AN971: EFR32 Radio Configurator Guide

This document describes the radio configurator graphical user interface (GUI) for Proprietary applications. With the help of the GUI, users can create standard or custom radio configurations for their RAIL-based radio applications. This document explains the role of each GUI item in the configuration.

KEY POINTS

• Radio configuration GUI for Proprietary applications is described
• User can create custom radio configurations

1. Single-PHY Flow

Once an EFR32-based project that uses the Proprietary protocol has been generated in Simplicity Studio (as described in *QSG138: Getting Started with the Silicon Labs Flex SDK for the Wireless Gecko (EFR32™) Portfolio*) an .isc file is automatically created. The radio configuration GUI settings are stored in this .isc file. When the .isc file is opened, a GUI is displayed with several tabs. Select "Radio config" for the radio configuration.

The radio configuration happens in two hierarchical steps. First, the user selects a radio profile in the "Select radio profile" drop-down menu. A radio profile may be any radio link technology. These radio links can be bound by standards (for example the Bluetooth wireless protocol or IEEE 802.15.4), by specific Tx/Rx technologies (such as LDC RX operation or frequency hopping operation) 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 in the "Select a radio PHY for selected profile" drop-down list. Each profile has "built-in" configurations and one "Custom settings" selection. This latter selection allows access to all the parameters as defined by the profile; the "built-in" configurations cannot be changed.

Once the radio profile and radio PHY have been selected, users can review and modify the "Profile options" selection that is essentially the configurator GUI itself. If any of the "built-in" radio PHYs are selected, no changes are allowed; the configuration can only be reviewed. If a "custom settings" radio PHY is selected, all the displayed parameters can be configured.

Note:
1. If you select a "built-in" configuration, then go back to the "Custom settings" option. The GUI retains the "built-in" configuration and you can edit it.
2. The GUI values in the "Advanced" section only reflect the true configuration if the project has already been generated.

Based on the selected radio profile, the profile options section may be restricted. For example, if a radio profile is selected that is bound by a standard, the profile options section only allows users to set the base frequency. All other options are defined in the standard.

Only one PHY can be configured at any one given time from the GUI.

Upon clicking [**Generate**], the radio configurator updates the .isc file and creates the rail_config.c and rail_config.h files that contain the calculated radio PHY configuration, based on the current status of the GUI.

2. Multi-PHY Flow

As of version 2.4.0 of the Flex SDK, some example applications are shipped with the Multi-PHY configurator:

• Simple TRX Multi-PHY
• Wireless M-Bus (WMBus) meter
• Wireless M-Bus (WMBus) collector

In the future, all applications will use the Multi-PHY configurator, as it can also be used to define a single-PHY configuration.

The multi-PHY flow is slightly different from the single-PHY flow. Each config includes one or more protocol configurations and each protocol configuration includes one or more channel group configurations.

2.1 Protocols

Protocols can be switched using the `RAIL_ConfigChannels()` API, or can be used in Dynamic Multiprotocol applications. The protocol configuration is the same as the single-PHY configurator, except it does not have the Operational Frequency group.
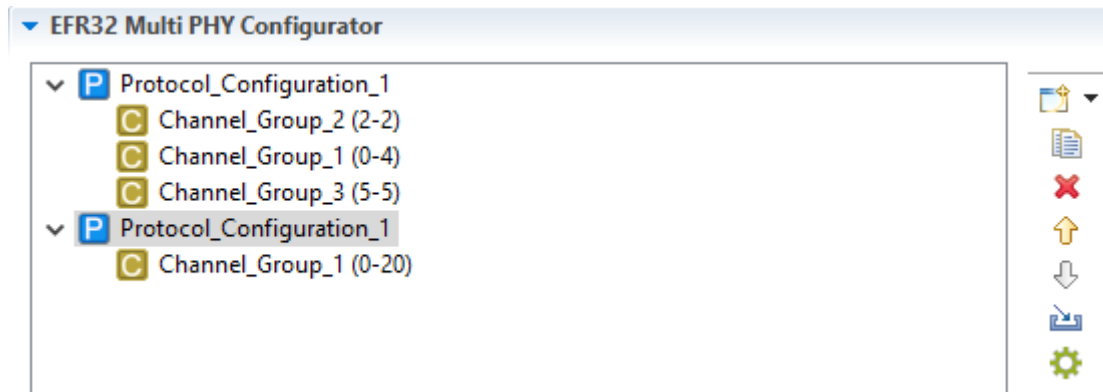
2.2 Channel Groups

Channel groups define one or more (sequential) channels, with a constant channel spacing between them. Channel groups can differ from each other in the radio configuration, and the differences can be configured through the Channel Group based properties icon

(. RAIL automatically detects when the channel change requires a channel group change as well. By default, the channel group configuration includes the Operation Frequency group, and a more flexible channel configuration.

The order of the channel groups will be the same in the `RAIL_ChannelConfigEntry_t` array as on the GUI. 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.

2.3 The Multi-PHY Interface

The Multi-PHY interface is at the top of the Radio Configuration tab.

The left pane displays the currently configured protocols and channel groups. After the channel groups, the channels configured in that group are shown in parentheses.

Icons that can be used to modify the list are on the left.

Create a new protocol or a new channel group.

Make a copy of a protocol or channel group.

Delete a protocol or channel group.

Move a protocol or channel group up.

Move a protocol or channel group down.

Import a legacy (single-PHY) .isc file, overwriting the currently selected protocol.

Configure channel group based properties.

For more details on Multi-PHY configuration, see the example in 4. Multi-PHY Configuration Example.

3. Configurator GUI

The input parameters on the GUI are arranged in subsections, each of which contains entries that logically go together. The entries in each GUI subsection are explained in detail in the following sections.

3.1 Operational Frequency

The operational frequency GUI subsection 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_TxStart and RAIL_RxStart 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

3.2 Crystal

The Crystal GUI subsection configures the frequency and the accuracy of the reference clock source (XO/ TCXO). Note crystal accuracy has two entries, one for Rx and one for Tx. Both of these numbers are needed to calculate the worst-case frequency offset between the nodes and to select a channel filter in the Rx side accordingly.

An on-chip capacitor bank presents the desired load to the crystal. The configuration of this capacitor bank is only available from the RAIL API through API functions `RAIL_GetTune` and `RAIL_SetTune`.

RX Crystal Accuracy in ppm
- Description: Sets the Rx node crystal accuracy
- Unit: ppm
- Min value: 0
- Max value: 200
- Applicability: Rx

TX Crystal Accuracy in ppm
- Description: Sets the Tx node crystal accuracy
- Unit: ppm
- Min value: 0
- Max value: 200
- Applicability: Tx

Crystal Frequency
- Description: Sets the crystal frequency
- Unit: MHz
- Min value: 38
- Max value: 40
- Applicability: Tx and Rx

3.3 Modem

The Modem GUI subsection 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.

Unit: Enumerated list:

FSK2: Frequency Shift Keying on two frequencies. Symbol shaping is also available on this modulation format. See the 'Shaping Filter' enry 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.

BPSK: Binary Phase Shift Keying: With BPSK the bit information is carried in the phase of the carrier. One symbol is represented by a phase shift of 0 (with regards to the carrier) while the other one is represented by a phase shift of 180 degree (with regards to the carrier). The implementation of BPSK also has amplitude modulation that brings down the output power when the phase rotations occur. The **shaping filter only affects the amplitude modulation** portion. Phase rotations will happen abruptly when the amplitude is ramped down.

DBPSK: Differential Binary Phase Shift Keying: With DPSK the phase rotations (0 or 180 degree) are measured with reference to the preceding symbol's phase as opposed to an absolute carrier phase. As with BPSK, amplitude modulation is also usedto bring down the output power when the phase rotations occur. **The shaping filter works in the amplitude domain.**

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.

ASK: Amplitude Shift Keying. ASK is an amplitude modulation where one symbol is represented by one power level while the other symbol is represented by another power level. 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: Offset Quadrature Phase Shift Keying: 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. In summary, 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 implemented without any amplitude modulation content, which would otherwise be present at 180 degree phase transitions. Half sine symbol shaping in the IQ amplitude domain is achieved by default without any shaping filter. Additional shaping in the frequency domain may be enabled to improve spectral properties of the signal if needed.

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 section 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.

Unit: Enumerated list

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.

Note: This field controls the polarity of Manchester coding / decoding if Manchester coding / decoding is enabled. In such a scenario, only MAP0 and MAP1 are allowed. (Manchester coding / decoding is not supported in more than one bit / symbol modulation formats such as OQPSK and 4FSK.)

Unit: Enumeration:

2FSK mode

Table 3.1. 2FSK Mapping Options

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

4FSK mode

Table 3.2. 4FSK Mapping Options

4FSK	-3dev	+dev	-dev	+3dev
MAP0	01	00	10	11
MAP1	11	10	00	11
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. 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

3.4 Packet Configuration

The Packet Configuration GUI section can be split in two parts. The first part is where the structure and the general properties can be configured on multiple tabs. The second part is a graphical representation of the selected packet structure whose elements can be further configured by clicking on them.

3.4.1 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.
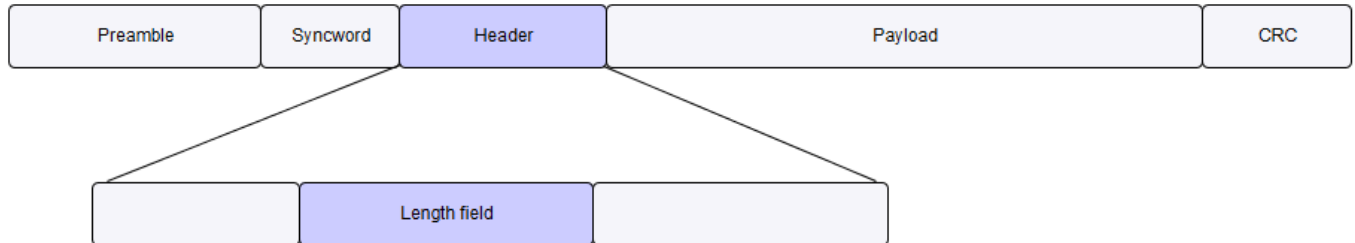
The configurator supports three types of payload length configurations:
- Fixed length
- Variable length
- Frame type

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.

Fixed Length

All packets has the same payload length, which is the configured by *Fixed Payload Size*.

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.

Variable Length

Each packet can have different payload length, defined by the Length field, described above.

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 header is only 1 byte, the decoder prefixes it with 0x00.
1. If Variable Length Byte Endian is 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.

Frame Type

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 physical length parameter. The code itself can be between 0-7, called Frame Type.

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.

3.4.2 Packet Structure and General Properties

This section describes the tabs related to general packet configuration.

FRAME_GENERAL Tab

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).
Unit:	Enumerated List
	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_ABORT. 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_ABORT. 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.
Unit:	Enumerated list
	FIXED_LENGTH: This is the simplest option whereby the frame length is hard-wired to a fix value. The exact length can be configured under tab FRAME_FIXED_LENGTH.
	VARIABLE_LENGTH: This is an option whereby the frame length information is available in the frame itself. This feature is especially valuable at the Rx side as frames with dynamically changing lengths can be received without any software intervention. When this option is selected the various parameters of the length information (most importantly the location in the frame) must be configured on tab FRAME_VAR_LENGTH.
	FRAME_TYPE: This is an option whereby the frame length information is available in the frame itself in a coded fashion. That is to say the frame information does not explicitly appear as a number but rather as a code that has to be further resolved to physical length parameter. When this option is selected the various parameters of the length information (like the code to length resolution amongst others) must be configured on tab FRAME_TYPE_LENGTH.
Applicability:	Tx and Rx

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

FRAME_FIXED_LENGTH Tab

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

FRAME_VARIABLE_LENGTH

This section configures the various properties of the variable length coding scheme (such as location length, endianness of the length word) if VARIABLE_LENGTH option is selected.

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: 4095

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: 4095

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: 12

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

FRAME_TYPE_LENGTH

This section configures the various properties of the variable length coding scheme if the FRAME_TYPE option is selected. At maximum 8 frame types can be configured. Frame type 0 is coded with a value of 0, type 1 with a value of 1 and this sequence continuous up until type 7 with a code value of 7. Each of these frame types can be individually enabled or disabled. Each can have a unique frame length parameter assigned to it.

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 reception (X = 0–7).
Applicability:	Tx and 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

CRC

This section contains all the CRC-related configuration options. 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 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

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 3.3. 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

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

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 3.4. Whitening

Enumerated Value	Polynomial	Comment
None	NA	Whitening is disabled
PN9	X9+X5+1	
PN9_BYTE	X9+X5+1	
BLE	X7+X4+1	BLE

Note: PN9 and PN9_BYTE both have the same polynomial. The difference in the operation is that in PN9_BYTE mode the output appears byte-wise reversed compared to PN9 operation. This mode of operation is only recommended for legacy system support where bit reversing is mandatory.

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

3.4.3 Packet Graphical UI

This part of the Packet Configuration GUI subsection offers a graphical representation of the selected packet structure. You can configure each block or element by clicking on it. This section describes each element.

Preamble

The preamble element 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 given preamble in this section. 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 than 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.

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

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 de-modulated 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.

Two sync words can be defined. In such a scenario the receiver is looking for both words simultaneously and asserts SYNC_DETECT if either is found. Which sync word has been detected is returned by the RAIL API function `RAILCb_RxPacketReceived`.

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.** To avoid confusion from this, the GUI displays the sync word in a binary string format as it appears in the air once the configuration has been generated. 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.** To avoid confusion from this, the GUI displays the sync word in a binary string format as it appears in the air once the project has been generated. 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.
Unit:	N/A
Min value:	0x0000 0000
Max value:	0xFFFF FFFF
Applicability:	Tx and Rx

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:	254 (16 for variable length configs)
Applicability:	Tx and Rx

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 in RAIL API structure `RAIL_TxData_t`.

Payload CRC Enable

Description:	If this checkbox is enabled the payload content is included in the CRC calculation.
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

3.5 Symbol Coding

The Symbol Coding GUI subsection 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).

Three kinds of symbol coding are supported by the device: 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.

Symbol Encoding

Description:	This drop-down entry selects the symbol coding method.
Unit:	Enumerated list

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 FSK Symbol Map entry (for example, 0→01 and 1→10). 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).

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 despreading. 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).

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

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 (that is, the inversion of odd-indexed bit values) is used only for OQPSK modulation. Inversion is used for all 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 3.5. 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

Note:
1. DSSS spreading factor and DSSS chipping code length must be equal.

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

Table 3.6. Example DSSS Configuration

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

Table 3.7. Example Chip Sequences

Chip Sequence #	Bits	Chip Sequence	Note
0	00	01001101	Base
1	01	11010100	Base right shifted by 4 bits
2	10	10110010	Base inverted
3	11	00101011	Base right shifted by 4 bits and inverted

3.6 Channel Coding

The Channel Coding GUI subsection 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.

FEC algorithm

Description: This drop down selection entry configures the FEC algorithm to be used.

Unit: Enumerated list

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 section 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 section 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

3.7 Testing

The Testing GUI subsection 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.

Applicability: Rx

3.8 Advanced

The Advanced GUI subsection contains advanced controls that can be used for fine tuning. Most apply to the demodulator. **These controls take on their initial values once the configuration has been generated.** 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. **For the change(s) to take effect a new generation is needed. The settings in this subsection are meant for advanced users only.**

3.8.1 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 at the PREAMBLE section. 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.
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.
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

3.8.2 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 theswitching 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.
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:	on 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 category1 compliant narrowband applications.

Applicability   Rx

3.8.3 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 at this subsection 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:	1
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.

Unit: Enumerated list

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_ALLWAYS_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

3.8.4 Channel Bandwidth

Channel Bandwidth

Description:	Although the channel 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.
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

3.8.5 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 section. 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 `RAILCb_RxPacketReceived`.
Unit:	N/A
Min value:	0
Max value:	255
Applicability:	Rx

Disable SRC

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

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.

Unit: 2 dB

Min value: 1

Max value: 10

Applicability: Rx

3.8.6 Antenna

Antenna Diversity Mode

Description: This entry selects the operation mode for Rx antenna diversity. The purpose of Rx antenna diversity is selecting one of two antennas with the better signal quality for packet reception. When Rx antenna diversity is enabled the radio keeps toggling between the two antennas until timing detect occurs on one of them. The dwell time on each antenna is automatically determined by the configured timing detection scheme. After timing detection is asserted on one antenna the algorithm can follow various paths based on the configuration (see below).

Unit: Enumerated list

Disable: This option disables Rx antenna diversity.

ANTSELFIRST: The antenna will be selected for packet reception where timing detection occurs (aka the first antenna). There is no further signal quality evaluation between the two antennas. Use this option only if the preamble budget does not allow for selecting the best algorithm.

ANTSELCORR: In this mode the signal quality measurement is based on correlation. After timing detection the algorithm immediately switches to the other antenna and takes a correlation measurement (by acquiring timing via timing detection). The two measurements are compared and the algorithm selects the better antenna for packet reception. If the better antenna is the current antenna the Rx carries on; if the better antenna is the other antenna the Rx switches onto that one, reacquires timing and carries on with packet reception.

ANTSELRSSI: In this mode the signal quality measurement is based on RSSI. The algorithm is similar to ANSELCORR.

Applicability: Rx, not available on xG1 revision

Diversity Select-Best repeat

Description: This feature always makes the antenna diversity algorithm go back to the antenna on which timing detection occurred (aka the first antenna) for a recheck on signal quality. This feature is an extra insurance that the signal qualification measurement is correct on the first antenna.

Unit: Enumerated list

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 revision

Common RX/TX circuit

Description: This control will be checked if the Rx and Tx matching networks are connected at a common point. This configuration is also referred to as a direct tie configuration. In the subG frequency range of 500 MHz to 1000 MHz the IR calibration uses a test signal that is looped back externally from the Tx to Rx. This option cannot be applied if the Tx / Rx paths are not connected together. In such cases the IR calibration applies an internal loop back. The internal loop back calibration performance is worse in this frequency region.

Unit: Self-explanatory enumerated list

Applicability: Rx

3.9 Multi-PHY Configurator Only Options

Protocol name (Protocol)

Description:	This entry configures the name of the RAIL_ChannelConfig_t struct for the protocol.
Unit:	String
Restrictions:	Same as C variable restrictions (0-9, a-z, A-Z and underscore only, can't start with a number).

Channel group name (Channel Group)

Description:	This entry 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.
Unit:	String
Restrictions:	Same as C variable restrictions (0-9, a-z, A-Z and underscore only, can't start with a number).

First channel number (Channel Group)

Description:	This entry 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 (Channel Group)

Description:	This entry 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 (Channel Group)

Description:	This entry sets the channel number offset in the channel group. The frequency of any channel will be Base_frequency + (Requested 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.
Min value:	0
Max value:	65535

Transmit Power Limit (Channel Group)

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. PA conversion should be set up to use this, see *AN1127: Power Amplifier Power Conversion Functions in RAIL 2.x* for details.
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
Applicability	Tx

4. Multi-PHY Configuration Example

Consider the following requirements:

1. A 5-channel (channels 0-4) FSK with 100 kHz channel spacing, starting at 905 MHz, using the built-in 100 kbps 915 MHz protocol as a base.

2. At 905.2 MHz (channel 2 in the above group), the transmit power is limited at 10 dBm.

3. Another channel (channel 5) with the same setup as the first, but at 907 MHz.

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) will be required.

To create a configuration covering these requirements, do the following in the Multi-PHY configurator, where each step corresponds to a requirement above:

1. The 5-channel FSK can be set up using the same concepts as a single-PHY config. Set up the protocol according to the requirements, then set up the channel plan as follows:

2. 905.2 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. Therefore, if channel 2 is defined again with the 10 dBm limit, and made 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 and set both the first and last channel number to 2 (leaving the offset at 0):

3. 907 MHz cannot be defined with the group created in the first step, as it does not follow the 100 kHz channel spacing. To meet this requirement, create a new channel group with a single channel (channel 5) at 907 MHz:

Channel Group General Settings

Channel Group name | Channel_Group_2

Operational Frequency

Base frequency | 907 | MHz   Channel Spacing | 0 | kHz

Channel Configuration

First Channel Number | 5   Channel Number Offset | 5

Last Channel Number | 5   ☐ Transmit Power Limit | 20 | dBm

4. The bitrate of channel 5 is the default 100 kbps inherited from the protocol. It cannot be changed, but a new channel 6 can be created on the same frequency, with a different bitrate using a new channel group. Various channel group-based properties are available, but in this case enable a single one, the bitrate, and set up a new channel group for it:

Channel Group General Settings

Channel Group name | Channel_Group_2_500k

Operational Frequency

Base frequency | 907 | MHz   Channel Spacing | 0 | kHz

Channel Configuration

First Channel Number | 6   Channel Number Offset | 6

Last Channel Number | 6   ☐ Transmit Power Limit | 20 | dBm

Modem

Bitrate | 500 | kbps

This generates two small configuration difference arrays (delta configs), which configure the bitrate. RAIL will load this automatically when you switch between channel 6 and the others (see `RAIL_ChannelConfigEntry_t.phyConfigDeltaAdd` in the RAIL API documentation).

RAIL will always load the same amount of 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 configuration.
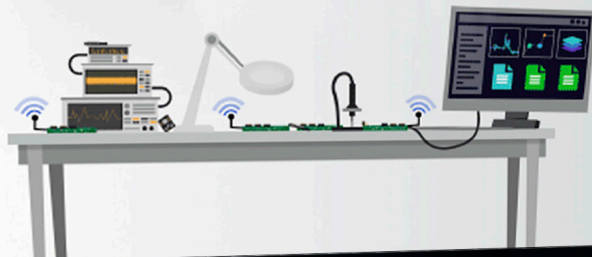
Simplicity Studio

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

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

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

http://www.silabs.com