# AN971: EFR32 Radio Configurator Guide

This document describes the radio configurator graphical user interface (GUI) for RAIL framework applications. With the help of the GUI, users can create standard or custom radio configurations to run their RAIL based radio applications. This document goes through the GUI items one by one explaining their role in the configuration.

KEY POINTS

- Radio configuration GUI for RAIL framework applications is described
- User can create custom radio configurations

# 1. Flow

The radio configuration happens in two hierarchical step. First the user has to select a radio profile. A radio profile could be any radio link technology. These radio links can be either bound by standards (i.e., BLE, IEEE 802.15.4) or specific Tx/Rx technologies (i.e., LDC RX operation, frequency hopping operation) or fully customarily.

Once the radio profile has been selected users can fill in the profile options section that is essentially the configurator GUI itself. Based on the selected radio profile, the profile options section may be restricted. As an example, if a radio profile is selected that is bound by a standard (like BLE) the profile options section will only allow users to set the base frequency as all the rest is defined in the standard.

A radio profile combined with the configured profile options results in a physical layer configuration, also referred to as a PHY. Most of the parameters of a PHY can only be configured from the GUI, however there are some parameters (i.e., frequency) that can also be configured by calling a RAIL function.

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

Upon hitting the Generate button the radio configurator calculator creates a PHY based on the current status of the GUI.

# 2. Configurator GUI

The input parameters on the GUI are arranged is sub sections each of which contains entries that logically go together. Each of these sub sections are explained in detail in the following sub chapters.

## 2.1 Operational frequency

This section contains entries for setting the operational frequency also referred to as the base channel frequency and the channel spacing values. The channel spacing value is used for relative frequency configuration where one can configure a frequency so many channel spacing away from the base channel frequency. The channel number can be passed along 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

## 2.2 Crystal

This section takes the frequency and the accuracy of the reference clock source (XO/ TCXO) that is used. Note that there are two entries for crystal accuracy: one for Rx and one for Tx. Both of these numbers are needed to calculate the worst case frequency offset between the nodes and select a channel filter in the Rx side accordingly.

There is an on-chip capacitor bank that presents the desired load to the crystal. The configuration of this capacitor bank is only available from the RAIL API, look for 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: Rx

**Crystal Frequency**

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

### 2.3 Modem

This section 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.

Unit: Enumerated list:

**FSK2**: Frequency Shift Keying on two frequencies. Symbol shaping is also available on this modulation format. See the 'Shaping Filter' 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: At 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 does also have amplitude modulation that brings down the output power when the phase rotations occur. The **shaping filter only has an effect on the amplitude modulation** portion, phase rotations will happen abruptly when the amplitude is ramped down.

**DBPSK**: Differential Binary Phase Shift Keying: At DPSK the phase rotations (0 or 180 degree) are measured referred to the preceding symbol's phase as opposed to an absolute carrier phase. As with BPSK amplitude modulation is also utilized to 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 presented by the absence of an RF carrier. Symbol shaping as selected by the shaping filter entry will get 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 will get 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. Without going into much detail 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 therefore can be and is indeed implemented without any amplitude modulation content (which would otherwise be there 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 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 will get 3 parity bits appended to it the Bitrate entry must be calculated as DR_net* 7/4.

Whenever DSSS or MANCHESTER symbol coding is applied each bit will be replaced by a longer sequence of chips in the modulator. It is important that in such cases the Bitrate entry does indeed expect the bit rate as opposed to the chip rate. Chip rate up (down) conversion will be 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 into this entry 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 regards 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<br><br>**None**: No symbol shaping filter is applied. Practical use cases of no shaping could be obtaining true FSK signals in 2FSK and 4FSK modulation formats and also obtaining half sine IQ shaped OQPSK signals. At all the rest of the cases some symbol shaping is recommended.<br><br>**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.<br><br>**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.<br><br>**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.<br><br>**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 will not have any 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. At 2FSK it is simply stating which frequency carries which bit; 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 is implemented on the chip. This configuration is shared between the modulator and demodulator.

Unit: Enumeration:

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 | 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

## 2.4 Packet Configuration

The packet configuration user interface can be split in two sections. The first one is an area where the structure and the general properties can be configured on multiple tabs whereas the second one is a graphical representation of the selected packet structure whose elements can be further configured by clicking on them.

### 2.4.1 Packet Structure and General Properties

This section goes through the various tabs related to general packet configuration.

**FRAME_GENERAL**

**Header Enable**

Description: This control enables or disables the HEADER field in the packet.

Applicability: Tx and Rx

**UART Mode**

Description: This control enables or disables the UART mode whereby start (0) and stop (1) bits are inserted before and after each word to be transmitted, respectively. Note, that these additional UART bits are added after CRC calculation at the Tx side and are removed before CRC calculation at the Rx side. Whitening and FEC (if enabled) however will be done on the additional UART bits too. This feature comes handy for emulating direct UART data transfer over the air.

Applicability: Tx and Rx

**Frame Length Algorithm**

Description: This control sets how the frame length information is determined. Note, that the frame does not include the preamble and sync word 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: Rx

**Frame Bit Endian**

Description: This control sets the bit endianness in the whole frame. Don't forget that this has not got an effect on the preamble and sync word sections. The enumerated options are self-evident.

Applicability: Tx and Rx

**FRAME_FIXED_LENGTH**

**FIXED_PAYLOAD_SIZE**

Description: This controls sets the frame length in bytes when FIXED_LENGTH option is selected. This control is only applicable to the Rx side. At the Tx side this information is passed down in RAIL function RAIL_TxDataLoad.

Unit: byte

Min value: 0

Max value: 255 (including the header field and all appended information)

Applicability: Rx

### FRAME_VAR_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. **All these configurations are only applicable in Rx mode.**

#### Variable Length Bit Endian

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

#### Variable Length Byte Endian

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

#### Length Includes CRC Bytes

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

#### Maximum Length

| | |
|---|---|
| Description: | This control decides whether the indicated length includes or excludes CRC bytes. |
| unit: | byte |
| Min value: | 0 |
| Max value: | 4095 |
| Applicability: | Rx |

#### Minimum Length

| | |
|---|---|
| Description: | If the decoded length in bytes is smaller than this value the packet will be discarded. |
| unit: | byte |
| Min value: | 0 |
| Max value: | 4095 |
| Applicability: | 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: | Rx |

#### Variable Length Bit Location

| | |
|---|---|
| Description: | This control sets a “pointer” to the LSB bit of the variable length word. As an example if the length word is 12 bits long the LSB bit may not fall to an exact boundary of one byte. With this parameter the start of the length word (more precisely the LSB bit of the length word) can be exactly located within a byte. |
| unit: | bit |
| Min value: | 0 |
| Max value: | 7 |
| Applicability: | Rx |

### FRAME_TYPE_LENGTH

This section configures the various properties of the variable length coding scheme if FRAME_TYPE option is selected. All these configurations are only applicable in Rx mode. 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 of them 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: | 0 |
| Max value: | 240 |
| Applicability: | Rx |

#### Frame TypeX Valid

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

A further 3 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: | 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: | 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: | 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 CRC value in the Tx frame reception is deemed successful.

#### TX CRC Bit Endian

Description: This control sets the bit endianness of the CRC in the Tx side. The enumerated options are self-evident. On a symmetric link this configuration shall match the Rx's.

Applicability: Rx

#### RX CRC Bit Endian

Description: This control sets the bit endianness of the CRC in the Rx side. The enumerated options are self-evident. On a symmetric link this configuration shall match the Tx's.

Applicability: Rx

#### CRC Byte Endian

Description: This control sets the byte endianness of the CRC both at 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 both at the Tx and Rx sides.

Applicability: Tx and Rx

#### CRC Input Padding

Description: In some standards it is required that the CRC value 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. I.e., if 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 | BLE |
| 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 chip 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 will be “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 2.4. 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

### 2.4.2 Packet Graphical UI

In this section of the GUI the different building blocks of the radio packet can be configured. The following sub chapters give a brief explanation of each of these building blocks.

#### Preamble

The preamble section describes how the transmitted preamble looks like 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.<br><br>When DSSS symbol coding is enabled the preamble base pattern defaults to DSSS chipping code base and this entry becomes irrelevant.<br><br>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 will directly get transmitted.<br><br>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. As an 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. If we stay at the previous example where 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 length 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 will be transmitted and neither will it be detected at the Rx side. |
| Unit: | bit |
| Min value: | 0 |
| Max value: | 2^21 – 1 = 2097151 (If set to maximum preamble transmission becomes continuous.) |
| Applicability: | Tx and Rx |

### Sync Word

This is the section where users can 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 too as radio packets with different (than expected) sync words will get dropped automatically.

It is possible to define two sync words. In such a scenario the receiver is looking for both simultaneously and asserts SYNC_DETECT if either is found. Which sync word has been detected is returned by RAIL API function **RAILCb_RxPacketReceived**.

When DSSS symbol coding is applied, sync word is detected on the already decoded bit stream; Whereas when Manchester coding is enabled, 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: | 1 |
| 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 MSB bits are used. Note that when DSSS symbol coding is enabled, the sync word will also get coded at the Tx side and will get 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 MSB bits are used. Note that when DSSS symbol coding is enabled, the sync word will also get coded at the Tx side and will get 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 |

### Header

The header provides for a logically separated field from the payload data with possibly different configurations. That is, it is possible to exclude the header field from CRC calculation and whitening. As a most common use case the length of the payload may be placed in the HEADER field when variable length frame detection is enabled although this is not a requirement. Note that from within RAIL there is no separate function to write the HEADER content, it is regarded as part of the payload there. Refer to RAIL API structure RAIL_TxData_t for more details.

#### CRC Header

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

**Whiten Header**

| | |
|---|---|
| Description: | If this checkbox is enabled the header content will be 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: | 0 |
| Max value: | 254 |
| 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 will be included in the CRC calculation. |
| Applicability: | Tx and Rx |

**Payload Whitening Enable**

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

### 2.5 Symbol Coding

This section contains controls that are related to symbol coding. Symbol coding is the concept that translates bits (or groups of bits) to be transmitted to modulation symbols that will represent the bit (or group of bit) in the modulation domain (frequency, phase, amplitude).

**Three kinds of symbol coding are supported on the chip: 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)**: Practically this option means that no symbol coding will be done, the incoming bits are just simply passed through to the modulator.

**Manchester**: Manchester coding replaces each bit with two chips based on the selection in the Manchester Coding Mapping entry (i.e., 0-> 01 and 1 ->10). At the Rx side Manchester decoding is done in a similar fashion whereby each pair of chips are 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 will ensure 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. The chip rate will be twice as much as the bit rate which means that the signal bandwidth is doubled in the air. Also note that **Manchester coding does not apply to the preamble and sync word sections**.

**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 (This operation is also referred to as spreading). DSSS greatly impacts the spectral properties of the transmitted signal. The chip rate in the air can be way 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 week 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. Practically 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

#### Manchester Code Mapping

Description: This drop-down entry selects the Manchester coding mapping (or polarity).

Unit: Enumerated list

**Default**: Bits are replaced in the following fashion: 0->01 and 1->10.

**Inverted**: Bits are replaced in the following fashion: 0->10 and 1->01.

Applicability: Tx and Rx

#### Differential Encoding Mode

Description: There is a built in feature in the modem that 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 (i.e, 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 from with binary cyclic right shifting and/or inversion or complex conjugation. Complex conjugation (i.e., 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 auto correlation (i.e., the base code and its cyclic shifted versions has 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 essentially the chip rate to bit rate ratio.

Unit: N/A

Valid values: 2, 4-32

Applicability: Tx and Rx

**Table 2.5. 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 2.6. Example DSSS Configuration**

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

**Table 2.7. 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 |

### 2.6 Channel Coding

This section contains controls that are 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 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).

Applicability: Tx and Rx

### 2.7 Advanced

This section contains advanced controls that could be used for fine tuning mostly the demodulator if needed. These controls take on their initial values once a generation has been run. If a value is to be tuned (i.e., stepped a few notches away from its initial value) its corresponding control must be enabled by activating the tick box next to it. For the change(s) to take effect in the radio configuration a new generation shall be run. This section is meant for advanced users only.

### 2.7.1 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 “hit” 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 measured to be a desired signal it issues TIMING_DETECT. A signal measures 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 less errors than a given threshold. At TIMING_DETECT the symbol sampling instant is also adjusted to compensate for baud rate offsets.

After TIMING_DETECT detection on a few more timing windows must also yield a positive answer for asserting 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 PREAMBE_DETECT has been asserted search for SYNC WORD begins.

Once TIMING_DETECT has been asserted the timing algorithm will still be running to continuously 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 is 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 will still keep on measuring and adjusting 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 how many consecutive timing window 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 the result of the correlation measurement is checked against. If the measured correlation is less than this threshold TIMING_DETECT will not get 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 will not be asserted unless all samples taken within the observation window are above a certain threshold level. With amplitude modulation formats (OOK and ASK) TIMING_DETECT will not be asserted unless the measured strong /weak signal periods match the baud rate period. This GUI entry sets a relative signal strength level for above purposes. |
| Unit: | N/A |
| Min value: | 0 |
| Max value: | 100 |
| Applicability: | Rx |

### 2.7.2 AGC

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

The AGC consist of two control loops. One is the RF front end control loop (also referred to as the FAST LOOP) the other one 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 already indicates the FAST LOOP is way faster than the channel filter control loop as the latter one includes the propagation delay of the channel filter itself. This delay is inversely proportional to the bandwidth. You can think of the AGC operation 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 right at the (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 will drive the gain configuration so that this target value will be 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 shall 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. AGC period adjusts the length of the measurement window. We have the usual trade-off here: the longer the measurement window the more accurate the measurement will be, this however comes at a price of a longer preamble requirement at the Tx side. AGC period adjusts the measurement window in the following fashion: T_meas = 2^AGC_PERIOD, where the unit is typically one baud. |
| Unit: | typically one baud 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 |

### 2.7.3 AFC

Slightly differing crystal frequencies will cause frequency offsets between Tx and Rx nodes at the carrier frequency. In the receiver there are mechanism that can compensate or cancel this frequency offset. Offset compensation happens inside the demodulator whereby the measured frequency offset is taken into account at developing the slicing thresholds 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, only one can be enabled at any one time.

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 whole 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 will also degrade 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 will not be any risk of losing frequency alignment to subsequent less accurate offset measurements. A prime example would be 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. With AFC, however there is also a feature which only makes the algorithm start at PREAMBLE_DETECT. This has the benefit that the AFC algorithm will not “wander off” too far on noise before a valid packet arrives. A downside of this approach though is that the range of the AFC will be smaller as preamble must be detected without any offset compensation.

The controls at this section allows 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 will be elementary window * 2^Frequency_Offset_Period.<br><br>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.<br><br>For AFC operation at least 40 bits of preamble is required. |

Unit: Enumerated list

**Disabled**: This option disabled both the AFC and interval compensation mechanisms that may come handy 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 (so long, that frequency drift could potentially occur).

Applicability: Rx

**AFC Frequency Limit**

Description: The control puts a limit on the absolute measured frequency error beyond which no compensation will occur. This is useful for stopping the radio from receiving packets above certain absolute frequency offsets (i.e., packets in the adjacent channel).
**Note:** Setting this value to 0 will disable 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

### 2.7.4 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 will get selected. |
| Unit: | kHz |
| Min value: | 0.1 |
| Max value: | 3400 |
| 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 a specific frequency must be eluded for the image. Note that there are restrictions on the IF frequency so the closest available will get selected. |
| Unit: | kHz |
| Min value: | 150 |
| Max value: | 1900 |
| Applicability: | Rx |

### 2.7.5 Miscellaneous

**PLL Bandwidth**

| | |
|---|---|
| Description: | This control allows for manual adjustment on the PLL synthesizer's bandwidth. This parameter typically plays a role at the Tx side when frequency/phase modulation is used. As in such cases modulation is done through the synthesizer its bandwidth must be scaled with the bandwidth of the modulating signal. |
| Unit: | Enumerated list where bandwidth values are stated in kHz. |
| Min value: | 250 |
| Max value: | 3000 |
| Applicability: | Tx |

**RX Baudrate Offset**

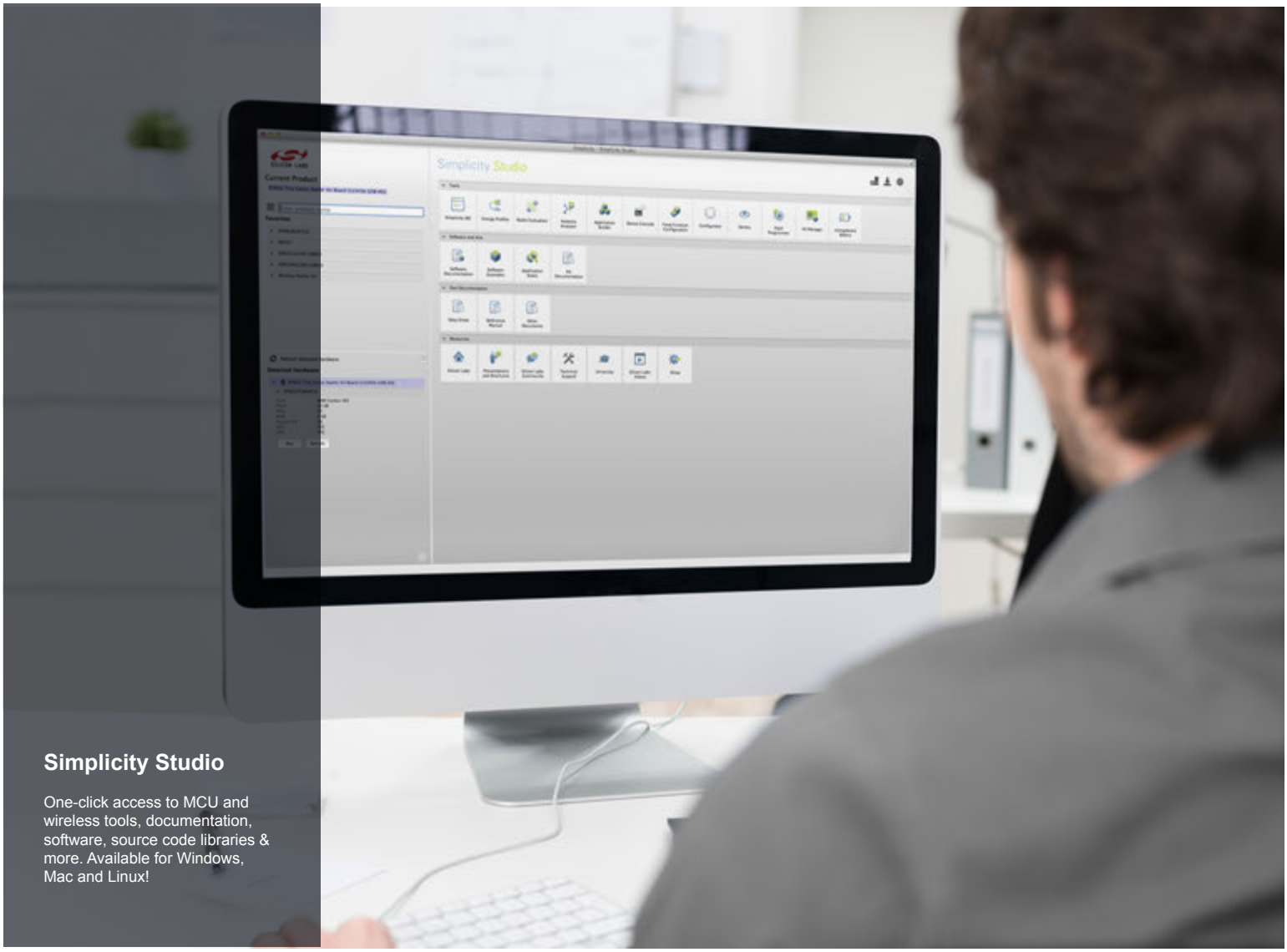
| | |
|---|---|
| Description: | This control allows for offsetting the Rx nominal bitrate from the bitrate set in the MODEM section. Practically this means offsetting the receive bitrate from the Tx bitrate. |
| Unit: | Hz |
| Min value: | 0 |
| Max value: | 100 000 |
| Applicability: | Rx |

**RSSI Update Period**

| | |
|---|---|
| Description: | RSSI Update Period adjusts the length of the measurement window and update interval on the RSSI measurements. The control adjusts the measurement window and update interval in the following fashion: T_meas/update = 2^RSSI_Update_Period, where the unit is typically one baud. In other words the measured RSSI value will be averaged over the set time period. |
| Unit: | typically one baud time (geared with above equation) |
| 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 1st one is the average correlation value as measured on the 1st eight symbols in the frame (that is after sync word) the 2nd one is the number of symbols in the frame that has a correlation value lower than a threshold set in this entry. The signal quality indicator itself referred to as link quality indicator (LQI) can be accessed with RAIL API function RAILCb_RxPacketReceived. |
| Unit: | N/A |
| Min value: | 0 |
| Max value: | 255 |
| Applicability: | Rx |




## 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 LABS®

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

**http://www.silabs.com**