an additional 0x00 byte will be appended to the byte stream for CRC calculation. Note that that additional 0x00 padding bytes are not transmitted.
- Applicability: Tx and Rx

CRC Polynomial

- Description: This control selects the CRC generator polynomial.

Table 2.3. CRC Polynomial Selections

Enumerated Value	Polynomial	Width [bit]	Comment
Disabled	NA	NA	No CRC will be calculated / checked
CRC_8	X8+X2+X+1	8	
CRC_16	X16+X15+X2+1	16	
CCITT_16	X16+X12+X5+1	16	IEEE 802.15.4
DNP_16	X16+X13+X12+X11+1X10+X8+X6+X5+X2+1	16	w-MBUS
BLE_24	X24+X10+X9+X6+X4+X3+X+1	24	Bluetooth protocol
CRC_32Q	X32+X31+X24+X22+X16+X14+X8+X7+X5+X3+X+1	32	
ANSIX366_1979	X32+X26+X23+X22+X16+ X12+X11+X10+X8+X7+X5+X4+X2+X+1	32	IEEE 802.15.4g
ZWAVE	X8+1	8	ITU-T G.9959 FCS

- Applicability: Tx and Rx

CRC Seed

- Description: This control configures the initial value of the CRC calculation engine also referred to as CRC seed. Note that the length of the seed must be equal to the CRC width as listed in above table.
- Unit: N/A
- Min value: 0x0000 0000
- Max value: 0xFFFF FFFF
- Applicability: Tx and Rx

2.6.7 CRC Output

This section contains the output configuration options of CRC. Once the CRC is calculated, it can be passed through further operations to comply with protocol requirements.

CRC Output Bit Endian

- Description: This control configures in which order the bits are output from the CRC calculator engine.
- Applicability: Tx and Rx

CRC Byte Endian

- Description: This control sets the byte endianness of the CRC at both the Rx and Tx sides. The enumerated options are self-evident.
- Applicability: Tx and Rx

CRC Invert

- Description: This control allows for inverting the calculated CRC at both the Tx and Rx sides.
- Applicability: Tx and Rx

Table 2.4. Some Commonly Used CRC Configurations

Protocol	Polynomial	Seed	Input Bit Endian	Padding	Output Bit Endian	Byte Endian	Invert
IEEE 802.15.4 (2B)	CCIT_16	0x0000	LSB first	False	MSB first	MSB first	False
IEEE 802.15.4 (2B)	ANSIX366_1979	0xFFFFFFFF	LSB first	True	MSB first	MSB first	True
WMBus	DNP_16	0x0000	MSB first	False	MSB first	MSB first	True
Bluetooth	BLE_24	0x00555555	LSB first	False	MSB first	MSB first	False

2.6.8 Whitening

The device allows for data whitening operation that is primarily used to improve the spectral properties of the transmitted signal by making the symbol distribution more even within a packet. It is especially useful to break long zero or one transmissions. Some standards also make this operation mandatory. Whitening is implemented by XOR-ing the data to be transmitted with a pseudo random data stream. At the Rx side de-whitening is done by repeating the same operation.

Whitening Output Bit

- Description: Whitening is implemented with a linear feedback shift register (LFSR). This control sets which bit of the shift register is “tapped” and XOR’ed with the data bit to be transmitted.
- Unit: bit
- Min value: 0
- Max value: 15
- Applicability: Tx and Rx

Whitening Polynomial

- Description: This control sets the polynomial of the LFSR.

Table 2.5. Whitening

Enumerated Value	LFSR Type	Polynomial	Comment
None	NA	NA	Whitening is disabled
PN9	Galois	X9+X5+1	
PN9_BYTE	Fibonacci	X9+X5+1	Byte-wise reversed output of PN9_Fibonacci
PN16	Galois	X16+X14+X13+X11+1	
BLE	Galois	X7+X4+1	BLE
Bytewise XOR	NA	X8+1	Input is bytewise XORed with the LSB of the seed

- Applicability: Tx and Rx

Whitening Seed

- Description: This control configures the initial value of the LFSR that implements whitening.
- Unit: N/A
- Min value: 0x0000
- Max value: 0xFFFF
- Applicability: Tx and Rx

Table 2.6. Some Commonly Used Whitening Configurations

Protocol	Polynomial	Seed	Output Bit
IEEE 802.15.4	PN9	0x01F0	0x00
Bluetooth	BLE	0x0065 (for channel 37)	0x00

2.6.9 Preamble

The preamble card describes how the transmitted preamble appears and how long it is. Note that even though these entries primarily configure the transmitted preamble they also have a significant bearing on the receiver's configuration.

The receiver's timing and frame detection algorithms and control loops (AGC, AFC) will be tailored to the preamble defined in this card. In other words the entries here do not only define the transmitted preamble at the Tx side but also define the **expected preamble** at the Rx side. It follows then that **it is only possible to create symmetrical configurations** as far as transmitted and expected preambles are concerned. That is, it is not possible to create a configuration that has a long preamble transmission in Tx mode and expects a short preamble in Rx mode. For this use case, the TX preamble can prolonged via the `RAIL_SetTXAltPreambleLength()` RAIL API.

Note that in cases where the preamble is too short for reliably doing the bit time synchronization by the receiver, the sync word is used as the qualifier pattern. See more about this use case in section 2.10.1 Timing Detection.

Preamble Base Pattern

- Description:
  - The preamble is composed of a repeating base pattern. This entry defines what this base pattern is. Typically it is chosen to be an alternating 1010 pattern. Note that the shortest binary base pattern for creating such a preamble is: 10 (or 01). The pattern is transmitted MSB first.
  - When DSSS symbol coding is enabled the preamble base pattern defaults to DSSS chipping code base and this entry becomes irrelevant.
  - When Manchester coding is enabled, the preamble does NOT get coded. (Neither does the sync word.) It follows then that whatever is defined here as the base pattern gets directly transmitted.
  - At 4FSK and OQPSK modulations the preamble can only take on values of two symbols only (out of the four). At 4FSK these are the ±3 dev frequency symbols by default, whereas at OQPSK these are the ±90 degree phase symbols by default. Practically this means that the preamble is 2FSK-coded even though 4FSK modulation is selected and MSK-coded even though OQPSK is selected. For example, at 4FSK modulation a preamble base pattern of b01 will look like -3 dev, +3 dev in the air.
- Unit: N/A
- Min value: b0
- Max value: b1111
- Applicability: Tx and Rx

Preamble Pattern Length

- Description: This entry defines the length of preamble base pattern in bits. For example, if a 1010 binary pattern is built from the shortest base pattern of 10, this entry must be set to 2.
- Unit: bit
- Min value: 1
- Max value: b4
- Applicability: Tx and Rx

Preamble Length Total

- Description: This entry defines the overall length of the transmitted (and expected) preamble pattern in bits. Note that this can only be an integer multiple of the length of the base pattern. If this parameter is set to 0 no preamble is transmitted in Tx mode, nor will it be looked for (and detected) in Rx mode.
- Unit: bit
- Min value: 0
- Max value: 2^21 – 1 = 2097151 (If set to maximum the preamble transmission becomes continuous.)
- Applicability: Tx and Rx

2.6.10 Sync Word

These entries configure the sync word in the radio packet. The sync word serves as a delimiter in the radio frame after which the demodulated bits will be stored in the Rx FIFO. Note that inherently sync word also implements packet filtering/addressing functionality, as radio packets with different than expected sync words are dropped automatically.

Note that in cases where the preamble is too short for reliably doing the bit time synchronization, the sync word is used as the qualifier pattern. See more about this use case in section 2.10.1 Timing Detection.

Two sync words can be defined. The radio can either search for sync0 only (default operation), or both (when `RAIL_RX_OPTION_ENABLE_DUALSYNC` is used). In such a scenario the receiver is looking for both words simultaneously and asserts `RAIL_EVENT_RX_SYNC1_DETECT` or `RAIL_EVENT_RX_SYNC2_DETECT` according to the sync word received. Which sync word has been detected is also returned in `RAIL_RxPacketDetails_t` by the RAIL API function `RAIL_GetRxPacketDetails`.

When DSSS symbol coding is applied, the sync word is detected on the already decoded bit stream, whereas when Manchester coding is enabled, the sync word is detected on the raw “chip” stream.

The sync pattern is transmitted MSB first.

Sync Word Length

- Description: This entry defines the length of the sync word(s) in bits. Granularity is one bit.
- Unit: bit
- Min value: 2
- Max value: 32
- Applicability: Tx and Rx

Sync Word 0

- Description: This entry defines sync word 0. **Note that only Sync Word Length number of LS bits are used, *which* are sent out MSB first.** Note that when DSSS symbol coding is enabled, the sync word also gets coded at the Tx side and is detected after decoding at the Rx side. However, when Manchester symbol coding is enabled, the sync word does not get coded at the Tx side and gets detected without any decoding at the Rx side.
- Unit: N/A
- Min value: 0x0000 0000
- Max value: 0xFFFF FFFF
- Applicability: Tx and Rx

Sync Word 1

- Description: This entry defines sync word 1. **Note that only Sync Word Length number of LS bits are used, *which* are sent out MSB first.** Note that when DSSS symbol coding is enabled, the sync word also gets coded at the Tx side and is detected after decoding at the Rx side. However, when Manchester symbol coding is enabled, the sync word does not get encoded at the Tx side and gets detected without any decoding at the Rx side. Note that it also requires `RAIL_RX_OPTION_ENABLE_DUALSYNC` configured with `RAIL_ConfigRxOptions()` to enable it.
- Unit: N/A
- Min value: 0x0000 0000
- Max value: 0xFFFF FFFF
- Applicability: Tx and Rx

Sync Word Tx Skip

- Description: When enabled, the sync word will be not transmitted, and only used for RX.
- Unit: N/A
- Min value: N/A
- Max value: N/A
- Applicability: Tx

2.6.11 Frame Header

The header provides for a logically-separated field from the payload data with possibly different configurations. That is, the header field can be excluded from CRC calculation and whitening.

Another use of the Header is that the length field must be part of the Header, to be more precise, it must be part of the Header's last byte.

Note that RAIL does not have a separate function to write the HEADER content. It is regarded as part of the payload there.

CRC Header

- Description: If this checkbox is enabled the header content is included in the CRC calculation. Note that this does not mean that the HEADER has its own CRC calculated, but rather it is included in the calculation of the CRC that gets appended to the payload field.
- Applicability: Tx and Rx

Whiten Header

- Description: If this checkbox is enabled the header content is whitened as per the whitening configuration.
- Applicability: Tx and Rx

Header Size

- Description: This entry defines the length of the HEADER field in bytes.
- Unit: byte
- Min value: 1
- Max value: 255 (7 - frame length - adjust if variable length is selected)
- Applicability: Tx and Rx

2.6.12 Frame Payload

Payload is the field in the radio packet where useful data is located. The data that goes into this field (combined with the header) is set by the `RAIL_WriteTxFifo()` or the `RAIL_SetTxFifo()` RAIL function.

Insert/Check CRC after payload

- Description: If this checkbox is enabled the payload content is included in the CRC calculation and the CRC is sent out after the payload in TX and checked in RX operation.
- Applicability: Tx and Rx

Payload Whitening Enable

- Description: If this checkbox is enabled the payload content is whitened as per the whitening configuration.
- Applicability: Tx and Rx

2.7 Symbol Coding

The Symbol Coding card contains controls that are related to symbol coding. Symbol coding is the concept that translates bits (or groups of bits) to modulation symbols that represent the bit (or group of bit) in the modulation domain (frequency, phase, amplitude).

The symbol coding supported depends on the device model.

- **NRZ, Manchester, and DSSS.** NRZ (Non-Return to Zero) directly maps bits to modulation symbols. Manchester coding replaces bits (1 and 0) with two chips (10 or 01), and finally DSSS replaces group of bits with a longer configurable chip sequence. The three options are mutually exclusive.
- **Beginning with the EFR32xG23, 3 out of 6, UART (no validation), UART (with validation) and Linecode symbol coding were added to the list .** For earlier models, 3 out of 6, UART (no validation), and UART (with validation) coding is available to set as Frame Coding Method on the Frame General card.

Symbol Encoding

- Description: This drop-down entry selects the symbol coding method.
- Units:
  - **NRZ (Non Return to Zero)**: In practice, this option means that no symbol coding is done and the incoming bits are just simply passed through to the modulator.
  - **Manchester**: Manchester coding replaces each bit on the payload with two chips based on the selection in the Manchester Code Mapping entry. At the Rx side Manchester decoding is done in a similar fashion whereby each pair of chips is replaced by a bit. Manchester coding is recommended in case long zero or one trails of data are to be transmitted especially in OOK and ASK modulation modes. The Manchester coding scheme ensures that there are always chip transitions regardless of the input data, which is beneficial for the slicing (especially at OOK and ASK demodulation) and bit clock recovery circuits in the demodulator. **Note that the chip rate at Manchester coding is the same as the bit rate without Manchester coding. In practice, this means that the payload content as seen in the air will be twice as long in time.** Also note that **Manchester coding does not apply to the preamble and sync word sections.** Manchester coding/decoding is not supported in more than 1 bit / symbol modulation formats (4FSK / OQPSK). When Manchester decoding is enabled preamble polarity must be configured as it will be seen by the Rx for robust operation (that is, the end of the configured preamble and sync word must match the Tx signal). When Manchester decoding is disabled this is not a requirement.
  - **DSSS (Direct Sequence Spread Spectrum)**: DSSS takes a bit or group of bits and replaces them with a longer chip sequence based on the configuration of the DSSS-related entries, an operation also referred to as spreading. DSSS greatly impacts the spectral properties of the transmitted signal. The chip rate in the air can be much higher than the bit rate, which can greatly increase the bandwidth of the signal, hence the name of spread spectrum. The spreading factor is defined as Chip_rate / Bit_rate. At the receive side decoding is done in the demodulator whereby a received chip sequence is detected and replaced by the corresponding bit or group of bits. This operation is also referred to as de-spreading. The chip sequences that represent different groups of bits are chosen to have weak correlation (i.e., they look different) so even if chip sequences are not demodulated perfectly the corresponding bit values can still be detected. This mechanism gives rise to a so-called **processing gain** in the receiver. In practice, it means that sensitivity on the bit sequence will be better than on the chip sequence by as much as the processing gain. The possible maximum value of the processing gain is the spreading factor itself. DSSS is typically used when relatively low data rates are to be transmitted. The advantages of DSSS are more immunity against narrowband interferers and the possibility of using relatively inaccurate reference sources (XOs).
  - **UART_NO_VAL**:Start (0) and a single stop (1) bits are inserted before and after each word to be transmitted, respectively. This feature is useful for emulating direct UART data transfer over the air. Data whitening and/or FEC are executed after coding in the Tx chain, and before decoding in the Rx chain. There is no validation in the Rx chain; the start/stop bits are removed from the frame, but not checked.
  - **UART_VAL**:Start (0) and a single stop (1) bits are inserted before and after each word to be transmitted, respectively. This feature is useful for emulating direct UART data transfer over the air. FEC is executed after coding in the Tx chain, and before decoding in the Rx chain. The Rx chain uses validation. An error results in RAIL_EVENT_RX_PACKET_ABORTED. The coding is handled by the same engine that handles whitening, so whitening cannot be used with UART_VAL frame coding.
  - **MBUS_3OF6:** Implements 3 out of 6 coding/decoding, described in EN13757-4 for Wireless M-Bus mode T (replaces every 4 bits with 6 bits, where each 6-bit code word has 3 “1” bits). The Rx chain uses validation. An error results in RAIL_EVENT_RX_PACKET_ABORTED. The coding is handled by the same engine that handles whitening, so whitening cannot be used with MBUS_3OF6 frame coding.
  - **Linecode:** Maps 0 to 0011 symbol and 1 to 1100 symbol.
- Applicability: Tx and Rx

Manchester Code Mapping

- Description: This drop-down entry selects the Manchester code mapping.
  - **Default:** Replacing 0→01 and 1→10.
  - **Inverted:** Replacing 0→10 and 1→01.
- Applicability: Tx and Rx

Differential Encoding Mode

- Description: A built-in feature in the modem allows for differential encoding on the bit stream. This encoding scheme can be applied to any modulation format where one symbol corresponds to one bit (that is, OQPSK and FSK4 excluded). This coding scheme is applied to the entire frame starting from sync word on (sync word excluded). Note that differential encoding precedes symbol encoding (so Manchester or DSSS coding will come after differential encoding). Also note that DBPSK symbol coding comes on top of this scheme too. Differential coding is a powerful tool to break long zero and one sequences in the data stream. It may however “flatten out” alternating patterns to one or zero trails.
- Unit: Enumeration

DISABLED	Differential Encoding is disabled.
PR0	Tx/Rx the XOR-ed value of the Raw bit and the last Raw bit. Initial Raw bit is 0.
RE0	Tx/Rx the XOR-ed value of the Raw bit and the last Encoded bit. Initial Encoded bit is 0.
PR1	Tx/Rx the XOR-ed value of the Raw symbol and the last Raw bit. Initial Raw bit is 1.
RE1	Tx/Rx the XOR-ed value of the Raw bit and the last Encoded bit. Initial Encoded bit is 1.

- Applicability: Tx and Rx

DSSS Encoding

The following three parameters configure the DSSS symbol coding mechanism. There are mutual restrictions in between these parameters. You can find all the valid combinations in the table below the entry descriptions. Note that it is possible to enter invalid combinations into the entry boxes; in such cases the PHY generation will fail, so make sure you do a double check against the table.

DSSS Chipping Code Base

- Description: This entry serves as the base chip sequence code. All the other codes are generated with binary cyclic right-shifting and/or inversion or complex conjugation. Complex conjugation is used only for OQPSK modulation, since the inverted chip sequence has the same RF signal format, in other words those are strongly correlated. Complex conjugation is a mathematical operation of a complex number where the result of the real part is equal to the original number’s base part and the imaginary part is equal in magnitude but opposite in sign. In a binary representation of a 2-bit symbol (for example for OQPSK) that is the inversion of odd-indexed bit values. The indexing starts with 0 and is calculated from the least significant bit. Inversion is used for the rest of the modulations. Select a code that has weak autocorrelation (that is, the base code and its cyclic-shifted versions have weak correlations.)
- Unit: N/A
- Min value: 0x00 00 00 00
- Max value: 0xFF FF FF FF
- Applicability: Tx and Rx

DSSS Chipping Code Length

- Description: This entry specifies the length of the chipping code. The chipping code length and the spreading factor (see below) determines how many bits are coded into one chip sequence. See the table below for this information.
- Unit: bit
- Valid values: 2, 4, 8, 16, 32
- Applicability: Tx and Rx

DSSS Spreading Factor

- Description: This entry defines the spreading factor that is essentially the chip-rate to bit-rate ratio.

  Note: Rx sensitivity performance degrades if a spreading factor higher than 8 is configured. Therefore, it is recommended that the spreading factor be less than or equal to 8.

- Unit: N/A
- Valid values: 2, 4-32
- Applicability: Tx and Rx

Table 2.7. Valid DSSS Configuration Combinations

DSSS Spreading Factor	DSSS Chipping Code Length	Bit/Chip Sequence	Binary Cyclic Right Shift
4 to 32	4 to 32 (1)	1	0
2	4	2	2
2	8	4	1
4	8	2	4
4	16	4	2
8	16	2	8
8	32	4	4
16	32	2	16

(1) DSSS spreading factor and DSSS chipping code length must be equal.

Each byte of the packet is separated into BCS long chunks, where BCS is the Bit/Chip Sequence (defined in the previous table). On the transmitter's side these chunks will be fetched up with the endianness defined by the corresponding field of the packet (for example, **Frame bit endianness** in case of payload bytes). Then the chunks will be encoded to chip sequences keeping their bit endianness. It means that the endianness settings set the chunk endianness within the byte, and not the bit endianness of the chunk or the chip endianness. On the receiver side the endianness configuration applies similarly to the decoded chunks' endianness within the received bytes. The chip codes are always sent out in LSB first order (the top right bit of the Base is transmitted first).

The number of the chip codes is equal to 2 squared to the power BCS. The chip sequences are generated by shifting the Base by BCRS (Binary-Cyclic-Right Shift defined in the previous table) multiplied by the value of the BCS-1 least significant bits of the chunk and inverted (or complex-conjugated in case of OQPSK modulation) if the chunk's most significant bit is 1. (If BCS=1 only the other sequence is generated by inversion/complex-conjugate operation.)

All the chipping codes for the following example with DSSS Chipping Code Base of "01001101" are listed below.

Table 2.8. Example DSSS Configuration

DSSS Spreading Factor	DSSS Chipping Code Length	Bit/Chip Sequence	Binary Cyclic Right Shift
4	8	2	4

Table 2.9. Example Chip Sequence for 2FSK

Bits (bin)	Operation	Chip Sequence in LSB First Order (Bin)	Chip Sequence in LSB First Order (Hex)
0	Base	1001001001101111	926F
1	Base right-shifted by 2 (B>>2)	1110010010011011	E49B
10	B>>4	1111100100100110	F926
11	B>>6	1011111001001001	BE49
100	B>>8	0110111110010010	6F92
101	B>>10	1001101111100100	9BE4
110	B>>12	0010011011111001	26F9
111	B>>14	0100100110111110	49BE
1000	Base inverted (B!)	0110110110010000	6D90
1001	B!>>2	0001101101100100	1B64

Bits (bin)	Operation	Chip Sequence in LSB First Order (Bin)	Chip Sequence in LSB First Order (Hex)
1010	B!>>4	0000011011011001	06D9
1011	B!>>6	0100000110110110	41B6
1100	B!>>8	1001000001101101	906D
1101	B!>>10	0110010000011011	641B
1110	B!>>12	1101100100000110	D906
1111	B!>>14	1011011001000001	B641

Table 2.10. Example Chip Sequences for OQPSK

Chip Sequence	Bits	Chip Sequence	Notes)
0	0	0111101011001001	Base (B)
1	1	1110101100100101	Base right-shifted by 2 bits (B>>2)
2	10	1010110010010111	B>>4
3	11	1011001001011110	B>>6
4	100	1100100101111010	B>>8
5	101	0010010111101011	B>>10
6	110	1001011110101100	B>>12
7	111	0101111010110010	B>>14
8	1000	1101000001100011	Base complex-conjugated (B*)
9	1001	0100000110001111	Base complex-conjugated and right-shifted by 2 bits (B*>>2)
10	1010	0000011000111101	B*>>4
11	1011	0001100011110100	B*>>6
12	1100	0110001111010000	B*>>8
13	1101	1000111101000001	B*>>10
14	1110	0011110100000110	B*>>12
15	1111	1111010000011000	B*>>14

2.8 Channel Coding

The Channel Coding card contains controls related to channel coding. Channel coding is a mechanism whereby redundant information is added to the bit sequence, based on which the receiver can detect and correct bit errors. This mechanism is also referred to as Forward Error Correction (FEC). The result of FEC is increased sensitivity at the price of a longer data stream. Note that convolutional code type requires a RAM buffer (allocated automatically in *rail_config.c*), with the following size, depending on Constraint length:

Table 2.11. Channel Coding

Constraint Length	Buffer Size (Bytes)
2	16
3	32
4	64
5	128
6	384
7	768

FEC algorithm

- Description: This drop-down selection entry configures the FEC algorithm to be used.
- Units:
  - **NONE**: No FEC is applied.
  - **FEC_154G**: Implements convolutional coding with the following parameters. This scheme is primarily devised for the 802.15.4 OQPSK 250 kbps PHY.

    Code type: Convolutional

    Generator polynomial 0: 0x0D

    Generator polynomial 1: 0x0E

    Constraint length: 5

    Puncturing: None

    Coding rate: ½

    Interleaving: Enabled

    Note that when this FEC coding scheme is utilized the bitrate input in the Modem card must be set to twice the net data rate based on the coding rate of ½ (i.e., two bits are generated for each input bit).
  - **FEC_154G_K7**: Implements convolutional coding with the following parameters. This scheme is primarily devised for the 802.15.4 OQPSK 250kbps PHY.

    Code type: Convolutional

    Generator polynomial 0: 0x6D

    Generator polynomial 1: 0x4F

    Constraint length: 7

    Puncturing: None

    Coding rate: ½

    Interleaving: Enabled

    Note that when this FEC coding scheme is used the bitrate input in the Modem card must be set to twice the net data rate based on the coding rate of ½ (that is, two bits are generated for each input bit).
- Applicability: Tx and Rx

2.9 Testing

The Testing card contains controls that enable/disable special configurations for particular tests. Note that enabling any of these controls overrides a number of other configuration settings without notification.

Reconfigure for BER (Bit Error Rate) testing

- Description: This control enables a BER testing mode that allows for undertaking BER measurements on a **continuous PN9 sequence.** When this checkbox is enabled, a special demodulator/detection configuration is set that can lock onto the longest 1010 trail (and the subsequent pattern) in the PN9 sequence. On top of this a packet configuration is selected that "de-whitens" the continuous PN9 sequence to a static all-zeroes signal. This signal is then observed by the MCU and all the digital one "glitches" are counted as bit errors. BER testing is supported by the RAILtest application. For more details on how to run the tests, refer to *AN972: EFR32 RF Evaluation Guide*.

  Note 1: Whenever this option is enabled, the configuration **can only be used for BER testing purposes.** If packet-based functionality is desired, a new configuration without this option enabled must be generated and loaded. RAILtest does not know what configuration is running "under" it. Therefore, it is the user's responsibility to keep track of this.

  Note 2: No frequency error compensation/cancellation is done in this mode. Make sure the frequency alignment is accurate between signal source and receiver.

  Note 3: Only two state modulation formats are supported **so you cannot use this feature on 4FSK and OQPSK** configurations.

  Note 4: When testing OOK modulation, restart the BER testing from the RAILtest application whenever the test signal power level changes. This is needed for the receiver to update its slicing threshold.

  Note 5: The feature only works with NRZ symbol coding, so make sure MAN coding or DSSS coding is disabled.

  Note 6: This feature is not enabled for EFR32 Series 2 products.
- Applicability: Rx

2.10 Advanced

The Advanced card group contains advanced controls that can be used for fine tuning. Most apply to the demodulator side. If a value is to be tuned (that is, stepped a few notches away from its initial value) its corresponding control must be enabled by checking the box next to it. **The settings in this group are meant for advanced users only.**

2.10.1 Timing Detection

Timing detection is responsible for recognizing a desired signal and extracting the symbol timing information from it. It is implemented as a correlation detector that correlates the expected and the real incoming signal continuously and signals if it has found a valid correlation peak by asserting TIMING_DETECT.

The detector observes the incoming signal for a given observation period, referred to as a timing window, and if the signal within this time period is determined to be a desired signal the detector issues TIMING_DETECT. A signal is determined to be a desired one if the correlation value is higher than a given threshold in the timing window and the demodulated symbol stream contains fewer errors than a given threshold. At TIMING_DETECT the symbol sampling instant is also adjusted to the middle of the symbols.

Detection on additional timing windows (the number determined by the "Number of Timing Windows to Detect" described below) must also yield a positive answer for the detector to assert PREAMBLE_DETECT.

Both the length of the timing window and the number of times it has to consecutively yield a positive answer for a valid PREAMBLE_DETECT are configurable.

Once PREAMBLE_DETECT has been asserted the search for SYNC WORD begins.

Once TIMING_DETECT has been asserted the timing algorithm still runs continuously to compensate for baud rate drifts in the packet. This mechanism is referred to as timing resynchronization. Note that this mechanism does not change the frequency of the nominal bit clock. It only adjusts its phase for optimal sampling.

Number of Errors Allowed in a Timing Window

- Description: This entry sets how many baud errors are allowed (i.e., tolerated) in the demodulated data stream over a timing sequence for valid TIMING_DETECT. If DSSS symbol coding is enabled the number of errors will apply to chips as opposed to bauds.
- Unit: baud
- Min value: 0
- Max value: 4
- Applicability: Rx

Number of Symbols in Timing Window

- Description: This entry adjusts the length of one timing window. By default the length and the bit sequence of a timing window are defined by two entries in the Preamble card. The length is defined by entry “Preamble Pattern Length” and the bit sequence is defined by entry “Preamble Base Pattern”. If Preamble Pattern Length = 4 and Preamble Base Pattern = 5 the binary sequence will be 0101 in a timing window. Note that the number of symbols in a timing window must be an integer multiple of Preamble Pattern Length. A special use case is when the preamble is short, <20 bits. In this scenario it is suggested to set this field to 0, in which case the 2FSK-modulated sync word will be used as the timing qualifier sequence. To increase the accuracy of the timing detection, use a longer sync word (try to include preamble bits as well).
- Unit: symbol
- Min value: 1 (a value of 0 means that sync word becomes the timing sequence)
- Max value: 60
- Applicability: Rx

Timing Resync Period

- Description: Once timing has been detected the demodulator continues to measure and adjust for baud rate differences. This mechanism is referred to as timing resynchronization. This control configures how frequently new timing information should be available for compensation purposes.
- Unit: timing sequence
- Min value: 1 (a value of 0 disables timing resynchronization)
- Max value: 15
- Applicability: Rx

Number of Timing Windows to Detect

- Description: This entry defines the number of consecutive timing windows in which detection must yield a positive answer for a valid PREAMBLE_DETECT. A value of 1 means that TIMING_DETECT and PREAMBLE_DETECT get asserted at the same time.
- Unit: timing window
- Min value: 1 (0 also means 1)
- Max value: 16
- Applicability: Rx

Timing Detection Threshold

- Description: This is a relative threshold level against which the result of the correlation measurement is checked. If the measured correlation is less than this threshold TIMING_DETECT is not asserted. Longer timing windows require a higher threshold. This field affects the ratio of false detects seen. A higher value means stronger qualification, but it can reduce receiver sensitivity.
- Unit: N/A
- Min value: 0
- Max value: 255
- Applicability: Rx

Timing Samples Threshold

- Description: Signal strength can be made to be a requirement for timing detection. If this feature is enabled TIMING_DETECT is not asserted unless all samples taken within the observation window are above a certain threshold level. With amplitude modulation formats (OOK and ASK) TIMING_DETECT is not asserted unless the measured strong /weak signal periods match the baud rate period. This entry sets a relative signal strength level for the above purposes.
- Unit: N/A
- Min value: 0
- Max value: 100
- Applicability: Rx

2.10.2 AGC

Automatic gain control (AGC) is responsible for adjusting the receiver gain to an optimal level on any reception. The optimal level is the minimum gain at which reception is still robust. This approach provides the most headroom for blockers in the receive chain and thus results in good blocking performance.

The AGC consists of two control loops. One is the RF front end control loop, also referred to as the FAST LOOP. The other is the channel filter control loop. The main difference between the two is where exactly the power measurement is taken in the receive chain the loop acts upon. As its name indicates the FAST LOOP is much faster than the channel filter control loop, as the latter includes the propagation delay of the channel filter itself. This delay is inversely proportional to the bandwidth. The AGC operation can be viewed as a coarse and fast control in the FAST LOOP, complemented by a slower, more accurate control in the channel filter loop.

AGC Hysteresis

- Description: A hysteresis feature is provided for the AGC to prevent events whereby the AGC keeps toggling between two gain configurations if the measured power value is at or close to the switching threshold level. This phenomenon is also referred to as AGC chattering. The measurement thresholds for gain changes to opposite directions are shifted by as many dBs as indicated by this entry.
- Unit: dB
- Min value: 0
- Max value: 8
- Applicability: Rx

AGC Power Target

- Description: This is the target power level as measured by the channel filter loop. The AGC drives the gain configuration so that this target value is measured after the channel filter. A higher value gives stronger signal for detection, but reduces headroom against high level interferer signals.
- Unit: dBm
- Min value: -40
- Max value: 8
- Applicability: Rx

AGC Speed

- Description: This entry selects the operation mode of the AGC with regards to its speed. The AGC must be settled by preamble detection so the speed configuration is directly related to the length of the preamble the receiver is expecting and also the bandwidth the receiver is configured to.
- Unit: Enumerated list with self-explanatory items (NORMAL, FAST and SLOW).
- Applicability: Rx

AGC Period

- Description: The channel filter AGC loop measures the true signal power by performing an RSSI (Radio Signal Strength Indicator) measurement. The AGC period adjusts the length of the measurement window. The longer the measurement window the more accurate the measurement will be, however this comes at a price of a longer preamble requirement at the Tx side. The AGC period adjusts the measurement window in the following fashion: T_meas = 2^AGC_PERIOD, where the unit is one symbol.
- Unit: Symbol time
- Min value: 0
- Max value: 7
- Applicability: Rx

AGC Settling Delay

- Description: This time delay parameter configures the time between two gain adjustment cycles in the channel filter loop. Ideally this parameter reflects the delay through the channel filter and the demodulator. The idea is that the effect of one gain adjustment must be propagated through the receive chain before the next adjustment cycle starts. The unit of this parameter is the AGC‘s clock period.
- Unit: AGC clock period
- Min value: 0
- Max value: 63
- Applicability: Rx

AGC Backoff Scheme

- Description: This entry controls in which order the various gain blocks reduce gain in the receive chain. This feature is only applicable to IC revision xG12 and above.
  - **SCHEME_1**: This scheme is the default and recommended configuration. Unless superior blocking performance is required use this. This scheme optimizes for sensitivity under all gain conditions. **All the other schemes trade off sensitivity (under blocking conditions) for blocking performance.** Note that sensitivity in non-blocking conditions is not affected. This is the configuration xG1 IC revision is using.
  - **SCHEME_2**: Not used.
  - **SCHEME_3**: This scheme is used by the built-in BLE and Zigbee 2.4 GHz PHYs.
  - **SCHEME_4**: This scheme is recommended for ETSI category 1-compliant narrowband applications in the 169 MHz band to meet the blocking specifications without a SAW filter.
- Applicability: Rx

2.10.3 AFC

Slightly differing crystal frequencies will cause frequency offsets between Tx and Rx nodes at the carrier frequency. The receiver has a mechanism that can compensate or cancel this frequency offset. Offset compensation happens inside the demodulator. The measured frequency offset shifts the slicing threshold for frequency and phase modulated signals. Offset cancellation happens with an interaction between the demodulator and the PLL synthesizer whereby the measured frequency offset (going through some gearing) directly adjusts the synthesizer frequency. Offset compensation and offset cancellation are mutually exclusive.

Offset cancellation is referred to as automatic frequency control (AFC), while offset compensation is referred to as INTERNAL compensation.

AFC yields better sensitivity performance in the entire frequency error range as it tunes the receive filter onto the incoming signal. With INTERNAL compensation the receive filter is not tuned at all so when the incoming signal gradually drifts out of the receive filter bandwidth, sensitivity also degrades gracefully.

Neither AFC nor INTERNAL compensation is available on amplitude modulated signals such as ASK and OOK.

Both the AFC and INTERNAL compensation algorithms can be frozen at particular events during packet reception. This has the benefit that, once frequency offset has been acquired and cancellation has been settled through the synthesizer with AFC, there is no risk of losing frequency alignment to subsequent, less-accurate offset measurements. A prime example is a packet payload with many repeating consecutive symbols that make the frequency offset measurement difficult.

The events at which freezing is possible: TIMING_DETECT, PREAMBLE_DETECT and FRAME_DETECT. FRAME_DETECT signals SYNC WORD detection.

Typically both AFC and INTERNAL compensation starts immediately upon entering into Rx mode. AFC, however, also has a feature that makes the algorithm start only at PREAMBLE_DETECT. This has the benefit that the AFC algorithm does not “wander off” too far on noise before a valid packet arrives. A downside of this approach is that the range of the AFC will be smaller as the preamble must be detected without any offset compensation.

The controls in this card allow for fine-tuning some of the parameters of the AFC and INTERNAL (frequency offset) compensation algorithms.

Frequency Offset Period

- Description: This configuration provides gearing on the length of the calculated elementary frequency error measurement window. The final measurement window is elementary window * 2^Frequency_Offset_Period.

  When only INTERNAL compensation is enabled the length of the elementary window equals the timing sequence. When AFC is enabled the elementary window is typically chosen to be 4-8 baud long.
- Unit: as per above
- Min value: 0
- Max value: 7
- Applicability: Rx

Frequency Compensation Mode

- Description: This drop-down list enables selection on compensation mode, freezing event and AFC starting event.

  For AFC operation at least 40 bits of preamble is required.
- Units:
  - **Disabled**: This option disables both the AFC and interval compensation mechanisms, which may be useful for debugging purposes.
  - **INTERNAL_LOCK_AT_PREAMBLE_DETECT**: Internal compensation is enabled and it gets frozen at PREAMBLE DETECT.
  - **INTERNAL_LOCK_AT_FRAME_DETECT**: Internal compensation is enabled and it gets frozen at FRAME DETECT (i.e., at SYNC WORD detect).
  - **INTERNAL_ALWAYS_ON:** Internal compensation is enabled and it never gets frozen. This is an option that is recommended in direct mode.
  - **AFC_FREE_RUNNING**: AFC is enabled and it never gets frozen. This is an option that may be used in direct mode with long (> 40 bit) preambles.
  - **AFC_LOCK_AT_PREAMBLE_DETECT**: AFC is enabled and it gets frozen at PREAMBLE_DETECT.
  - **AFC_LOCK_AT_FRAME_DETECT**: AFC is enabled and it gets frozen at FRAME DETECT (i.e., at SYNC WORD detect).
  - **AFC_START_AT_PREAMBLE_LOCK_AT_FRAME_DETECT**: AFC gets enabled at PREAMBLE DETECT and it gets frozen at FRAME DETECT (i.e., at SYNC WORD detect). This may be a viable option for small frequency offsets in packet mode when preamble length >= 40 and sync word is DC balanced.
  - **AFC_START_AT_PREAMBLE_FREE_RUNNING**: AFC gets enabled at PREAMBLE DETECT and it never gets frozen afterward. This may be a viable option for small frequency offsets in packet mode when preamble length >= 40 and the payload has very long DC balanced content (long enough that frequency drift could potentially occur).
- Applicability: Rx

AFC Frequency Limit

- Description: This control puts a limit on the absolute measured frequency error beyond which no compensation occurs. This is useful for stopping the radio from receiving packets above certain absolute frequency offsets (such as packets in the adjacent channel).

  Note: Setting this value to 0 disables the limiting mechanism.
- Unit: kHz
- Min value: 0
- Max value: 500
- Applicability: Rx

AFC Step Scale

- Description: This configuration is only applicable when in AFC mode. The control allows for scaling the default feedback gain to the PLL synthesizer.
- Unit: N/A
- Min value: 0
- Max value: 2
- Resolution: 0.01
- Applicability: Rx

AFC Period

- Description: Controls the frequency offset average period for AFC.
- Unit: N/A
- Min value: 0
- Max value: 7
- Resolution: 1
- Applicability: Rx

2.10.4 Channel Bandwidth

Acquisition Channel Bandwidth

- Description: The channel filter bandwidth is calculated automatically from the modulation and XO accuracy parameters, this control allows for manually overriding it to a desired value. Note that there are restrictions on the filter bandwidths so the closest available gets selected.

  Note 1: The crystal accuracy input parameter is NOT used for OOK BW automatic calculation. The target is BW 75*Data rate, favoring those legacy applications using low data rate with inaccurate frequency sources.

  Note 2: For FSK, a good rule of thumb is the Carson formula (BW=2*Deviation+DataRate).

  Note 3: The actual bandwidth corresponds to the 3dB point of the channel filter bandwidth and may differ from the desired bandwidth. The actual bandwidth is different due to rounding and depending on the decimation settings and other internal front-end settings.
- Unit: kHz
- Min value: 0.1
- Max value: 2530
- Applicability: Rx

IF Frequency

- Description: Although the IF frequency is calculated automatically from the modulation and XO accuracy parameters, this control allows for manually overriding it to a desired value. This may be important if the image must not fall on a specific frequency. Note that there are restrictions on the IF frequency so the closest available gets selected.
- Unit: kHz
- Min value: 150
- Max value: 1900
- Applicability: Rx

LO Injection Side

- Description: This control provides for changing the LO injection side in the receiver. Changing the LO injection side mirrors the image receive band to the other side of the wanted channel. Use this feature if a particular band of interest is to be avoided on the image frequency.
- Unit: Self-explanatory enumerated list
- Applicability: Rx

2.10.5 Miscellaneous

TX PLL Bandwidth

- Description: This control allows for manual adjustment on the PLL synthesizer's bandwidth in Tx mode. By default the PLL bandwidth will be configured to value that can accommodate the whole modulation bandwidth of the Tx signal. Adjust the Tx PLL bandwidth only if better out of band emission performance is required. Note that reducing the PLL bandwidth may filter the wanted signal too.
- Unit: Enumerated list where bandwidth values are stated in kHz.
- Min value: 250
- Max value: 3000
- Applicability: Tx

RX PLL Bandwidth

- Description: This control allows for manual adjustment on the PLL synthesizer's bandwidth in Rx mode. By default the PLL bandwidth will be configured to 250 kHz in Rx mode. Adjust the Rx PLL bandwidth only if better selectivity performance is required.
- Unit: Enumerated list where bandwidth values are stated in kHz.
- Min value: 250
- Max value: 3000
- Applicability: Rx

RX Baudrate Offset

- Description: This control allows for offsetting the Rx nominal bitrate from the bitrate set in the Modem card. Practically this means offsetting the receive bitrate from the Tx bitrate.
- Unit: Hz
- Min value: 0
- Max value: 100 000
- Applicability: Rx

RSSI Update Period

- Description: RSSI Update Period adjusts the length of the measurement window and update interval in the following fashion: T_meas/update = 2^RSSI_Update_Period, where the unit is one symbol time. In other words the measured RSSI value will be averaged over the set time period.
- Unit: symbol time
- Min value: 0
- Max value: 15
- Applicability: Rx

Signal Quality Indicator Threshold

- Description: The demodulator generates two signal quality indicator numbers that are based on the measured correlation values on each symbol in the frame. The first is the average correlation value as measured on the first eight symbols in the frame (that is, after sync word). The second is the number of symbols in the frame with a correlation value lower than the threshold set in this entry. The signal quality indicator, referred to as link quality indicator (LQI), can be accessed with the RAIL API function `RAIL_GetRxPacketDetails()`.
- Unit: N/A
- Min value: 0
- Max value: 255
- Applicability: Rx

SRC Operation

- Description: SRC stands for Sample Rate Converter. SRC is available on IC revision xG12 and above. The purpose of the SRC is to adjust the sampling rate in the demodulator so that an integer oversampling rate (OSR) is achieved after channel filtering. Integer OSR helps symbol synchronization that in turn has a beneficial effect on sensitivity. By default the SRC is enabled.

  Disable the SRC if a revision xG1 compatible configuration is required and/or signal propagation minimization is required (i.e., for fast clear channel assessment by an RSSI measurement).
- Unit: Self-explanatory enumerated list
- Applicability: Rx

Enable FEC on Tx

- Description: This feature should be used with IEEE 802.15.4 SUN FSK PHYs if the transmitted packet requires FEC.
- Applicability: Tx

Enable dynamic FEC

- Description: This feature should be used with IEEE 802.15.4 SUN FSK PHYs. If this mode is enabled, two sync word is configured, indicating if FEC is used on packet. Must be used with the `RAIL_IEEE802154_ConfigGOptions(railHandle,RAIL_IEEE802154_G_OPTIONS_ALL, RAIL_IEEE802154_G_OPTION_DYNFEC)` API call.
- Applicability: Tx

Fast Detect Enable

- Description: This feature was introduced on the EFR32xG23 and is only supported by 2(G)FSK modulation. Enables fast preamble detection, which is necessary for channels scanning or low duty cycle (also referred as PSM, Preamble Sense Mode). To turn on this feature, additional RAIL Signal Qualifier (SQ) functions must be called.
- Applicability: Rx

ETSI Category 1 Compatibility

- Description: This entry is very specific to ETSI Rx category 1 compliancy in the 169 MHz band. It boosts blocking performance to meet the -20 dBm specification without the need of an external SAW filter.

  Only enable this if you are developing a narrowband (≤ 12.5 kHz operating channel) 169 MHz ETSI category 1 compliant receive application.
- Unit: Self-explanatory enumerated list
- Applicability: Rx

Target Oversampling Rate

- Description: This entry sets a target value on the oversampling rate (OSR). OSR is the number of signal samples in one symbol in the digital demodulator. You may want to increase this value for better DR offset tolerance and / or faster Rx chain propagation delay.
- Unit: ratio
- Min value: 3
- Max value: 8
- Applicability: Rx

OOK Slicer Level

- Description: This entry configures the slicing threshold in OOK demodulation. OOK demodulation is done through a PEAK detector that follows the instantaneous RSSI stream with a fast attack (fast charge) slow decay (slow discharge) algorithm. The slicing threshold is an RSSI level above which the signal is considered a digital one and below a digital 0. The slicing threshold is developed as the PEAK detector value minus the OOK slicer level which is adjustable in this entry. Decreasing this value improves sensitivity at the price of robustness. A higher value is more robust against fluctuations on the amplitude of the incoming signal, but introduces an artificial threshold above the noise level (determined by the channel filter bandwidth). A lower value means less signal power is needed above noise, to generate a logic 1, but noise itself also can hit the threshold, making the output also noisy. For EFR32xG12 to EFR32xG22 generations, this value is set dynamically during reception and should not be set manually. For EFRxg23 and on, this property is not active anymore, as the radio configurator for these models calculates this value automatically, and reliably.
- Unit: 2 dB
- Min value: 1
- Max value: 10
- Applicability: Rx

IR Cal Power Level

- Description: This parameter sets the PA power level raw code during image calibration when the external calibrating signal loopback is selected. This may be configured if a certain sub GHz PHY is operating above 462 MHz and the Rx and Tx paths are not connected together outside the chip. Such applications may include simple two antenna Rx/Tx designs, FEM (Front End Module)-based designs and front-end RF switch designs (that is, antenna diversity). Note that for this control to take effect SHARED_RX_TX_PATH must be selected in the Common Rx/Tx circuit drop-down menu (Antenna card in the Advanced group) even though the design is SPLIT_RX_TX_PATH. Note also that in the 2.4 GHz band no on-chip IR calibration can be run, so this control is not applicable there.
- Background: Above 462 MHz the IR calibrating signal is looped back to the Rx outside of the chip as it is routed through the PA. This is the only way to provide the neccessary SNR on the calibrating signal in this frequency band for the calibration to reach optimal rejection. If the Rx and Tx paths are not connected together (see control Common RX/TX circuit) outside the chip the above scheme cannot work, and the calibrating signal is looped back inside the chip as a fallback option. In this band, however, the internal loopback calibration performance is capped at ~ 40 dB rejection. While this is still an improvement on outlier parts, on average it is a degradation. This is the default operation when SPLIT_RX_TX_PATH is selected.

  This control is provided to overcome the limited performance on IR calibration by increasing the PA power level for the external loopback calibration to bridge the isolation gap between the unconnected Tx and Rx paths. The power level can be adjusted in units of power level raw codes where 0 is the minimum level and 248 (on 20 dBm parts) is the maximum level in the sub GHz space. As isolation between the Tx and Rx paths may differ greatly between designs, finding the right value is a manual process. The process must be done on the exact HW configuration to be implemented when the calibration is run (that is, antennae attached to antenna ports or antenna terminated with measuring instruments at final test).

  To find the correct power level, begin with the PA power level raw code 0 and keep increasing the code. Read back the calibration coefficients and also measure image rejection. After a certain code level both image rejection and calibration coefficients stabilize and remain stable until high power levels where the rejection performance may degrade again due to saturation in the Rx chain. Select a code on the plateau that gives sufficient margin (such as 5 PA power level codes) on the lower end. For information the minimum power level for optimal calibration on the Rx input is -54 dBm.

  Note that IR calibration is a once in a product lifetime calibration. It may be done in the field after the first power-up of the device. The above described method, however, is **strictly recommended to be done at manufacturing test** as the increased power level during calibration may violate regulatory standard limits.

  In summary the above described alternative IR calibration method shall be used if:
  - the design is operating above 462 MHz (and below 1000 MHz) and
  - the Rx and Tx paths are not connected outside of the chip and
  - ~ 40 dB image rejection is not acceptable in the product and
  - the calibration can be performed in-house at manufacturing test.
- Unit: PA power level code
- Min value: 0
- Max value: 248
- Applicability: Tx and Rx

Byte Position of Dynamic Length Byte

- Description: By default, the variable length byte location is the same as the last byte of the header. Enabling this field can be used to override that, and set it directly.
- Unit: Bytes
- Min value: 0
- Max value: 4096
- Applicability: Tx and Rx

Length of the First Word

- Description: On reception, create the first received bytes from less than 8 bits. This can be used to "bitshift" the frame. Upper bits are padded with 0 in the downloaded frame. Example: To drop the first 19 bits of the received frame, set this to 3 and ignore the first 3 bytes of the downloaded frame.
- Unit: bits
- Min value: 1
- Max value: 8
- Applicability: Tx/Rx

2.10.6 Antenna

About Antenna Diversity

An EFR32 can select between two antennas (referred to as ANT0 and ANT1) for packet transmission and for packet reception, to mitigate the effect of fading due to multipath propagation experienced particularly in indoor environments. Rx and Tx antenna selection are independent of each other. Tx selection is handled completely on the application side where a manual override for Rx selection is also possible.

Indoor fading typically causes 5-6 dB reduction in received signal strength. In practice this means that if one antenna falls into a “dead spot”, resulting in bad or no reception, switching to the other one can result in an improved link performance assuming correct antenna placement. Fading conditions vary over time due to moving objects and people, therefore a dynamic and agile antenna validation and selection feature is needed.

The Antenna Diversity feature facilitates two kinds of automatic methods for Rx antenna selection on top of the manual one, namely the Select-First-Good (SFG) and the Select-Best (SB) algorithms. Only SB takes into account signal quality for antenna selection, where signal/antenna quality is based on RSSI measurements.

When antenna diversity is enabled, the radio keeps toggling in Rx between the two antennas until timing detect occurs on one of them. Only the switching pattern and selection strategy differs for the two main modes, SFG and SB.

Note: The radio reverts back to SFG whenever the input signal level is perceived by the AGC block to be high enough for proper detection. This is typically between -65 to -80 dBm, well above sensitivity levels. When low signal levels are reached, antenna diversity will switch its operation mode back to the requested and preferred one automatically.

For deterministic operation, after every antenna switching not only the timing detection but the AGC block plus DC and frequency compensation are also restarted. Therefore a **proper antenna dwell time, and thus the preamble length, will be a main factor in using antenna diversity effectively**. The latter is critical in giving enough time to complete the detection and selection process before the sync word arrival, in every possible situation. Different modes require different amount of preamble time for their proper operation.

The required preamble length is dependent on:

- Length of the timing sequence used for Timing Detection
- AGC and detection block settle times at antenna switching
- Internal processing time
- RSSI measurement time

At any point during or after the selection process, when timing is lost, antenna switch will occur and the operation restarts, in other words the radio goes back to timing search.

The antenna dwell time is automatically determined by the configured timing detection scheme, AGC and RSSI measurement settings, and highly correlates with the timing window length.

Note: The internal base profile PHYs were created without antenna diversity in mind. Thus at least the timing detection scheme (number of symbols in timing window, number of timing windows to detect, timing samples threshold), preamble length, AGC settling delay, and RSSI measurement settings must be reviewed under the Advanced tab of the radio configurator when enabling antenna diversity. Since the driving event behind antenna diversity is timing detection, a robust yet fast timing detection scheme is required to save preamble time in general and to avoid false timing detects triggering unnecessary antenna selection process resulting in possible packet miss.

Note: Due to the short antenna dwell times Automatic Frequency Cancellation feature is not available, only the Internal Frequency Compensation can be used.

After timing detection is asserted on one antenna the algorithm can follow various paths based on the configuration in Antenna Diversity Mode.

Antenna Diversity Application Support

The various Silicon Labs software stacks and SDKs such as EmberZNet, Silicon Labs Thread, and Flex support antenna diversity on different levels and with different feature lists. For those implementations please consult the corresponding documentation regarding antenna diversity feature support, for example *AN1181: Configuring Antenna Diversity for EmberZNet*.

Custom and/or stackless implementations should rely on the RAIL API in using antenna diversity. We give a quick summary here, but for specifics please search the RAIL documentation available here: https://docs.silabs.com/.

To use antenna diversity with the RAIL API, first call `RAIL_ConfigAntenna()` to configure the GPIO pins that will drive the external RF switch.

The Rx behavior of the radio can be configured with `RAIL_ConfigRxOptions()` passing either `RAIL_RX_OPTION_ANTENNA0`, `RAIL_RX_OPTION_ANTENNA1` for static, manual antenna selection (meaning Rx diversity off); or `RAIL_RX_OPTION_ANTENNA_AUTO` for dynamic, automatic selection as an argument (meaning Rx diversity on).

The algorithm of Rx antenna diversity itself, when enabled, is governed by the configuration in the Radio Configurator's Antenna card (Advanced group).

The antenna selection for a Tx operation is handled at the Tx starting API functions like `RAIL_StartTx()` and similar by passing options `RAIL_TX_OPTION_ANTENNA0` or `RAIL_TX_OPTION_ANTENNA1`. If neither or both of these option is set at the function call then Tx takes place on the antenna defined/determined by the last Rx or Tx operation. So, by default (not passing an antenna-related Tx option) all Tx operation takes place on the antenna where the last packet was received by antenna diversity.

The Auto-ACK message is transmitted on the antenna where the ACK-ed frame in question was received by antenna diversity.

Antenna Diversity Mode

- Description: This entry selects the operation mode for Rx antenna diversity. EFR32 can select between two antennas (referred to as ANT0 and ANT1) for packet transmission and for packet reception as well, to mitigate the effect of fading due to multipath propagation experienced especially in indoor environments. Rx and Tx antenna selection are independent of each other. Tx selection is handled completely on the application side where a manual override for Rx selection is also possible.
- Units:
  - **Disable:** This option disables automatic Rx antenna selection. Manual selection is still possible.
  - **ANTSELFIRST:** Enables the SFG mode. In this mode the receiver is alternating between ANT0 and ANT1 during timing search while looking for a valid timing pattern on Rx signal. As soon as a valid timing has been detected, SFG selects the actually active antenna for the rest of the frame, or until timing is lost. If timing is lost, the antenna is switched, and the radio is restarted. So, as its name suggests, the antenna is selected on which a valid timing pattern is detected first, regardless of signal quality metrics and without any related measurements. Use this option only if the preamble budget does not allow for selecting an SB algorithm.
  - **ANTSELRSSI:** Enables SB mode with RSSI-based quality metric (SB-RSSI). In SB mode the receiver alternates between ANT0 and ANT1 during the timing search looking for a valid timing pattern on the Rx signal. When a valid timing pattern is found, antenna diversity tries to select the best antenna for receiving the rest of the frame. To achieve this, the signal quality for the currently active antenna is saved/updated at every subsequent antenna switch. Therefore at the first timing detect event the algorithm already has a fresh quality metric for one antenna. To be able to perform a valid comparison between ANT0 and ANT1, the radio switches simultaneously with the timing detect event to the other antenna to perform a signal quality evaluation/update there. Finally, antenna quality results get compared, and the algorithm selects the better antenna for packet reception. If the better antenna is the current antenna then the Rx operation carries on with packet reception without further antenna switching. If the better antenna is the other antenna then the radio switches onto that one, reacquires timing and carries on with packet reception on that antenna.
- Applicability: Rx, not available on xG1 family.

Diversity Select-Best Repeat

- Description: This option makes the Select-Best diversity algorithms always go back to the antenna on which timing detection occurred first (aka the first antenna) for a recheck on signal quality because the measurement might be degraded as a result of a partially occupied measurement window by the Rx signal. This feature is an extra insurance that the signal qualification measurement is correct on the first antenna before evaluation and final antenna selection.
- Units:
  - **REPEATFIRST**: Signal qualification measurement is repeated on the first antenna.
  - **NOREPEATFIRST:** Signal qualification measurement is not repeated on the first antenna.
- Applicability: Rx, not available on xG1 family.

The worst-case antenna diversity SB scenario:

1. Switch to antenna B, no preamble in the air, no timing detected.
2. Switch to antenna A, preamble arrives late, no timing detected.
3. Switch to antenna B, preamble is in the air, antenna signal is weak for timing detection.
4. Switch to antenna A, timing detected and antenna quality QA1 measured.
5. Switch to antenna B, antenna quality QB measured.
6. (if **REPEATFIRST**) Switch to antenna A, measure antenna quality QA2.

   (if **NOREPEATFIRST**) Go to 7.
7. (if **REPEATFIRST**) QA2<QB, switch to antenna B, otherwise stay on antenna A.

   (if **NOREPEATFIRST**) QB<QA1, switch to antenna A, otherwise stay on antenna B.
8. Start packet search (timing, preamble, sync) on the best antenna selected in 7.

This worst-case sequence can be used to find the required minimum length of preamble for the select best diversity mode.

Common RX/TX circuit

- Description: This option is completely independent from the antenna diversity feature, and only takes effect on the reference signal's path used during the I/Q mixer's calibration process to achieve good image rejection (IR) performance. In the subGHz frequency range of 462 MHz to 1000 MHz the IR (I/Q) calibration uses by default a test signal that is looped back externally, going through the PA module and the Tx and Rx pins/paths. Below 462 MHz the IR calibration applies by default an internal (PLL) loop back path, only to ensure good IR performance over the whole subGHz region.

  Note: The internal loop back calibration performance is worse in the frequency region above 500 MHz.

  External (PA) calibration path cannot be applied when the Tx / Rx paths are not connected together. (For a workaround see the **IR cal power level** parameter.) In such cases the IR calibration must apply the internal (PLL) loop back. This control makes it possible to manually override the default behavior according to the actual board layout.
- Unit: Self-explanatory enumerated list
- Applicability: Rx

2.11 Special Radio Profile-Related Properties

2.11.1 Long Range Profile

Long Range Mode

- Description: This enum sets the Long Range profiles, which establish different data rates. The following table contains the default configuration options on Long Range Profile and the recommended Crystal Oscillator (XO) accuracy.

Frequency Band (MHz)	Data Rate (kbps)	(TX + RX) XO Accuracy (ppm +/-)
434/490	1.2	2.5
434/490	2.4	5
434/490	4.8	10
434/490	9.6	20
434/490	19.2	40
868/915	1.2	1.25
868/915	4.8	5
868/915	9.6	10
868/915	19.2	20
868/915	38.4	40
868/915	80	40

  For more information on Long Range Profiles and performances, see UG460: EFR32 Series 1 Long Range Configuration Reference.
- Unit: Enumerated list
- Applicability: Tx/Rx

2.12 Mbus Profile

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 (for example, 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.
- *3of 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

Mbus postamble length

Configures the postamble length in sets of two alternating chips (i.e. either ‘01’ or ‘10’). This is required for modeS and modeT, and a maximum of 4 pairs (8 chips) can be transmitted.

Preamble length total

Total length of the preamble in bits. Since EN13757-4 only specifies the minimum preamble length for a number of modes (which is used by our preconfigured modes), this setting can be used to configure it for a longer preamble.

2.13 Wi-SUN FSK Profile

Wi-SUN Mode

- Description: Configures the Wi-SUN mode. This protocol uses different frequency and modes based on region. The following table shows the modes and the associated modulation index and data rate:

PHY Operating Modes	Symbol Rate (ksymb/s)	Modulation Index
1a	50	0.5
1b	50	1.0
2a	100	0.5
2b	100	1.0
3	150	0.5
4a	200	0.5
4b	200	1.0
5	300	0.5

- Unit: Self-explanatory enumerated list
- Applicability: Tx/Rx

3. Multi-PHY Configuration Example

Consider the following requirements:

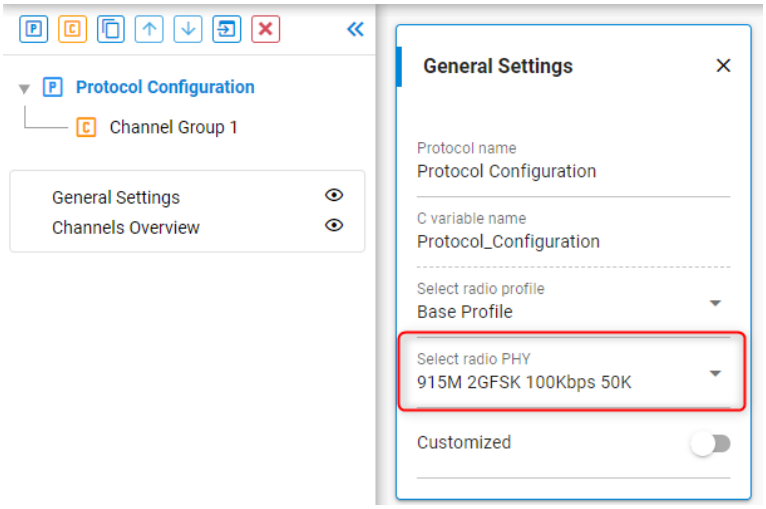
1. Five channels (channels 0-4) FSK with 300 kHz channel spacing, starting at 905 MHz, using the built-in 100 kbps 915 MHz PHY of the base profile as a base.
2. At 905.6 MHz (channel 2 in the above group), payload length should be 32 bytes (compared to the standard 16 on the other channels).
3. Add another channel (channel 5) with the same setup as the first group, but at 907 MHz, as a special channel distance.
4. On the 907 MHz channel it should be possible to receive frames with both 100 kbps and 500 kbps bitrates. Note: In order to implement this an additional channel definition (channel 6) is required.

To create a configuration covering these requirements, do the following in the Radio Configurator, where each section corresponds to a requirement above.

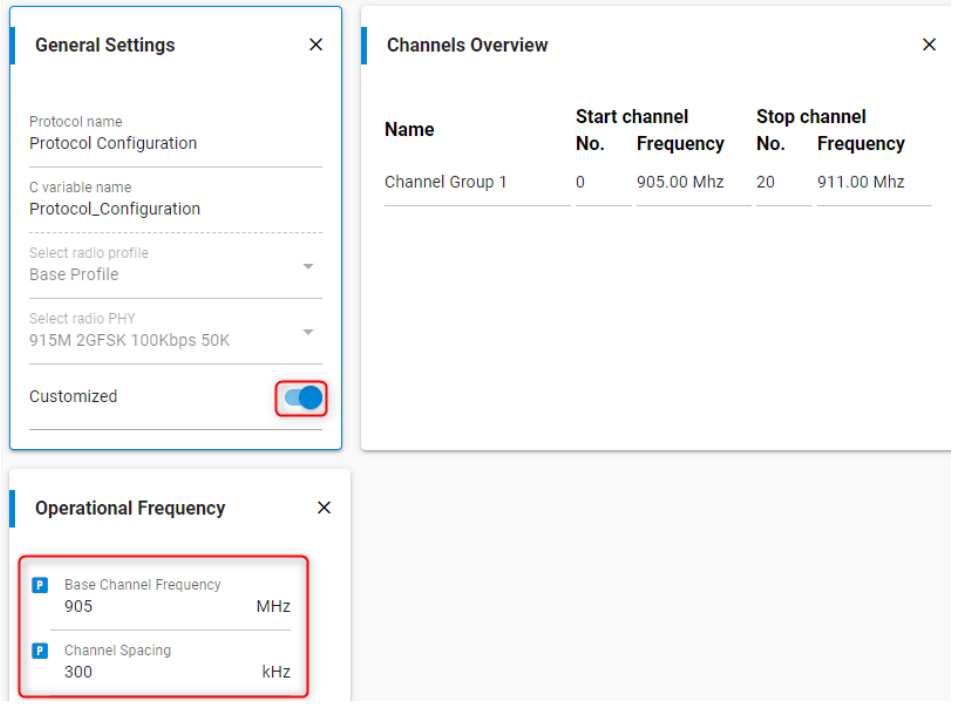
3.1 Configure the Base Profile

To set up the protocol according to the requirements in 1 above:

1. select the 915M 100 Kbps PHY as shown in the following figure:

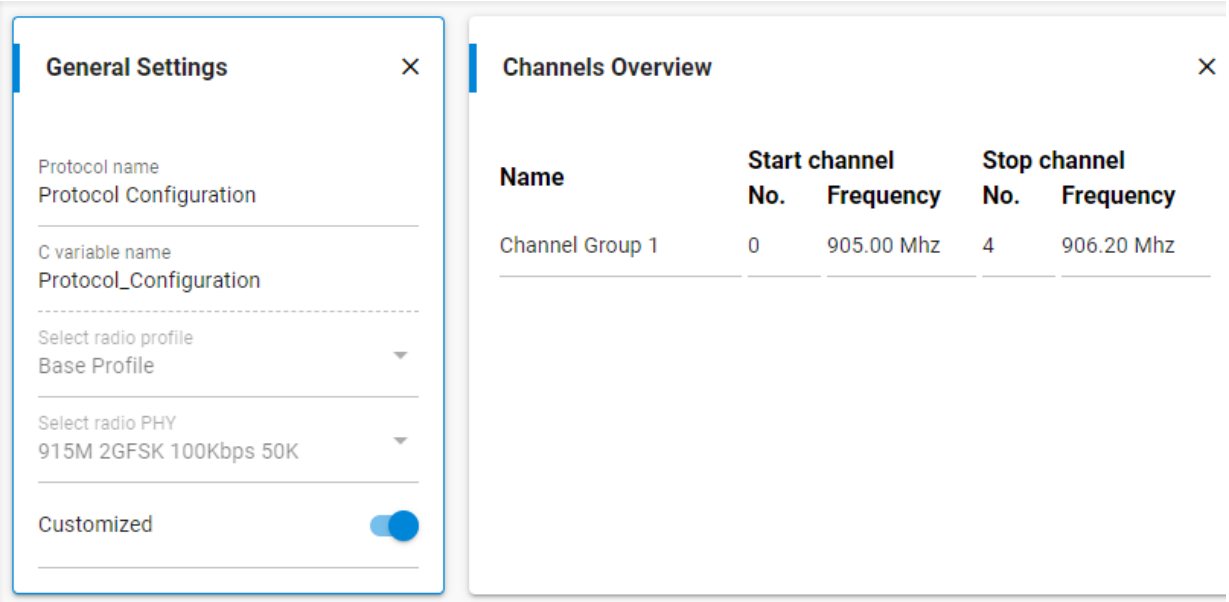


2. Enable customization and set up a 905 MHz base frequency with 300 kHz channel spacing:



Note the P icons next to the configured fields: This means the config deviates from the selected PHY by these settings.

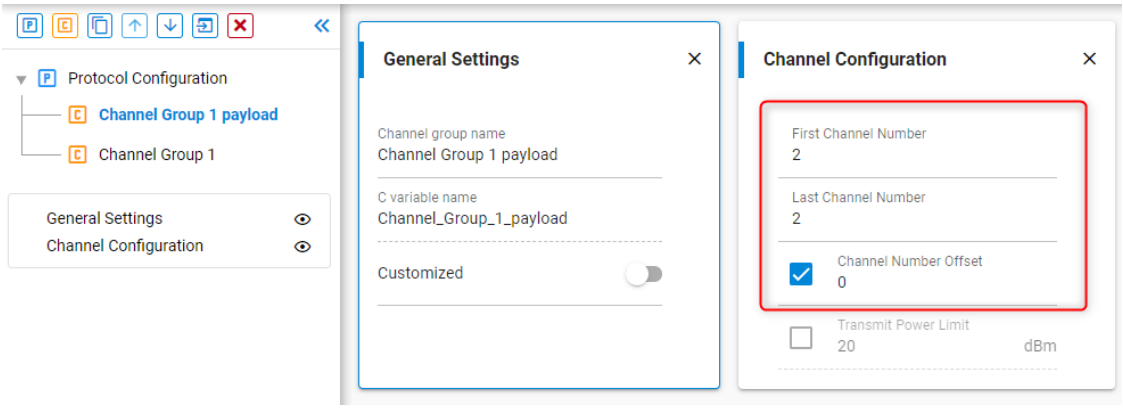
3. With this setup, by default we have one channel group with 21 channels from 0 to 20, so change the last channel number to 4 on the channels overview:



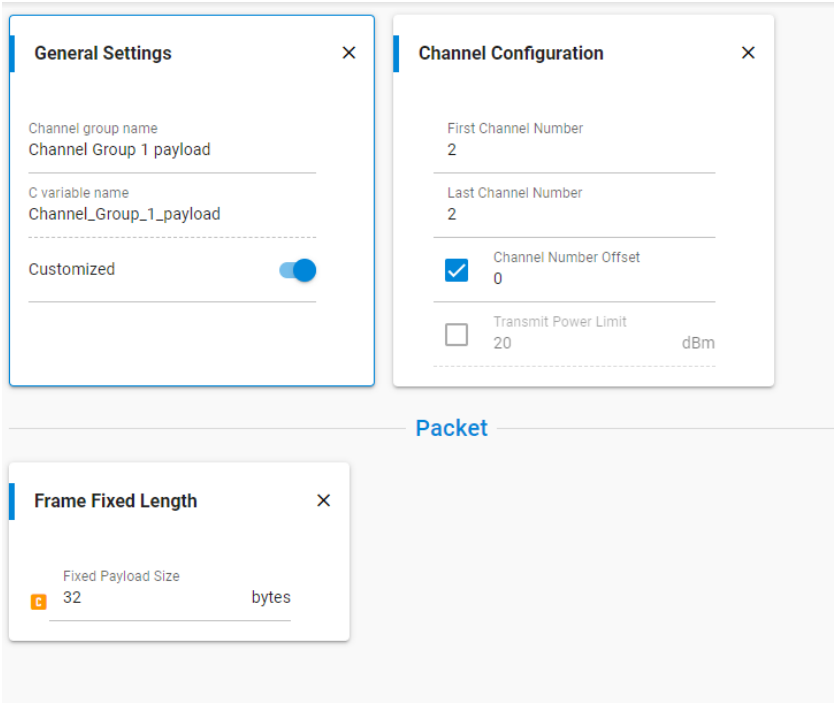
At this point, we have a single PHY config: Every radio parameter is configured on the protocol configuration, and the channel group only defines the channel numbers.

3.2 Change Payload Length

905.6 MHz is channel 2 from the previous step. RAIL allows defining the same channel in multiple channel groups, in which case it will always use the first channel group in which the given channel is defined. Therefore, if we define channel 2 again with the 32-byte length, and we make it the first channel group, it will override the previously configured channel 2. The easiest way to do this is to make a copy of the first channel group (click the **Copy** icon after selecting the source), move it to the first position (Up Arrow control), and set both the first and last channel number to 2, while setting Channel Number offset to 0:



Next, enable customization and configure the payload length:



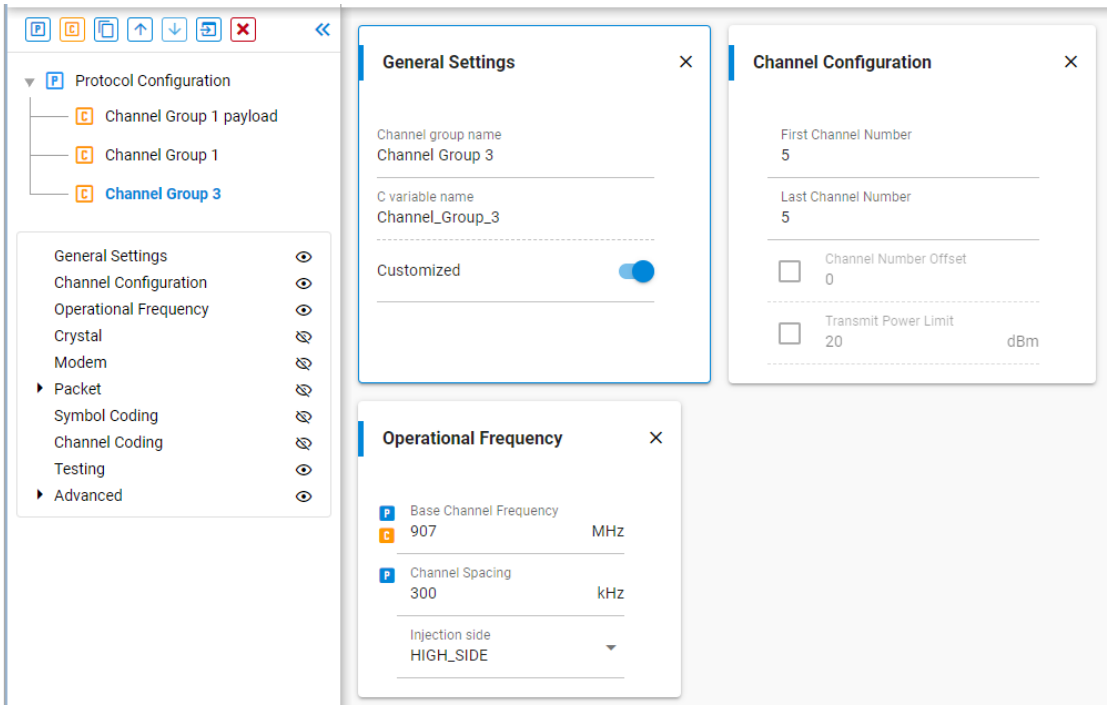
Note the C icon. It means the setting next to it is not inherited, it is configured on the channel group.

This generates two small configuration difference arrays (delta configs), which configure the payload size. RAIL will load one of these config deltas automatically when you switch between channels of different channel groups (see `RAIL_ChannelConfigEntry_t.phyConfigDeltaAdd` in the RAIL API documentation).

3.3 Add a 907 MHz Channel

907 MHz cannot be defined with the group created in the first step, as it does not follow the 300 kHz channel spacing. However, channel groups can differ in the properties that you enable as channel group-based properties.

Create a new channel group with a single channel (channel 5). Enable customization, and set the base frequency to 907 MHz:

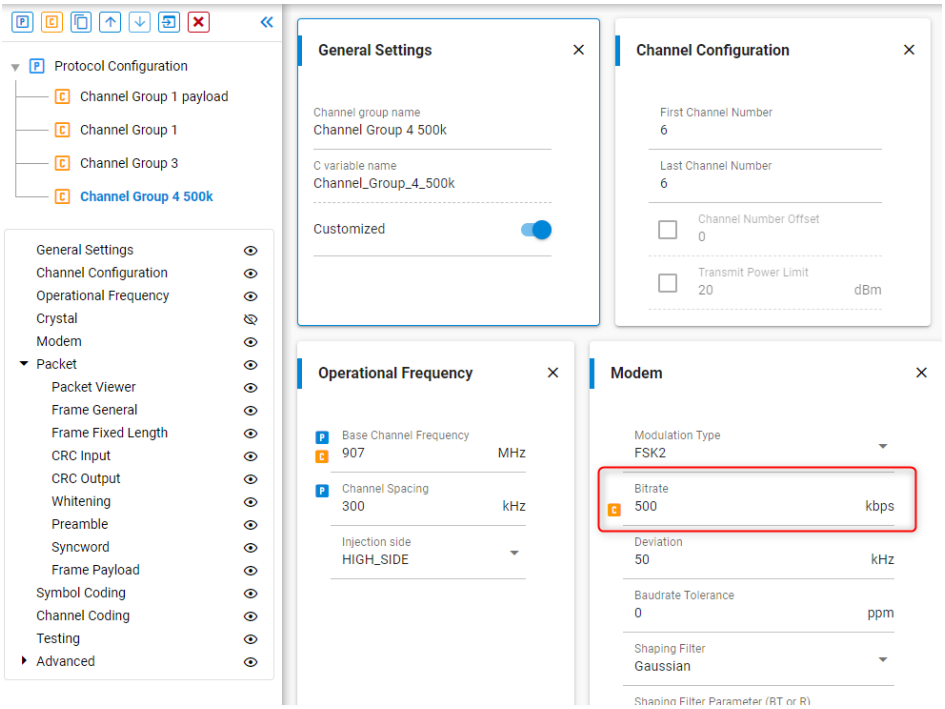


Note again the C and P icons. To summarize, the first means this channel group deviates from the protocol setting with this option. You can also see the P icon next to channel spacing, which means it inherits the value from the protocol. The settings without any icon are still inheriting their settings from the pre-configured 915 MHz 100 kbps PHY.

3.4 Create a New Channel

The bitrate of channel 5 is the default 100 kbps inherited from the protocol. It cannot be set to both 100 kpbs and 500 kbps, but a new channel 6 can be created on the same frequency, with a different bitrate using a new channel group.

Create and set up a channel group for channel 6 defining the same frequency as channel 5 has and modify the bitrate:



Note that this step is very similar to the process in section 3.2 Change Payload Length. Both processes customized channel groups, which will both generate the config deltas. The only difference is that, in section 3.2 Change Payload Length we created an override for an existing channel, while here we created a new channel.

RAIL will always load the same amount of configuration data when changing between channel groups. In this example, RAIL will reconfigure the bitrate even when changing between channel 0 and 2, which have the same bitrate.

If you go back to the protocol configuration, you can see the channels you set up on the channels overview:

Channels Overview

Name	Start channel		Stop channel	
	No.	Frequency	No.	Frequency
Channel Group 1 payload	2	905.60 Mhz	2	905.60 Mhz
Channel Group 1	0	905.00 Mhz	4	906.20 Mhz
Channel Group 3	5	907.00 Mhz	5	907.00 Mhz
Channel Group 4 500k	6	907.00 Mhz	6	907.00 Mhz

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

www.silabs.com