# USER MANUAL

## AX-RADIOLAB FOR

## AX5043, AX5045

Version 2.11a

20210127

ON Semiconductor®

# Table of Contents

# 1. Introduction

The AX-RadioLAB is a hands on configuration and firmware source code generation tool for the AX5043, AX5045 transceiver IC.

It creates C-code firmware projects based on the AXRadio API Manual (described in DOCU/ AXRadioAPIManual.pdf), which demonstrate the use of the AX5043, AX5045 transceiver together with the AX8052F100 and AXM0F243 SoC for packet transmission and reception in several modes. Packets can be transmitted periodically or upon pressing a button. On the receiver side the available options are:

- RX continuous (RX is always on)
- Wake on Radio
- RX Synchronized with TX (The RX is on for the time slots in which packets are expected only.)

Optionally received packets can be answered with an acknowledge packet in all modes.

AX-RadioLAB allows direct downloading of the created firmware to DVK-2x main boards for 8052 micro controller and AX8052F143 MINI-DVK. The GUI also enables direct downloading of the firmware into SoC AXM0F243-MINI-DVK. Further, the GUI supports AX5043, AX5045 radios with 8052 microcontroller on DVK-2x main boards. The downloaded firmware can be run and tested immediately. A dedicated firmware project implements basic tests like putting the transmitter into CW mode or measuring bit error rates.

The first step in working with AX-RadioLAB is to create a new AX-RadioLAB project. Firmware projects implementing the TX and RX mode selected in the AX-RadioLAB GUI are created inside your project directory. The AX-RadioLAB GUI makes it easy to configure physical parameters of the wireless link as well as the framing format and various possible functions general purpose pins available on the radios.

Optionally you can view and modify the generated firmware source code using the AXCode::Blocks IDE. The firmware also makes a nice starting point for your own development. You can always use AX-RadioLAB to modify wireless parameters of your project, even after modifying the firmware.

# 2. Installation

**Hardware and Software Requirements to Install AX-IDE**

- **Windows 7 or later**
- **at least 4GByte RAM**
- **1GB free hard disk space**
- **free USB port**

**Software requirements: For AX-RadioLAB to work correctly you need to install the AX-IDE first. Install all sub-components of the AX-IDE (AXSDB, SDCC, ARM-GCC, USB driver, AXCode::Blocks)**

**To verify and validate AX8052 MCU, compilers such as SDCC, IAR is required to be installed using the AX-IDE installer.**

**To verify and validate SoC AXM0F243, compiler such as ARM GCC is required to be installed using AX-IDE installer.**

**AX-Codeblocks IDE is used for both devices to edit the firmware.**

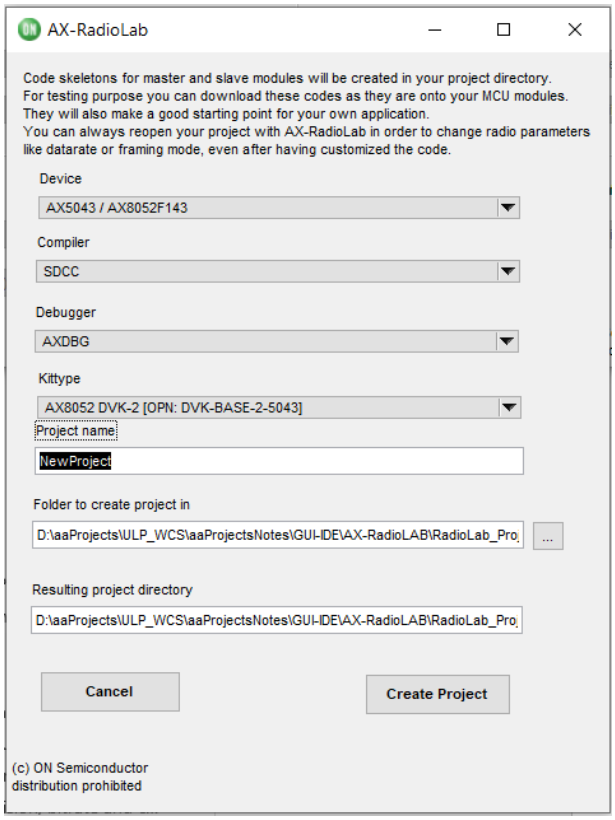
AX-Radiolab is installed as part of AX-IDE. If the software is already installed, uninstall the same before the new installation starts to in order to use the updated software.

At the end of installation, it may ask to reboot your computer. However, this is not mandatory.

# 3. Create a new Project

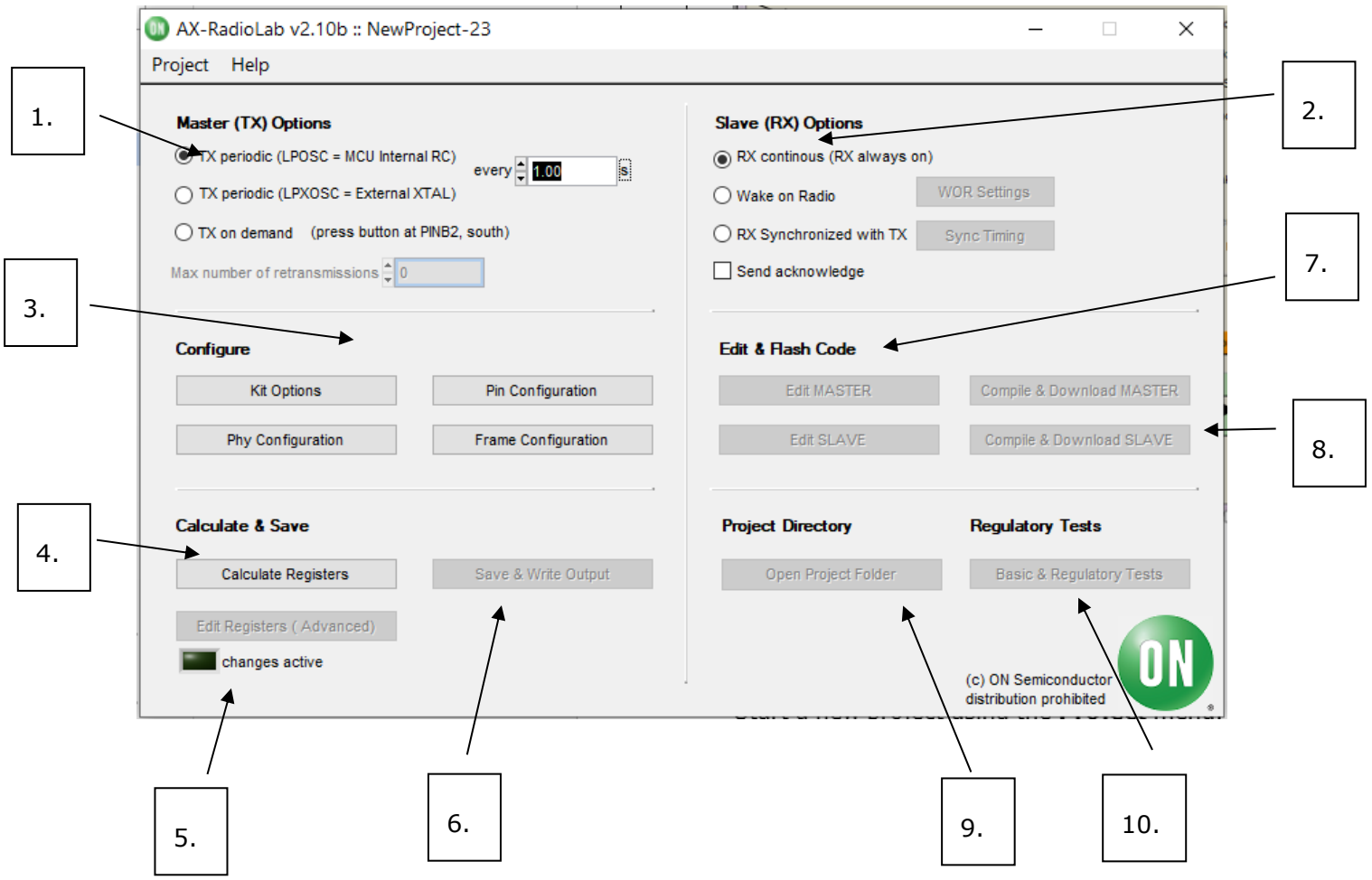
The AXRadiolab GUI can be used to create projects for various devices such as AX8052F143, X8052F145 and AXM0F243.

Open the AX-RadioLAB.exe to create a new project as below:




- Device: This populates the devices supported in the current Radiolab GUI namely AX5043/AX8052F143, AX5045 and AXM0F243.
- Compiler: Based on the chosen device, the compiler populates as below
  - For devices AX5043/AX8052F143, AX5045, supported compilers are
    - SDCC
    - IAR 8052
  - For Device AXM0F243
    - ARM GCC compiler
- Debugger: AXDBG is the only debug interface available with the current AX-Radiolab GUI
- Kit Types: This option is also populated based on the selected device
  - For AX5043/AX8052F143
    - DVK2 and Minikit is supported
  - For AX5045
    - DVK2 is supported
  - For AXM0F243
    - Minikit is supported

# 4. RadioLAB Main Panel




Start a new project using the **Project** menu. You have to specify a project directory where AX-RadioLAB will create firmware projects called MASTER, SLAVE and TESTS. To open an existing project use the directory chooser dialog to descend into the project directory (where the axradiolabstate.xml file resides) and hit done. Note: do not open the template_firmware_504x project inside the AX-RadioLAB installation directory, as this is the template from which the firmware is copied upon creating a new project.

Follow through the main panel step-by-step to configure your project:

1. Select the TX operation mode:

    - **TX periodic (LPOSC)**: Periodic packet transmission driven by the MCUs low power RC oscillator.

- **TX periodic (LPXOSC)**: Periodic packet transmission driven by the MCU's low power 32kHz tuning fork crystal oscillator
- **TX on demand**: Packet transmission upon pressing a button.

2. Select the RX operation mode: **RX Continuous (RX always on)**: Packet reception keeping the receiver continuously on. This mode is used for AC-powered systems and allows asynchronous packet reception with low latency.
   - **Wake on Radio**: Packet reception with the receiver waking up periodically and going immediately back to sleep if no signal is detected. The wake-up is driven by the Radio's low power RC oscillator and does not require any MCU activity. The MCU is only woken up once a valid packet has been received. This mode is well suited e.g. for infrequently operated remote control applications and achieves very low (idle) current consumption. **Mind that preambles longer than the time between RX wake-ups are necessary**. The latter also determines the RX latency.
   - **RX Synchronized with TX**: Periodic packet reception enabling the receiver only for the time slots in which packets are expected. This is appropriate whenever periodic data transmission is required. Timing is driven by the MCU's low power 32kHz tuning fork crystal oscillator and thus requires MCU activity. It is mandatory to use **TX periodic (LPXOSC)** on the transmitter side.
   - The **Send acknowledge** checkbox causes the slave module ("RX") to answer successful packet reception with an acknowledge packet. The master module ("TX") tries to receive the acknowledge packet. The smiley on the LCD indicates whether the acknowledge packet has been received. Failure is also signaled via LED2. Optionally the MASTER can retransmit a packet if acknowledge fails. This feature is enabled by setting **Max number of retransmissions** to a nonzero value. The acknowledge mode is the reason why the terminology MASTER and SLAVE (rather than TX and RX) has been chosen. The acknowledge feature can be selected in all TX and RX modes.

   TX and RX modes are summarized in Table 1. **Wake on Radio** and **RX Synchronized with TX** modes are described in more detail in section 3.1**Error! eference source not found.** and 4.2.

3. Buttons to open configuration panels for Kit Configuration, Pin Configuration, PHY layer and Framing. The PHY panel covers physical parameters of the wireless link such as carrier frequency, data rate and the modulation scheme. The Framing panel deals with packet delimiting, CRC etc.
4. **Calculate Registers** calculates register values to configure the selected device/Radio according to your Pin Configuration, PHY and Framing selections.
5. After calculating register values you can optionally tweak values using **Expert Settings**. Click values in the rightmost column of the **Expert Settings** panel to edit.

Tweaked values are sticky. They keep overriding the defaults even if the defaults have changed (e.g. upon making changes in the PHY panel and hitting **Calculate Registers**). Tweaked values are highlighted in yellow (or green if they accidentally correspond to the default values). The **changes active** indicator in the main panel reminds you that tweaked values are being used. Double click a tweaked register (column 1, 2 or 3) in order to return to the default value. Hitting **Reset Changes** at the bottom of the **Expert Settings** panel or clicking the **changes active** indicator in the main panel drops all tweaks.

6. **Save & Write Output**:

   The MASTER and SLAVE firmware projects are created in your project directory (if not present yet) upon hitting the **Save & Write Output** button. Manual changes in the firmware projects are never overwritten by the GUI except for the configuration files in your_projectdir /AX_Radio_Lab_output/.

   - Saves configuration and register values into your_projectdir\axradiolabstate.xml, allowing you to reopen your project with AX-RadioLAB.
   - Outputs files config.c, configmaster.[ch] and configslave.[ch] to your_projectdir\AX_Radio_Lab_output\
     These files are included by / linked to the MASTER, SLAVE and TESTS firmware projects.
   - The firmware projects MASTER and SLAVE are written into your project directory if they are not present yet.

7. The **Edit MASTER** and **Edit SLAVE** buttons open the corresponding firmware project from your project directory using the AXCode::Blocks IDE. This is handy for tweaking the firmware code, but touching the code is not necessary for first experiments with packet transmission. Instead you can download the demo firmware to the MCU directly from AX-RadioLAB (see next point). Button labels adapt to show which firmware projects are currently used.

8. The **Compile & Download MASTER** and **Compile & Download SLAVE** buttons compile and download the corresponding firmware onto the main board connected to the AXDBG USB debug adapter. A popup message indicating "Download Successful" or "Download Failed" is displayed. Note that **Compile & Download** does not work if AXCode::Blocks is currently opened. In this case just use **Save & Write Output** and then compile and download the code from within AXCode::Blocks.

9. Opens the AX-Radiolab project folder in explorer

10. Opens the **Basic & Regulatory Tests** panel, allowing you to transmit CW, simple patterns or random data and to measure bit error rates.

# 11. TX-RX Modes and timer usage

- AX8052 MCU

| Mode, Purpose | Used timer | Radio / MCU activity[1] |
|---|---|---|
| **TX periodic (LPOSC)**, Periodic packet transmission | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPOSC (640HZ RC oscillator) | MCU periodically wakes up, powers up the TX and initiates packet transmission |
| **TX periodic (LPXOSC)**, Periodic packet transmission (used with RX Synchronized with TX) | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPXOSC (32kHz tuning fork crystal oscillator) | MCU periodically wakes up, powers up TX and initiates packet transmission. |
| **TX on demand**, Packet transmission upon pressing a button or similar event. | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPOSC (640HZ RC oscillator) Only used for generating a timeout in ACK mode | MCU wakes up upon pressing a button, powers up the TX and initiates packet transmission. |
| **RX Continuous**, AC-powered systems (allows asynchronous packet reception) | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPOSC (640HZ RC oscillator) Only used for ACK timing and channel state polling. | RX is continuously on. Upon receiving a valid packet it generates an interrupt waking up the MCU. |
| **Wake on Radio**, Remote control (asynchronous packet reception at low idle currents at the expense of long preambles and increased latency) | AX5043 LPOSC (640Hz RC oscillator) for autonomous wake-up of the AX5043 MCU libmfwtimer infrastructure with timer0 clocked by MCU LPOSC (640HZ RC oscillator) Only used for generating a timeout in ACK mode | RX periodically wakes up and checks for signal. Upon receiving a valid packet it generates an interrupt waking up the MCU. |
| **RX Synchronized with TX**, Low power periodic data transmission | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPXOSC (32kHz tuning fork crystal oscillator) | MCU periodically wakes up, powers up the RX and waits for packet reception or timeout. |

**Table 1: Summary of TX and RX modes and corresponding firmware projects for AX8052 MCU**

[1] MCU activity apart from timekeeping wake-ups due to the running libmfwtimer infrastructure.

- AXM0F243

| Mode, Purpose | Used timer | Radio / MCU activity[2] |
|---|---|---|
| **TX periodic (LPOSC)**, Periodic packet transmission | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPOSC (40Khz RC oscillator) | MCU periodically wakes up, powers up the TX and initiates packet transmission |
| **TX periodic (LPXOSC)**, Periodic packet transmission (used with RX Synchronized with TX) | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPXOSC (32kHz tuning fork crystal oscillator) | MCU periodically wakes up, powers up TX and initiates packet transmission. |
| **TX on demand**, Packet transmission upon pressing a button or similar event. | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPOSC (40Khz RC oscillator) Only used for generating a timeout in ACK mode | MCU wakes up upon pressing a button, powers up the TX and initiates packet transmission. |
| **RX Continuous**, AC-powered systems (allows asynchronous packet reception) | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPOSC (40Khz RC oscillator) Only used for ACK timing and channel state polling. | RX is continuously on. Upon receiving a valid packet it generates an interrupt waking up the MCU. |
| **Wake on Radio**, Remote control (asynchronous packet reception at low idle currents at the expense of long preambles and increased latency) | AX5043 LPOSC (640Hz RC oscillator) for autonomous wake-up of the AX5043 MCU libmfwtimer infrastructure with timer0 clocked by MCU LPOSC (40Khz RC oscillator) Only used for generating a timeout in ACK mode | RX periodically wakes up and checks for signal. Upon receiving a valid packet it generates an interrupt waking up the MCU. |
| **RX Synchronized with TX**, Low power periodic data transmission | MCU libmfwtimer infrastructure with timer0 clocked by MCU LPXOSC (32kHz tuning fork crystal oscillator) | MCU periodically wakes up, powers up the RX and waits for packet reception or timeout. |

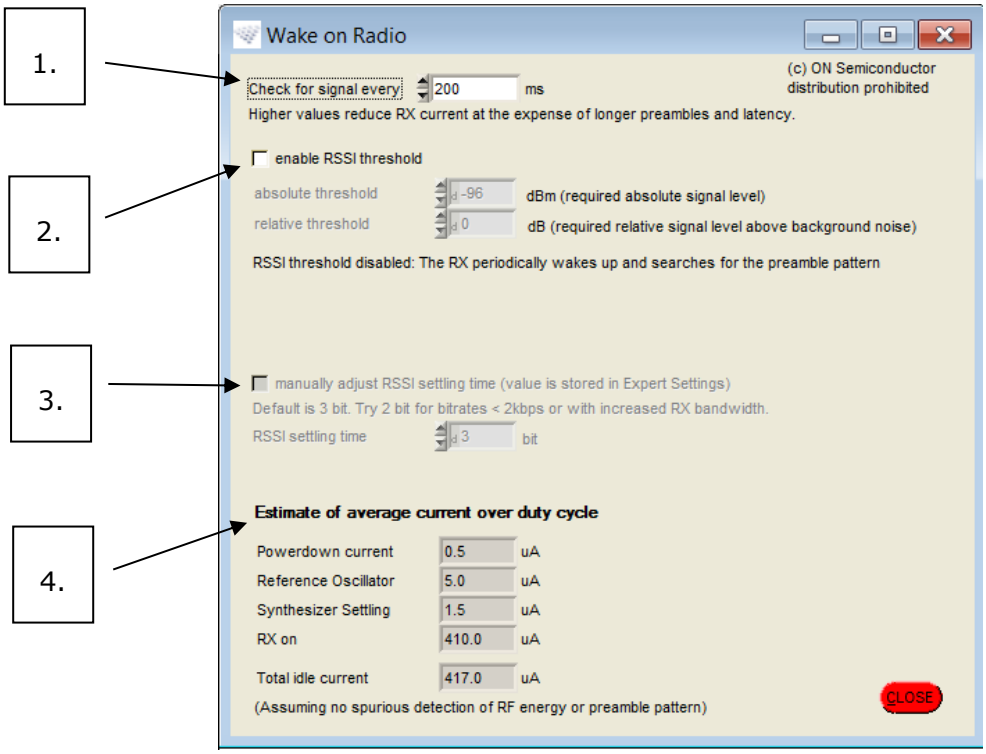
**Table 2: Summary of TX and RX modes and corresponding firmware projects for AXM0F243 MCU**

## 4.1. Wake on Radio Mode (WOR)

In **Wake on Radio** mode the receiver periodically wakes up and searches the channel for a signal. In absence of signal the receiver is switched off again immediately in order to save power. If a signal is detected the receiver stays on and looks for a valid packet delimiter. The Radio performs this task autonomously. Only after a valid packet has been received and is waiting in the FIFO, the Radio wakes up the micro controller via an

[2] MCU activity apart from timekeeping wake-ups due to the running libmfwtimer infrastructure.

interrupt. A typical application of this mode would be an infrequently operated remote control. The "WOR Settings" button opens the Wake on Radio panel:




1. This field configures how often the receiver wakes up and checks for signal. Waking up less often reduces the average RX (idle) current. However, it increases the necessary preamble length and average latency, since the preamble needs to be at least as long as the gap between two successive wake ups.

2. The "enable RSSI threshold" option determines how the RX searches for signal upon wake up:

    a. If this option is disabled the RX directly scans for preamble pattern. Sensitivity in this mode is comparable to continuous RX operation.

    b. If this option is enabled the RX first acquires an RSSI value. The scan for preamble pattern is performed only if RF energy is detected. This reduces current consumption (most notably at low data rates) since the RSSI measurement is faster than the scan for preamble pattern. The drawback is a reduction of sensitivity, since the RSSI threshold has to be set several dB above the sensitivity limit to prevent false positives. The RX proceeds to scanning for preamble if RSSI > absolute threshold AND RSSI >= (background RSSI + relative threshold). Background RSSI is the RSSI average over past wake ups.

3. This field allows customization of the RSSI settling time. Longer settling times give more stable RSSI measurements at the expense of higher current consumption. Changes are written to the expert panel.

4. The estimated (idle) RX current is displayed. The fractions for power down current, reference oscillator settling, synthesizer settling and RX on (RSSI acquisition or scan for preamble) are displayed separately.

Make sure you understand the following points:

- An important part of the process of powering-up the receiver and deciding whether a preamble signal is present takes a time inversely proportional to the data rate. To achieve minimum current consumption you should therefore use the maximum possible data rate (100kbps). The time the receiver searches for a preamble signal can be customized via the TMGRXPREAMBLE1 register in the expert panel.

- Frequency tracking (AFC) requires settling time, which adds to the receiver start-up time and therefore increases the average current consumption. Large data rates and precise frequency references render frequency tracking unnecessary.

- The average receiver (idle) current is proportional to the wake up frequency. On the other hand in the remote control mentioned above, the wake up frequency determines how quickly the receiver will react.

- Mind that the current estimate displayed in the main panel is the idle current in absence of a signal. Thus while the RX wakes up periodically, actual packet reception is the exception in this mode, occurring only e.g. if a remote control button was pressed. This is in contrast to the **Synchronized with TX** mode, in which a packet is expected each time the RX wakes up.

## 4.2. RX Synchronized with TX Mode

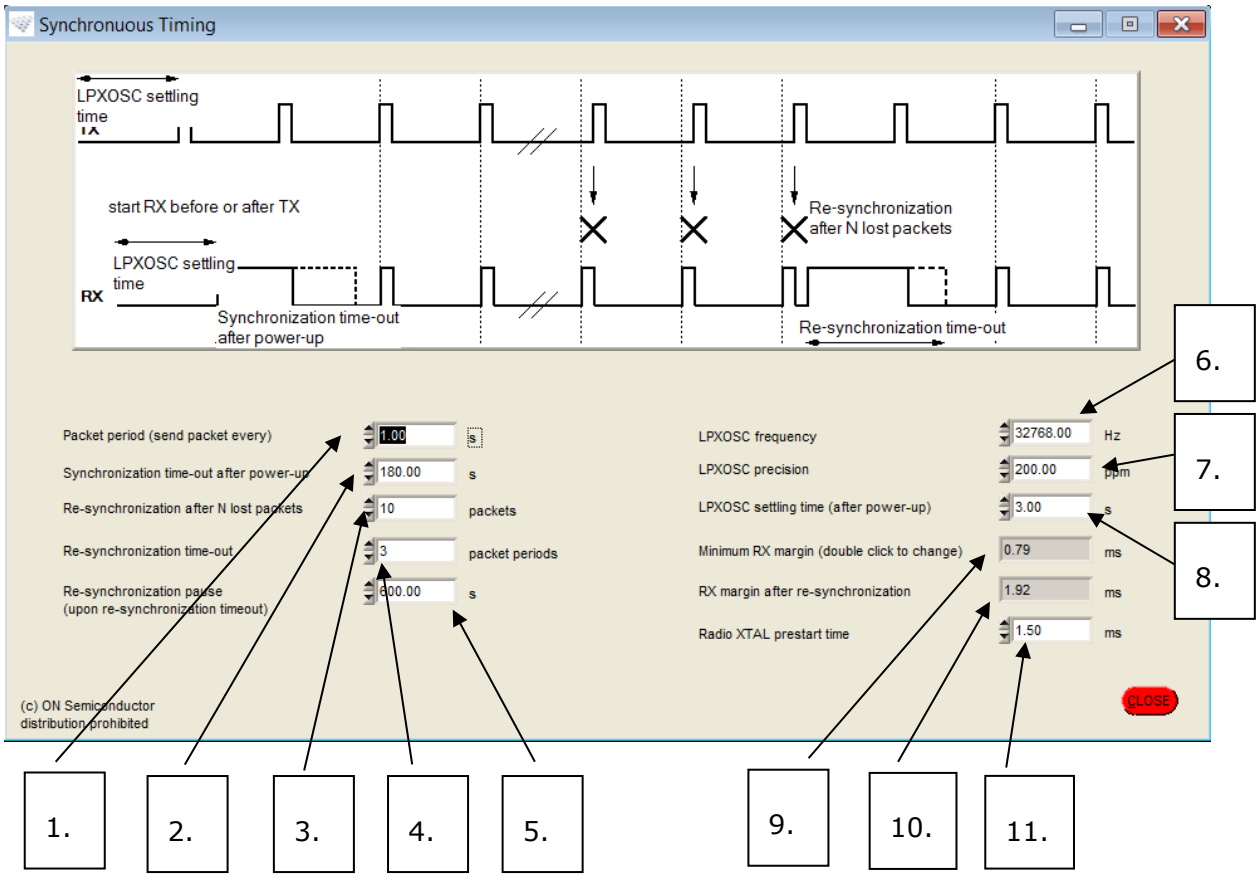
The purpose of **RX Synchronized with TX** mode is periodic reception of packets with minimal current consumption by enabling the RX only for the time slots in which packets are expected. Short preambles can be used on the TX side[3]. The following points explain how the **RX Synchronized with TX** mode works:

- **RX Synchronized with TX** mode is used together with **TX periodic (LPXOSC)** on the TX side. Timing is driven by the MCU's low power 32kHz tuning fork crystal oscillator which ensures an accurate packet frequency and thus allows for tight margins in the time slots in which the RX is enabled.
- After power-up LPXOSC is settled and the RX is switched continuously on until a packet is received or a timeout occurs. LED3 indicates the RX is running. When a packet is received the RX is put to sleep and only wakes up again for the time slot in which the next packet is expected.
- With each further packet reception the RX measures the effective time elapsed since the last packet and (via low pass filtering) corrects its own wake-up frequency accordingly. This corrects for crystal variations as well as temperature differences and slow gradients, and minimizes required margins in the RX-on time slots. (An increased margin is used for the packet reception right after synchronization, where the effective period has not been measured yet.)
- If no packet is received during an RX-on time slot, the RX margins are increased for the next time slot. If no packet is received during N subsequent time slots, the RX module switches back to synchronization mode, i.e. the RX is switched on continuously until a packet is received or a timeout occurs. (This synchronization mode is the same as for initial synchronization after power up. The only difference lies in the timeout interval: A timeout (in seconds) can be configured for initial synchronization after power-up (e.g. time to put batteries into the TX module). A second timeout (in packet periods) applies for re-synchronization after losing multiple packets in a row.
- If (re-)synchronization fails, (no packet is received until the timeout), the RX module is put to sleep for a configurable, typically long, period before trying to synchronize again. The rationale here is to limit RX power consumption in the case where the TX is off or out of reach.
- In **RX Synchronized with TX** the LCD displays the usual packet information. Pressing and holding button SW03 (south button on DVK-2 mainboards, connected to PINB2) during packet reception switches to the packet timing information display:

[3] This is in contrast to the asynchronous nature of **Wake on Radio** mode, where a preamble is at least as long as the RX sleep interval is required.

- "T:" is the current, low pass filtered packet period deviation measured in LPXOSC periods. Thus it indicates by how much the interval between packets is (on average) longer than programmed value from the perspective of the RX. It is understood, that the position of the next time RX on time slot is corrected by this deviation. Thus if TX and RX clocks run at slightly different but constant speeds, the corresponding timing deviation is displayed here, but the RX on time slots are perfectly aligned to the actual packets.
- "t:" indicates by how many LPXOSC cycles the last packet was late compared to the expected time, i.e. by how much it was misaligned to the RX on time slot. After acquisition of "T:" nonzero values of "t:" are due to timing fluctuations, which have to fit into the configured RX margin.

The **Sync Timing** button opens the Sync Timing panel, where various timing parameters can be configured.




1. **Packet period (send packet every)**: This is simply a duplicate of the **TX periodic (every)** field in the main panel for convenience.

2. **Synchronization time-out after power-up**: Time the RX tries to receive the first packet, before temporarily giving up and switching to re-sync pause mode. E.g. this is the time you have to put batteries into the TX module after powering up the RX.

3. **Re-Synchronization after N lost packets**: Determines in how many subsequent RX-on time slots packet reception can fail before the RX switches to back to synchronization mode.

4. **Re-synchronization time-out**: Time the RX tries to receive a packet for (re-) synchronization before temporarily giving up and switching to re-sync pause mode.

5. **Re-synchronization pause (upon re-synchronization time-out)**: Time for which the RX sleeps after failing to re-synchronize before trying again.

6. **LPXOSC frequency**: Frequency of the low power tuning fork crystal oscillator.

7. **LPXOSC precision**: Precision of the low power tuning fork crystal oscillator.

8. **LPXOSC settling time (after power-up)**: TX as well as RX modules wait for this time before transmitting / trying to receive packets. This is to prevent synchronization and period measurement with an insufficiently settled clock source.

9. **Minimum RX margin**: RX margin determines how long the RX is running, before the expected packet start. The same margin is also used as a timeout: The RX is powered down if no frame delimiter is detected by the time it was expected plus RX margin. (RX running means synthesizer running and analog baseband settled, but no AGC, AFC and bit synchronization, since there is no signal yet.) The minimum value specified here has to accommodate timing fluctuations in normal operation. A default value is computed from **packet period** and **LPXOSC precision**. Double click the field to change it manually. If packet reception fails in a given RX-on time slot, the firmware automatically increases the margin for the next time slot.

10. **RX margin after re-synchronization**: An increased RX margin which is used for the RX-on time slot right after (re-) synchronization. This is necessary, since (re-) synchronization only aligns the phases of periodic TX and RX. The RX needs a further packet to measure and adjust the wake-up period. Therefore the margin of the first RX-on time slot has to be large enough for the full, uncorrected timing deviation.

11. **Radio XTAL prestart time**: Determines how long the XTAL reference oscillator of the radio chip is settled before the RX synthesizer is started.

# 5. Kit Configuration Panel

In the Kit Configuration panel you can select the type of debug kit and the display options being used.

Available options:

**Devices/MCUs:**

Below listed devices are supported in the AX-Radiolab. Based on the device selection, the compiler displays the appropriate compiler option. For ex: SDCC and IAR 8052 for AX 8052 device, ARM-GCC for SoC AXM0F243 device. The display menu of Kit types is also updated based on the device selection.

**Devices List:**

- **AX5043/AX8052F143**
- **AX5045**
- **AXM0F243**

**Kit types based on the selected device:**

- DVK-2 / AX8052F100 + AX5043
- DVK-2 / AX8052F100 + AX5045
- F143-MINI-DVK / AX8052F143. F143-MINI-DVK does not support LCD display
- F243-MINI-DVK - F243-MINI-DVK does not support LCD display.

## 5.1. General Info

The kit configuration panel consists of device list, compiler list and kit types. This panel displays the devices, kit types, compilers and debuggers as selected by the User for reference. Enable/Disable LCD, Debug link etc could be selected/deselected by the user.

## 5.2. For AX8052F143

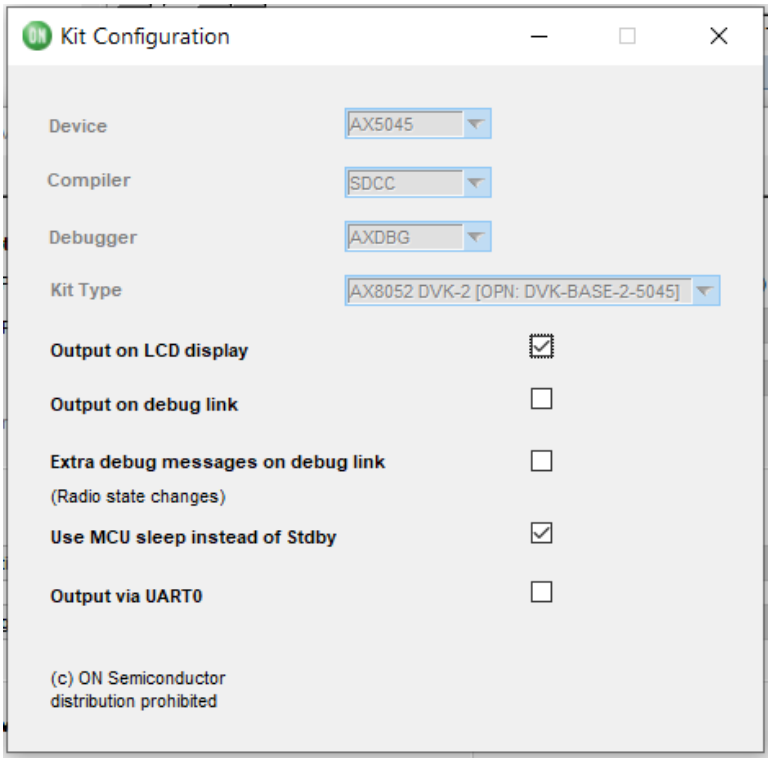
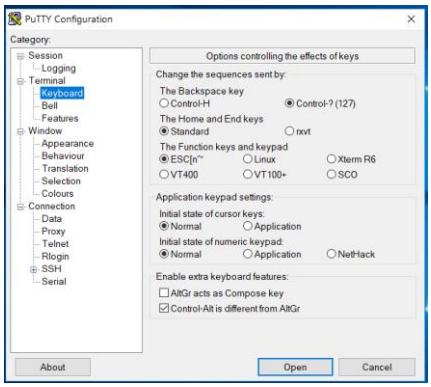



**Figure 1 Kit Configuration for AX8052F143/AX5045**

**Output on LCD display:** Packet numbers, bit and packet error rates, RSSI and other information is display on the LCD display of the DVK-2x

**Output via UART0:** Instead of sending the information to the LCD display it is send to UART0. This is the display mode for the MINI-DVK.

This option enables to print packet statistics. The PuTTY terminal settings should be configured to set baud rate = 115200, stop = 1, parity none, data = 8 bits, VT400 or VT100+ to be enabled.



**The usage of LCD and COM0 are mutually exclusive.**

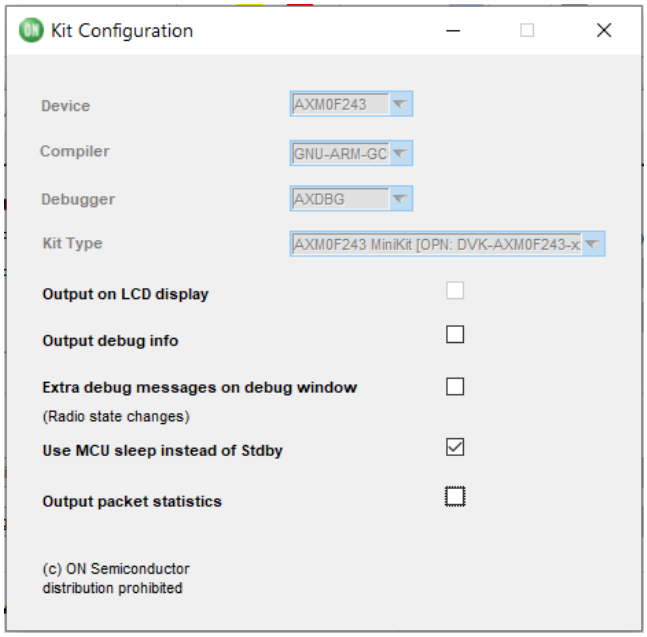
**Output on debug link:** In addition to one of the displays, the information is also send to the debug link.

**Extra debug messages on debug link:** This option produces extra information about the status change of the radio state on the debug link.

**MCU sleep instead of Stdby** option decides if the MCU uses sleep mode instead Stdby mode.

The above editable options are applicable to device AX5045.

## 5.3. For AXM0F243

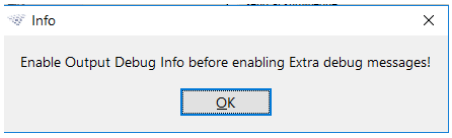



**Output on LCD display:** This is not enabled for MINIKIT as LCD display is not supported on the board.

**Output Debug Info:** For AXM0F243, the debug messages are transmitted to a serial terminal via UART0. This option enables to print debug information. The terminal settings should be configured to set baud rate = 9600, stop = 1, parity none, data = 8 bits.

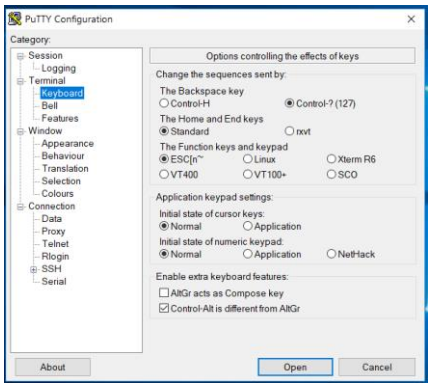
**Extra debug messages on debug window:** This option produces extra information about the status change of the radio state. This option is enabled only if serial output with debug info is selected. This option is disabled if serial output is selected with basic packet statistics. The below pop up appears when extra debug messages are enabled without

output debug info is enabled. This option should be enabled after output debug info is selected.



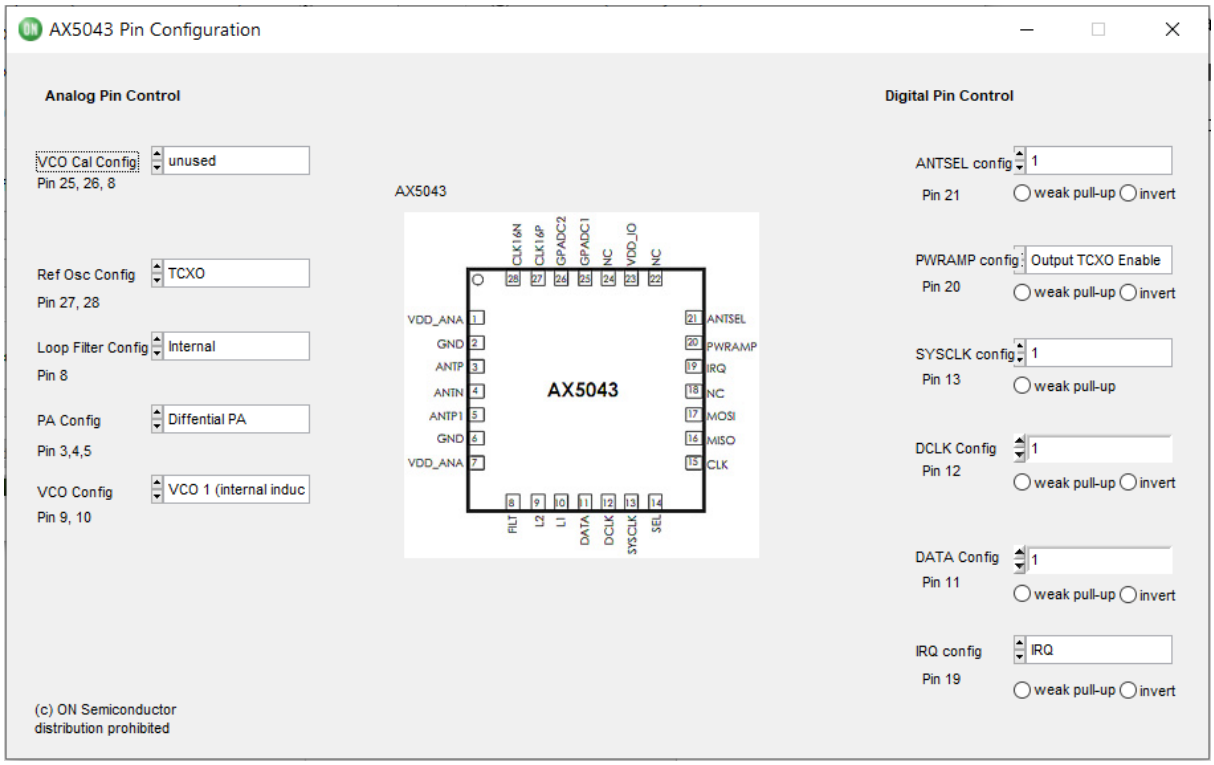

**MCU sleep instead of Stdby** option decides if the MCU uses sleep mode instead Stdby mode.

**Output Packet Statistics:** This option enables to print packet statistics. The PuTTY terminal settings should be configured to set baud rate = 115200, stop = 1, parity none, data = 8 bits, VT400 or VT100+ to be enabled.

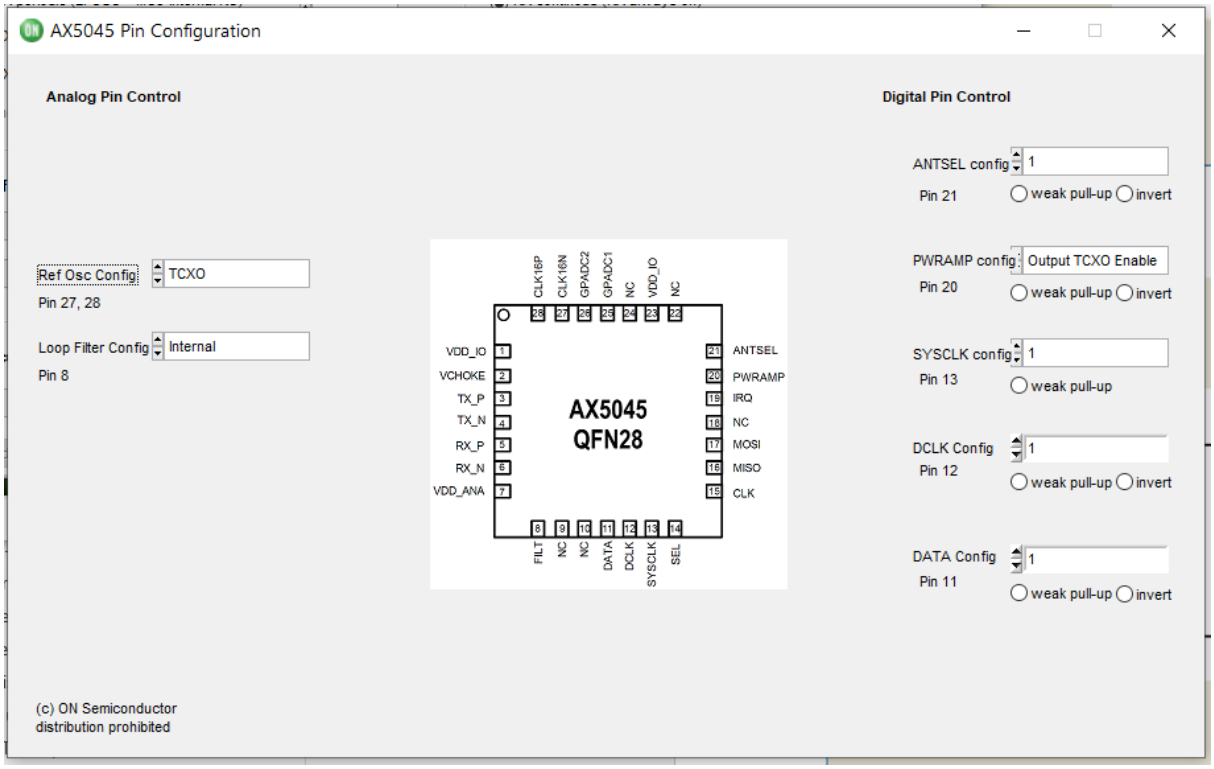


# 6. Pin Configuration Panel

For AX8052F143 device:



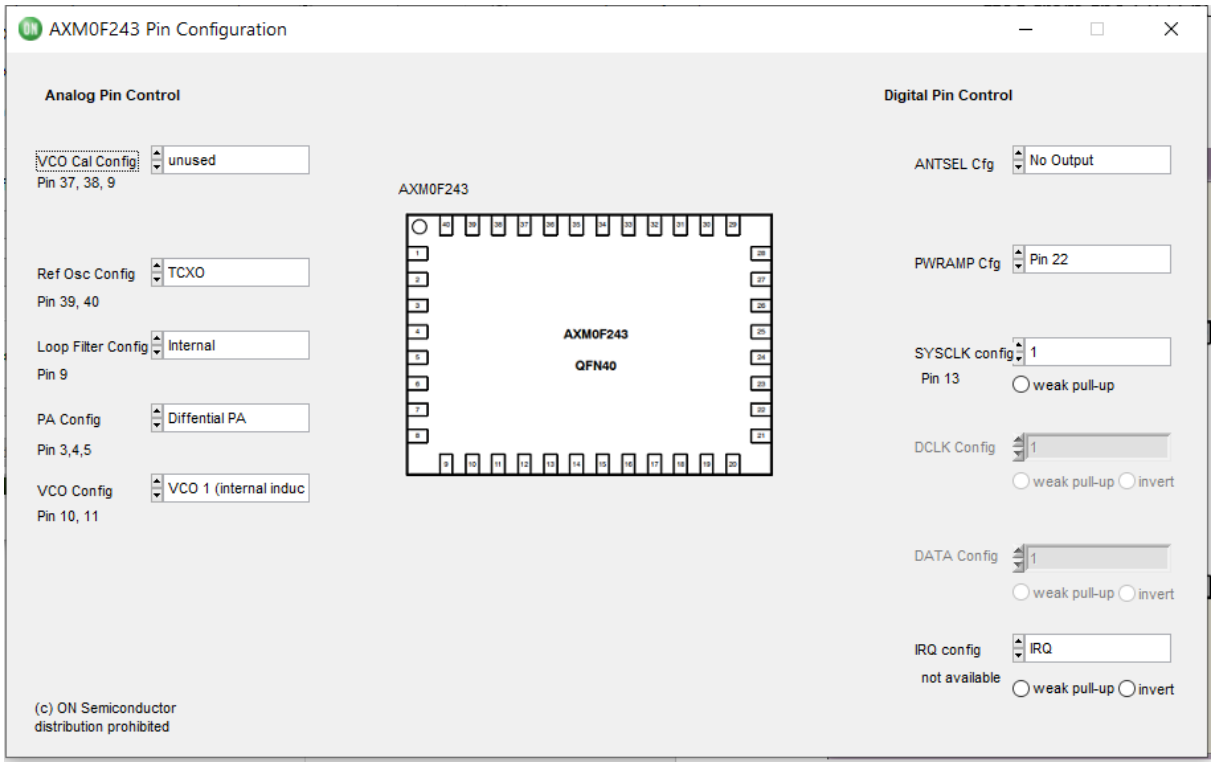
For AX5045 device:




- Ref Osc Config**:** select XTAL or TCXO
- PA Config: select differential, single ended or both power amplifier(s). This option is not available for AX5045.
- Loop Filter Config: determines whether the internal or an external PLL loop filter is used.
- VCO Config: Select VCO 1 (fully integrated VCO with on-chip inductor), VCO 2 (using an external inductor), or an entirely external VCO.This option available for only AX8052F143 device.
- ANTSEL Config: Pin can output TCXO enable signal, signal of the integrated DAC or Diversity Antenna Select.
- PWRAMP Config: Pin can output signal of the integrated DAC, Power Amplifier Control or TCXO enable signal. In the AX8052F143 SOC PWRAMP functionality is available on the PB2 GPIO pin.
- IRQ Config: Output IRQ. No other setting is sensible if the AX5043 module is connected to the main board. This option is available for AX8052F143 device.
- DCLK Config: use Modem Data Clock Output when observing modem data on the DATA pin.

- DATA Config: I/O Modem Data ; I/O Async Modem Data ; Output Modem Data (In this mode, during RX the received data appears on DATA. During TX data is transmitted from the FIFO but echoed on DATA.)

For AXM0F243 device,




- Ref Osc Config**:** select XTAL or TCXO
- PA Config: select differential, single ended or both power amplifier(s)
- Loop Filter Config: determines whether the internal or an external PLL loop filter is used.
- VCO Config: Select VCO 1 (fully integrated VCO with on-chip inductor), VCO 2 (using an external inductor), or an entirely external VCO
- ANTSEL Config: the ANTSEL is configured as "No Output" by default.
- PWRAMP Config: this Pin can output signal of external TCXO signal. The User is enabled to choose the pin on which the external TCXO is controlled.
- IRQ Config: Output IRQ. No other setting is sensible if the AX5043 module is connected to the main board.

# 7. PHY Panel

## 7.1. Overview

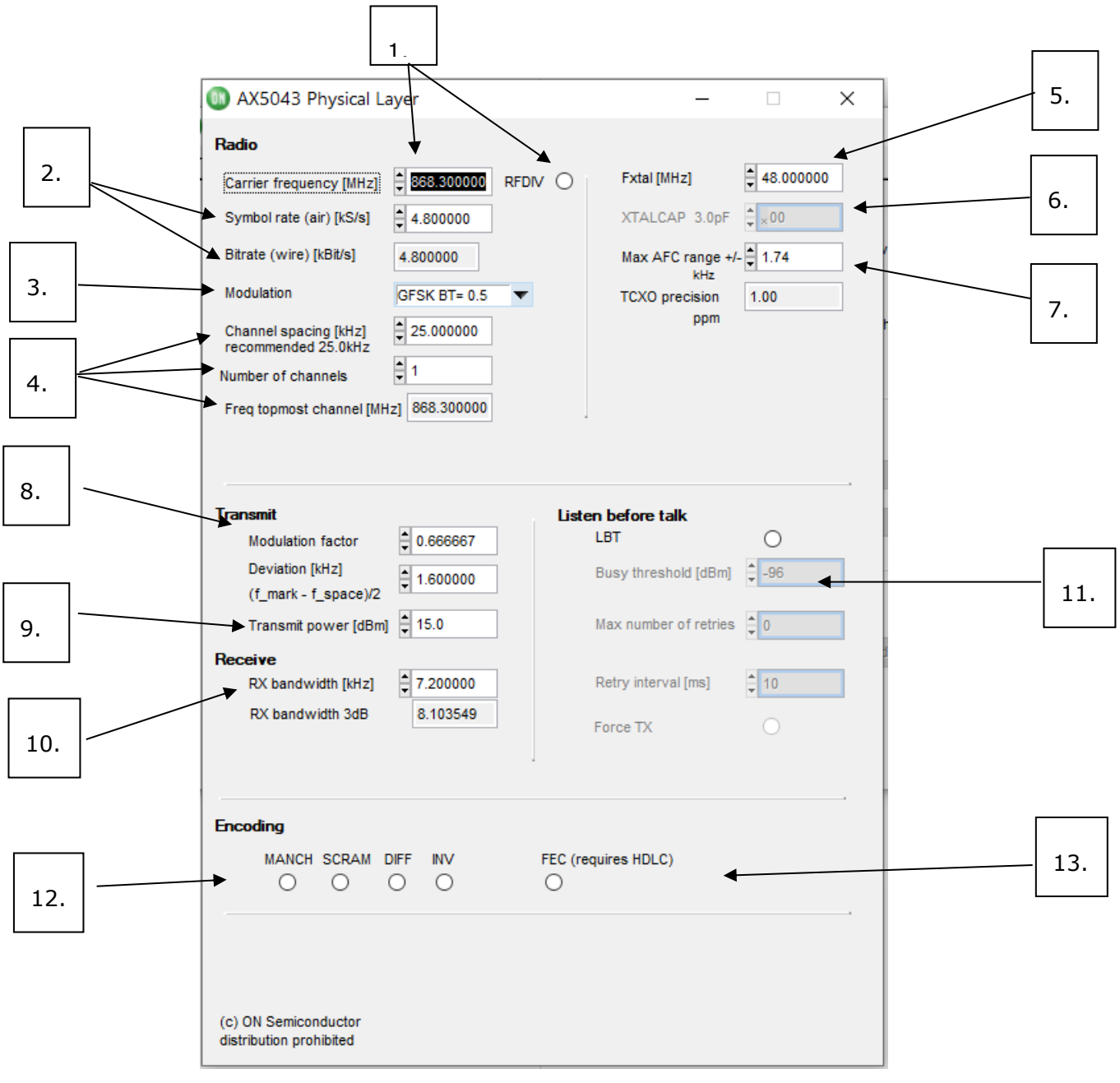



**Figure 1: Physical layer for AX8052F143/AX5045/AXM0F243**

1. Select Carrier Frequency. If more than one channel is configured (see below) the frequency selected here is that of channel number 0. RFDIV indicates that the VCO

is running at 4 times the LO frequency, rather than 2 times (depending on the frequency band). This is an indicator only unless you are using an external inductor and the selected frequency can be reached in both ways (using a different inductor value, of course).

2. Symbol rate & Bitrate specify the data rate. The values are identical except when using Manchester encoding (bitrate = 0.5*symbol rate) or 4-FSK (bitrate = 2*symbol rate).

3. Modulation Scheme.

4. Width of your channel. Red values signal a conflict with the TX and / or RX bandwidth. A number of channels can be specified. The firmware determines and stores the VCO range for each channel at startup. The carrier frequency of the topmost channel is displayed for convenience.

5. Frequency of the reference oscillator (XTAL / TCXO).

6. XTALCAP adjusts the on-chip XTAL load capacitance. Set to 0 when using a TCXO.

7. Max AFC range +/-: This value determines how far the frequency tracker (AFC) can walk in either direction. Setting it to zero disables frequency tracking.
   The necessary TCXO (or XTAL) precision is displayed. The computation is
   TCXO precision = (max AFC range) / ($2*f_{carrier}$), ensuring that the tracking range is sufficient to handle the case, where the TCXO of the TX and RX are maximally off in opposite directions.

   For more details and recommendations on setting up the AFC range and FSK deviation see chapter 11: XTAL/TCXO Precision, Frequency Tracking and Bandwidth.

   Note: Increasing the range of the AFC allows for the use of a less precise TCXO or XTAL. However, the drawbacks are

   a. Longer preambles may be necessary to settle the AFC. In WOR and RX Sync mode this enlarges the necessary receive windows and thus the duty cycled receiver current consumption.

   b. Larger RXBW may be necessary, since for successful tracking a sufficient amount of TX spectrum needs to fall into the RXBW. A warning is displayed if this is not guaranteed. See dedicated paragraph and figure below for details.

8. Modulation factor (often denoted as h).
   FSK deviation = 0.5 * modulation factor * Symbol rate = (f_mark-f_space)/2, sometimes called half deviation.
   4-FSK: Symbols are f_carrier + {-3, -1, +1, +3}*deviation

9. Transmit power in dBm.

10. Bandwidth of the RX channel filter, the upper field is used to input the desired value, the lower field shows the actual value.

11. Listen Before Talk (LBT) option: The TX measures the channel RSSI before sending a packet in order to assess whether the channel is free. If the RSSI value is found to be above busy threshold the TX can retry a number of times. If the channel continues to be busy the TX can either give up or transmit anyway (force TX).

12. Encoding: manchester, scrambler, differential and inversion.

13. Forward Error Correction (FEC). This encoding transparently adds redundancy to the transmit data in order to make it more immune against noise. FEC can only be used together with HDLC framing and it is mandatory to use HDLC flags (0x7E) as preamble characters. Other encoding modes such as SCRAM are not allowed and unnecessary in FEC mode, since FEC leads to data whitening by itself.

# 8. Framing Panel

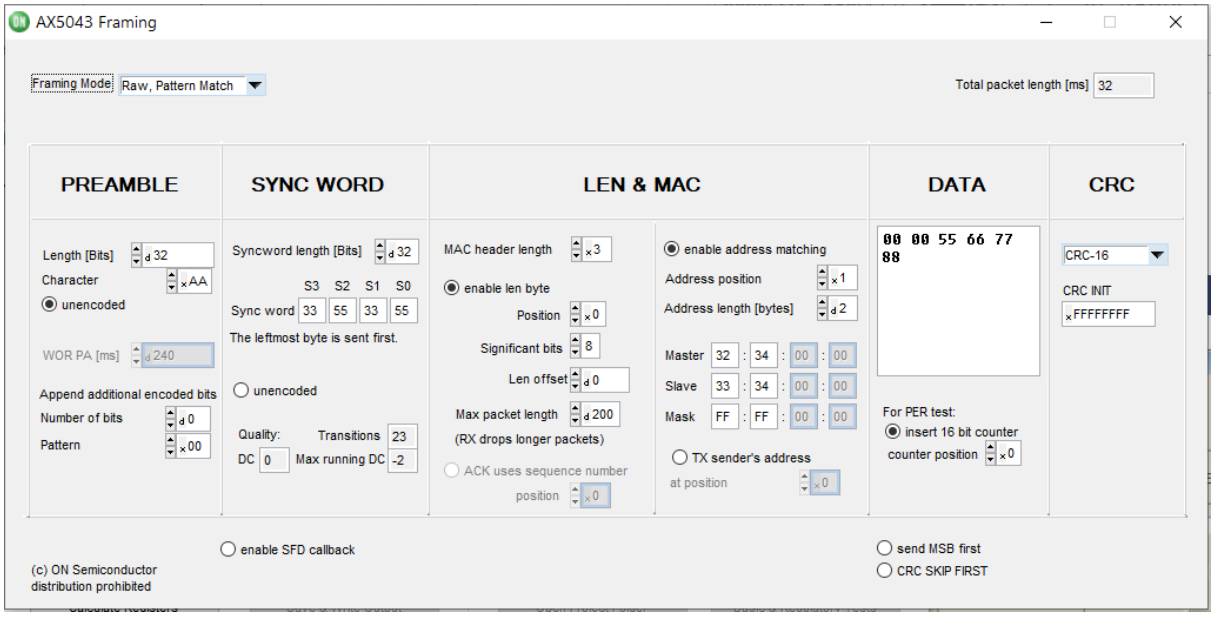



**Figure 2: Framing panel for AX8052F143/AX5045/AXM0F243**

**Framing mode**:

Options are HDLC, Raw Pattern Match and PN9 Compatibility. In Raw Pattern Match mode an arbitrary SYNC WORD of up to 32 bit marks the beginning of a frame. For legacy systems the PN9 Compatibility mode implements PN9 data whitening in software, based on the AX5043's Raw Pattern Match mode.

**Preamble:**

- Recommended default settings are chosen according to the selected modulation, encoding and precision of the reference clock.
- Specify the length of the preamble in bits, as well as one byte character which will be repeated length/8 times.
- The preamble character is always sent MSB first, irrespective of the "send MSB first" option in the DATA chunk. See comment on that option below.
- If length is not an integer multiple of 8, the fractional byte is sent last, e.g. for length = 33 bits the preamble character is sent 4 times followed by once the MSB of the preamble character.
- The "unencoded" option causes the preamble character to bypass the Manchester, scrambler, differential and inversion encoder. (See footnote to SYNC WORD for technical details about inversion.)

- In Wake on Radio mode an additional WOR preamble length (in ms) can be specified. This chunk is transmitted in addition to the number of bits specified above. The length of this chunk should not be shorter than the WOR period.
- Occasionally it may be necessary to conclude the preamble with a few special bits. This can be achieved using the fields below "Append additional encoded bits".

**SYNC WORD**

- In Raw Preamble Mode you can choose an arbitrary sync word of up to 32 bit as a start of frame delimiter. However it is recommended to use a sync word with well peaking auto correlation function.
- The SYNC WORD is always sent MSB first, irrespective of the "send MSB first" option in the DATA chunk. See comment on that option below.
- The "unencoded" option causes the SYNC WORD to bypass the manchester, scrambler, differential and inversion encoder[4].
  Correspondingly the RX matching unit detecting the SYNC WORD will operate on the bit stream before decoding.
- DC content, running DC content and the number of transitions are computed in order to indicate the quality of the selected SYNC WORD. Large DC or running DC components can impede correct frequency tracking, whereas a low number of transitions slows down time tracking.
- Selecting the "enable SFD callback" option causes the AXRadioAPI to notify the application level firmware each time a start of frame delimiter (a SYNC WORD or a HDLC flag) has been received.

**LEN & MAC**

- "MAC header length" specifies the length of the packet header, which can contain a length byte, a sequence number for acknowledge and addresses.
- In Raw Pattern Match mode a length byte can be used to communicate the length of the current packet to the RX. Alternatively (uncheck "enable len byte") the RX can be configured to expect packets of a fixed length.
- "Position" determines the position of the length byte inside the packet header. Zero means the length byte follows immediately after the SYNC word.

[4] Bypassing the encoder is achieved by setting the UNENC flag corresponding FIFO DATA chunk. This causes the chunk data to bypass manchester, scrambler and differential encoder, however inversion still applies. For convenience AX-RadioLab therefore inverts preamble and SYNC WORD bytes before writing them to the firmware code if "unencoded" and INV encoding are selected.

- "Significant bits" tells the RX how many bits of the length byte are used to encode the packet length. This is useful if your protocol uses a shared byte to encode the packet length plus something else.
- "len offset": The actual packet length (excluding SYNC WORD & CRC) assumed by the RX is len + len offset. E.g. set len offset to zero if in your protocol the length byte counts the number of bytes including the length byte. Set len offset = 1 if your length byte counts the number of bytes excluding the length byte itself. Negative values for "len offset" are allowed.
- If you check "enable len byte" in HDLC mode, the TX will send a length byte but it will be ignored by the RX.
- Specifying a maximum packet length can limit the impact of bit errors in the length byte.
- If the acknowledge feature is used an ACK sequence number field can be configured. This is not mandatory for acknowledge to work. However, without this field the SLAVE cannot detect duplicate packets occurring if packet RX has worked but the ACK packet got lost, causing the MASTER to retransmit the packet.
- Specify a device addresses of up to 4 bytes and a corresponding address mask in order to restrict reception to packets containing a matching address.
- "address position" specifies the starting position of the address inside the packet. Zero corresponds to the byte immediately after the SYNC WORD / HDLC flag.
- Optionally the sender's address can be added to the header. Note that address works on the destination address field (containing the slave's address in normal MASTER to SLAVE communication and the master's address for packets sent from SLAVE to MASTER, e.g. ACK packets in the demo firmware.)

**DATA**

- The DATA field allows you to specify a static sample payload which will be used by the demo firmware.
  Enter each byte as a two digit hex number (without leading 0x). Bytes may be separated by spaces.
- Optionally the firmware can insert a 16 bit counter into the data field at a specified position. This allows the RX to count the number of lost packets. The two bytes reserved for this counter are displayed as "00 00" in the data field.
- Checking the "send MSB first" button causes each byte in the DATA field to be sent MSB first. Otherwise each byte is sent LSB first.
  Note, however, that preamble character and SYNC WORD are always sent MSB first. (Technically, the RF chip will be configured to operate in MSB first or LSB first mode according to the "MSB first button", which applies to every byte. However, in LSB first mode the AX-RadioLAB GUI reverses the entered preamble and SYNC WORD bytes in order to keep the representation in the GUI MSB first.

The reason for this is, that the byte and nibble order of a SYNC WORD written left to right but read LSB first can be confusing.)

- "CRC SKIP FIRST" causes the first byte of the packet to be excluded from CRC calculation.

**CRC**

- Select CRC algorithm and initial value.
- The CRC is automatically calculated by the RF chip and appended to the packet. In HDLC mode the RX will store the received packet including CRC bytes. In Raw Packet Match mode, the RX strips the CRC bytes unless you manually set bit 5 in the radio specific PKTSTOREFLAGS register. For example: AX5043_ PKTSTOREFLAGS.
- Packets with incorrect CRC are silently dropped unless you manually set bit 2 in the Radio specific PKTACCEPTFLAGS register.
  For example: AX5043_PKTACCEPTFLAGS

# 9. Basic and Regulatory Tests Panel

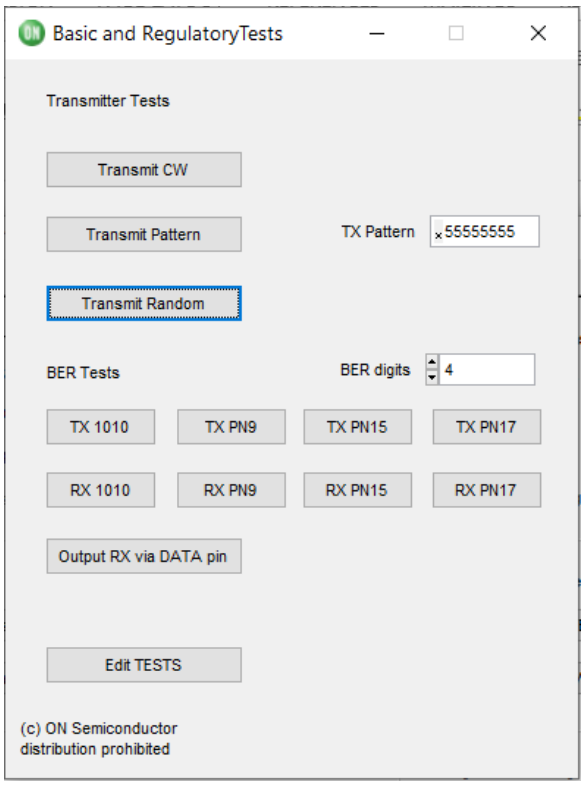



**Figure 3: Basic and Regulatory Tests panel for AX5043/AX8052F143/AX5045/AXM0F243**

- "Set CW" puts the RF chip to TX CW mode.
- "Set Pattern" transmits the specified 8 byte pattern forever. The byte specified by the two leftmost digits is sent first. Each byte is sent LSB first unless the "send MSB first" button in the framing panel is checked. Manchester, scrambler and differential encoder are bypassed (inversion encoding applies if selected!).
- "Set Random" transmits pseudo-random data forever. (This is realized using the scrambler.)
- "RX 1010" performs measurement of bit error rate (BER).
  Typically connect the RX to a signal generator sending an infinite 1010 stream, i.e. in FSK mode the generator should toggle between {+1, -1}*half deviation.
  In 4-FSK mode, the pattern {-3, -1, +1, +3}*half deviation should be applied.
  The BER value can be measured with 3, 4 or 5 digit precision.
- "TX 1010" transmits an infinite 1010 stream suitable for "RX 1010" BER measurement.

- "RX PN9", "TX PN9", "RX PN15", "TX PN15", "RX PN17", "TX PN17" are similar to "RX 1010" and "TX 1010", except that the BER measurement is performed using a pseudo-random data stream. In PN9 mode, the data stream is generated using the polynomial $x^9+x^5+1$, in PN15 mode, the polynomial $x^{15}+x^{14}+1$ is used, and in PN17 mode, the polynomial $x^{17}+x^{12}+1$ is used.
- "Output RX via DATA pin" allows observation of the received data via the DATA and DCLK pins. This is intended for performing BER tests where the bit errors are recorded by measurement equipment connected to the DATA and DCLK pins[5]. It is recommended to use measurement equipment which evaluates both, DATA and DCLK, since DATA can exhibit considerable jitter. This mode can be used for any kind of input data, 1010 data or pseudo random data streams. The raw received data is output on the DATA pin. Note, however, that the BER displayed on the LCD is correct only for 1010 data.
- Edit TESTS opens the firmware implementing the above test modes in the AXCode::Blocks IDE.

[5] The "RX1010", "RX PN9", "RX PN15" and "RX PN17" modes periodically settle the receiver with aggressive settings of the receiver tracking loops (RX parameters set 0) and then switch to the less aggressive settings (RX parameter set 3), where for instance the AGC loop is frozen completely. This is similar the situation of packet reception, where RX parameter set 3 is used after detection of a SYNC WORD. When measuring the bit error rate with external instrumentation this approach cannot be used. Receiving with RX parameter set 0 may lead to an increased bit error rate, while permanently receiving with RX parameter set 3 would not allow the AGC to settle. The "Output RX via DATA pin" therefore uses a variant of RX parameter set 3 where the AGC is running, typically with ¼ of the bandwidth of the RX parameter set 0 settings.

# 10. Comments on C-code Firmware

## 10.1. File Structure

The MASTER and SLAVE and TESTS firmware projects implement the corresponding modes using the AXRADIO API as a "driver" for the AX5043/AX5045 RF chip.

The AXRADIO API consists of the following files:

| File | Description |
|---|---|
| COMMON/axradio.h | AXRadio API declaration |
| COMMON/easyax5043.h | AXRadio AX5043 private header |
| COMMON/easyax5043.c | AXRadio AX5043 main code |
| COMMON/easyax5045.h | AXRadio AX5045 private header |
| COMMON/easyax5045.c | AXRadio AX5045 Radio code |
| AX_Radio_Lab_output/config.c | Parameters generated by AX-RadioLAB based on the selected device |

It is documented in DOCU/AXRadioAPIManual.pdf

The application level code consists of the following files:

| File | Description |
|---|---|
| MASTER/main.c | MASTER application level main code |
| SLAVE/main.c | SLAVE application level main code |
| TESTS/main.c | TESTS application level main code |
| COMMON/misc.[ch] | Application level helper files |
| SLAVE/display.c | Application level helper file |
| SLAVE/lposc.c | helper file |
| AX_Radio_Lab_output/configslave.[ch] | Application level parameters generated by AX-RadioLAB |
| AX_Radio_Lab_output/configmaster.[ch] | Application level parameters generated by AX-RadioLAB |

## 10.2. Define Statements for Tweaking the Firmware

**#define ADJUST_FREQREG** (SLAVE/main.c) if defined, after successful packet reception the receiver frequency is adjusted according to the tracked frequency offset. The frequency is not adjusted by the entire tracked frequency offset at once. Rather low pass filtering is applied. Low pass filtering is controlled by **FREQOFFS_K**, higher values give slower adjustments.

## 10.3. SYNC Packet RX Timing

In the AX-RadioLAB SYNC RX mode timing is controlled by the parameters `axradio_sync_xoscstartup`, `axradio_sync_slave_rxadvance`, `axradio_sync_slave_rxwindow` and `axradio_sync_slave_rxtimeout`. All times are measured in periods of libmf wtimer0, which is clocked by the low power crystal oscillator. As shown at the top of Figure 4 the radio XTAL oscillator of the SLAVE is enabled `axradio_sync_slave_rxadvance` before the frame delimiter (SFD) of the next packet is expected. After settling the XTAL for `axradio_sync_xoscstartup` the RX is enabled. The RX then settles and looks for an SFD, timing out after `axradio_sync_slave_rxwindow` if no SFD is received. If an SFD is received the RX stays on until successfully receiving a packet or until a timeout occurs after `axradio_sync_slave_rxtimeout`. (Not shown in the figure.)

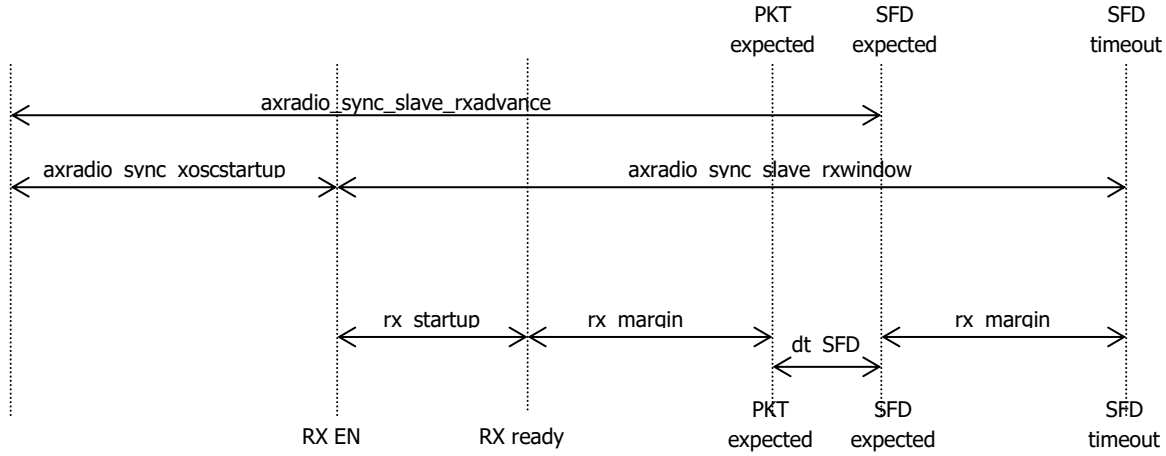



**Figure 4: SYNC firmware packet reception timing.**

AX-RadioLAB computes `axradio_sync_slave_rxadvance` and `axradio_sync_slave_rxwindow` based on the `rx_margin` which can be configured in the SYNC Timing panel as shown at the bottom of Figure 4. The goal is to have the RX ready `rx_margin` before the expected start of the packet. (Ready means that the synthesizer and the analog baseband are settled. AGC settling and bit synchronization can be done only if preamble signal is present.) The SFD timeout is set `rx_margin` after the expected SFD. Three instances of `axradio_sync_slave_rxadvance` and `axradio_sync_slave_rxwindow` are computed. The first instance contains the

increased `rx_margin` to be used for the first packet after (re-)synchronization when the effective packet period measurement is available yet. The second instance contains the `rx_margin` used in normal operation ("Minimum RX margin" in the SYNC Timing panel). The third instance has `rx_margin` scaled by a factor of 4 and is used after a failed packet reception.

Note: A processing delay occurs in the RX, i.e. the detection of the SFD is not signalled by the RX the very moment it is on air. On the other the RX should be enabled `rx_margin` before the packet is on air. Therefore `dt_SFD` is not simply the length of preamble plus SFD but also contains the processing delay of roughly 19 bit.

# 11. XTAL/TCXO Precision, Frequency Tracking and Bandwidth

This section gives some recommendations and background information on how to setup the PHY panel.

## 11.1. How to set up AFC range, FSK deviation and RX bandwidth

### Case 1: XTAL/TCXO precision, bitrate and carrier frequency are given

If XTAL/TCXO precision, bitrate and carrier frequency are given, how should the FSK deviation and the RX Bandwidth be set to achieve maximum sensitivity?

**Step 1**

Set carrier frequency and bitrate

Set Max AFC range so that XTAL/TCXO precision matches the precision of the crystal or TCXO being used. Note that changes over temperature of the XTAL/TCXO should be included.

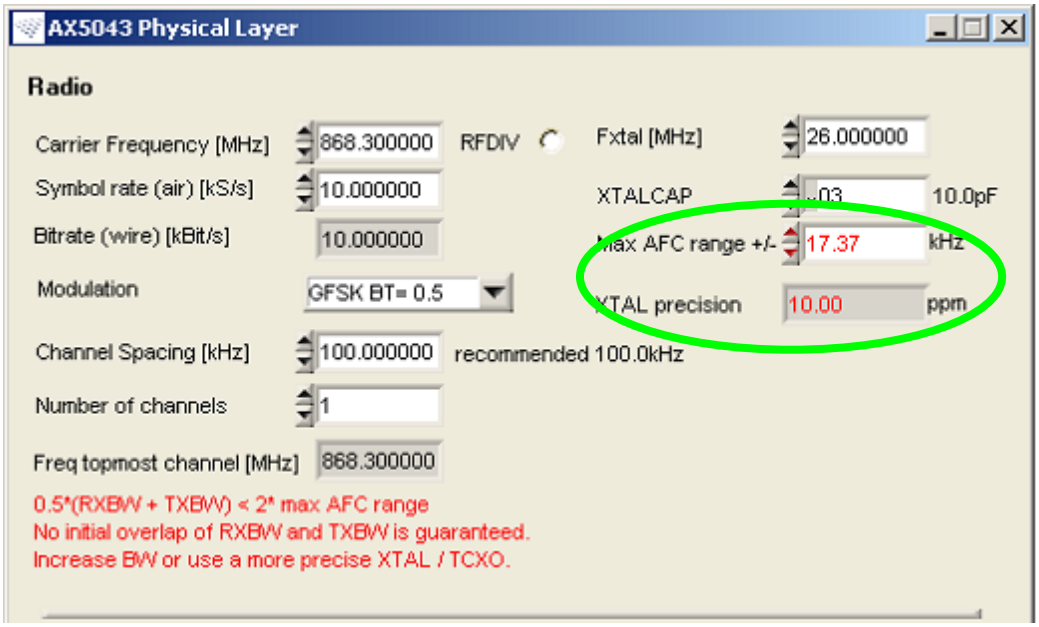



**Figure 5: 868.3 MHz, 10 kbps FSK, 10 ppm XTAL precision example for step 1 of PHY setup**

### Step 2

Set RX Bandwidth to 2 ● Max AFC range

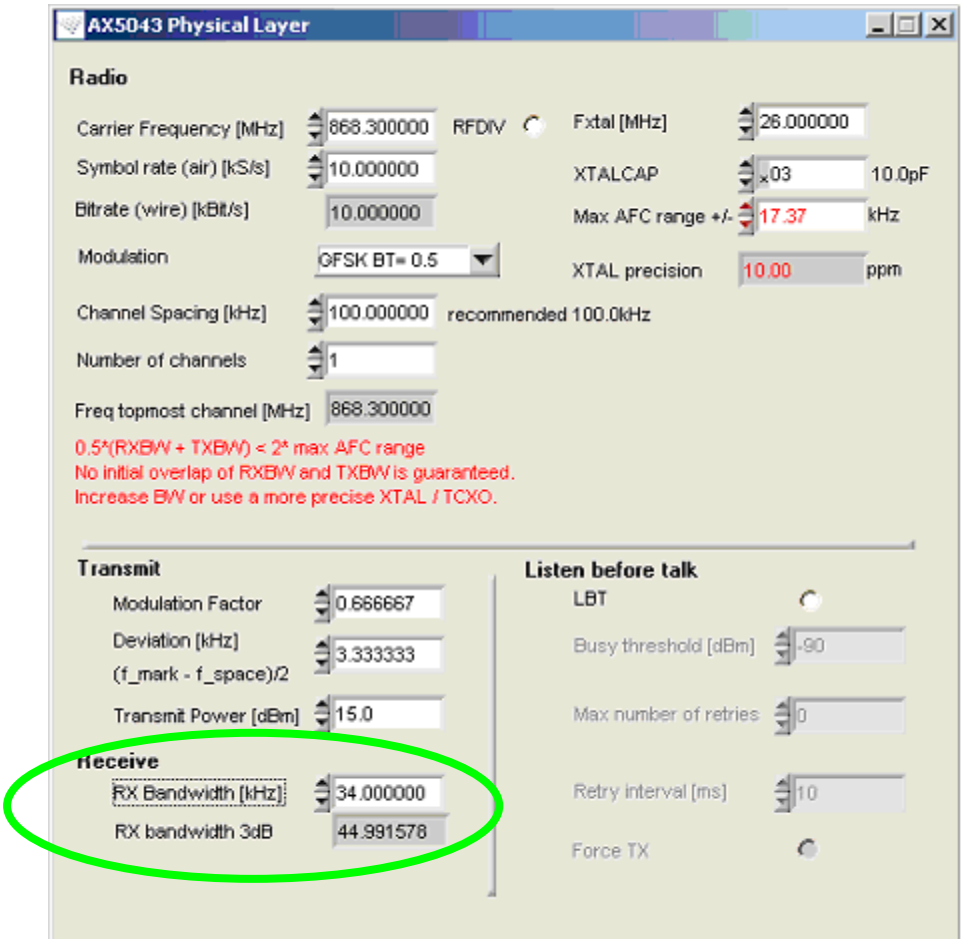



**Figure 6: 868.3 MHz, 10 kbps FSK, 10 ppm XTAL precision example for step 2 of PHY setup**

### Step 3

Increase the TX modulation factor or the deviation until the field XTAL precision is no longer red. This is typically at a value $Modulation\_Factor = \frac{2 \bullet Max\_afc\_range}{Bitrate} - 1$. If a modulation factor value larger than 5 emerges, then it is strongly recommended to increase the bitrate and to redo the setup procedure.

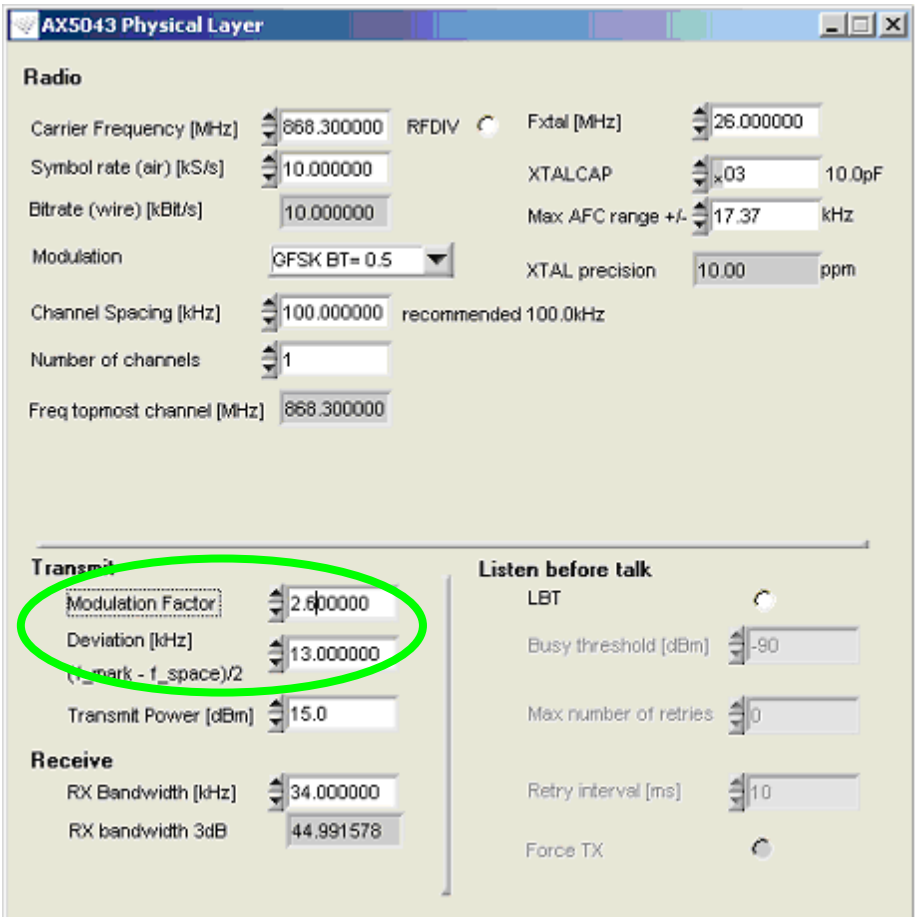



**Figure 7: 868.3 MHz, 10 kbps FSK, 10 ppm XTAL precision example for step 3 of PHY setup**

## Case 2: Bitrate and carrier frequency and FSK deviation are given

If the bitrate, carrier frequency and FSK deviation are given, how should the RX Bandwidth be set to achieve maximum sensitivity and what is the XTAL/TCXO precision that is required for this to work?

**Step 1**

Set carrier frequency, bitrate and FSK deviation or modulation factor.

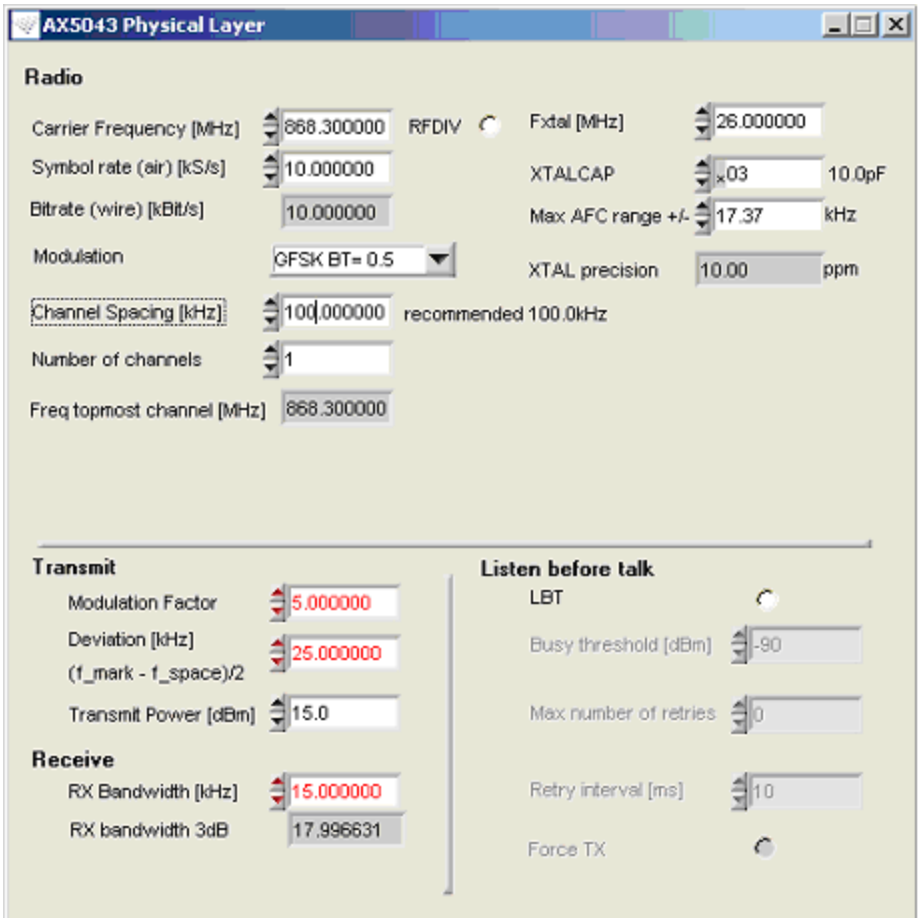



**Figure 8: 868.3 MHz, 10 kbps FSK, modulation factor 5 example for step 1 of PHY setup**

### Step 2

Set the RX bandwidth to $(1+Modulation\_Factor)\bullet Bitrate$.

If necessary increase the XTAL/TCXO precision by decreasing the Max AFC range until the red warning (No initial overlap of RX BW and TX BW is guaranteed.) disappears. This will occur at a Max AFC range value of RX BW/2.

If your XTAL/TCXO is not precise enough then increase the RX BW, but note that this will come at the cost of lower sensitivity.

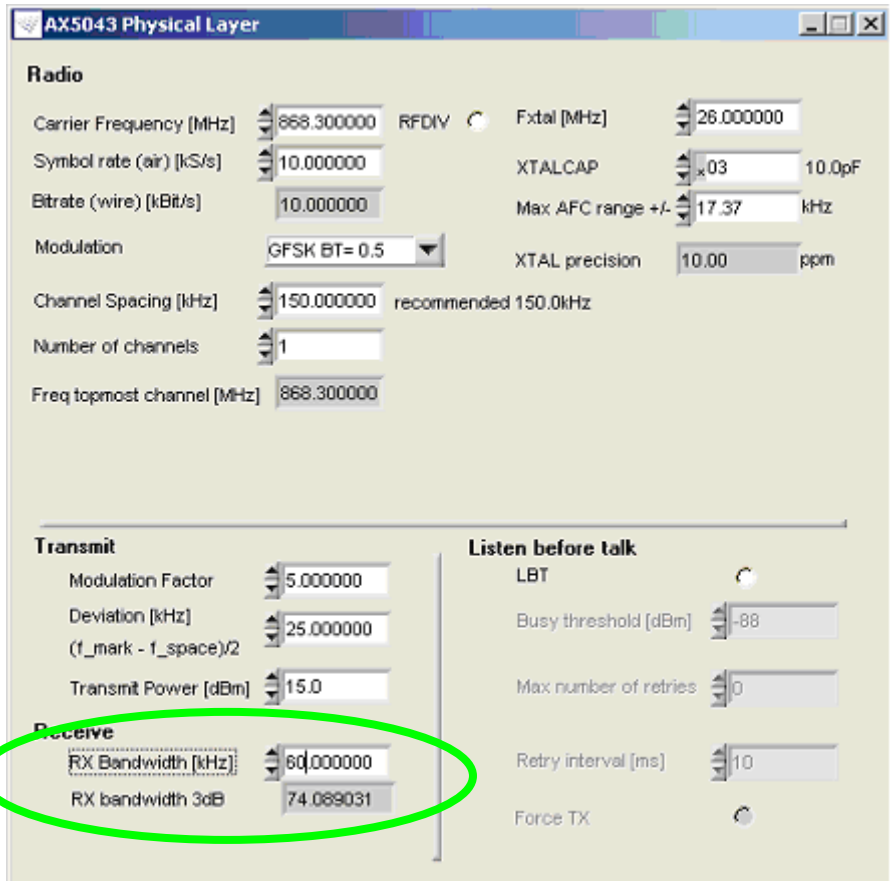



**Figure 9: 868.3 MHz, 10 kbps FSK, modulation factor 5 example for step 2 of PHY setup**

## 11.2. Background information

Without frequency tracking the worst case frequency deviation between RX and TX is

$$\Delta f = f_{carrier} \bullet (ppm_{TX} + ppm_{RX}),$$

Where $ppm_{TX}$ and $ppm_{RX}$ are the precisions of the crystals used in the transmitter and the receiver. The frequency tracker must be allowed to track this deviation, thus the "MAX AFC range" field in the AX-RadioLAB PHY panel (7) should be set to this value,

$$\text{Max AFC Range: } = \Delta f = f_{carrier} \bullet (ppm_{TX} + ppm_{RX}).$$

Note: AX-RadioLAB performs the inverse calculation. Given Max AFC Range, the necessary crystal precision is displayed. AX-RadioLAB assumes identical crystals are used in RX and TX, thus the displayed value is XTAL precision = $(ppm_{TX} + ppm_{RX})/2$.

The above setting allows the tracker to walk away $\Delta$f in the wrong direction while there is no signal (see Figure 10). In the worst case the receiver frequency can therefore be f°_RX when packet transmission starts, that is 2•Δf away from f_TX. For successful tracking a fraction of the TX spectrum should fall into the RX BW. Thus the yellow shaded region should be non-empty: (TX BW + RX BW)/2 ≥ 2•Δf. AX-RadioLAB issues a warning if this constraint is not fulfilled. Inserting

$$\text{RX BW = TX BW = } (1+h) \bullet BR = BR + 2 \bullet deviation$$

into the overlap condition leads to

$$h_{\min} = \frac{2 \bullet \Delta f}{BR} - 1.$$

Irrespective of the overlap condition $h \geq 0.5$ is required.

For optimal sensitivity RX $BW_{opt}$ = $(1+h) \bullet BR$ should be chosen. If values $h > 5$ emerges, increasing the bitrate should be considered as the RX BW for large h values has to be increased beyond RX $BW_{opt}$ to achieve stable operating conditions.

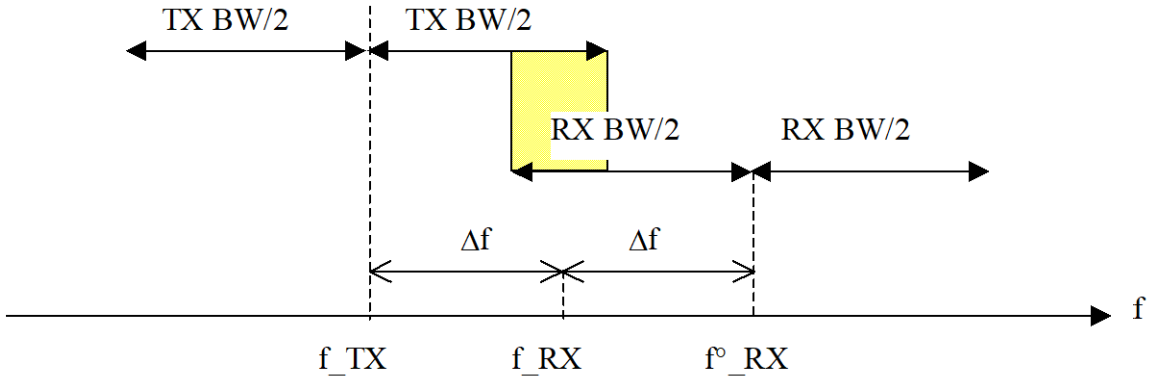



**Figure 10: Frequency precision, tracking and bandwidth**

Note: By default, AX-RadioLAB generated firmware adjusts the programmed receiver frequency upon successful packet reception. This feature can be disabled by commenting out the #define ADJUST_FREQREG statement in SLAVE/main.c.

## 12. Revision History

| | |
|---|---|
| Version2.8b | Updated for SoC AXM0F243 |
| Version2.8c | Updated for Panel GUI images |
| Version 2.8d | Updated section 4.2 |
| Version 2.8e | Updated serial terminal information for AX8052 |
| Version 2.8f | updated for devices AX8052F144 & AX8052F145 |
| Version 2.11a | Updated document for AX5045 |

# 13. Contact Information

**ON Semiconductor**
Oskar-Bider-Strasse 1
CH-8600 Dübendorf
SWITZERLAND

Phone +41 44 882 17 07
Fax +41 44 882 17 09
Email sales@onsemi.com
www.onsemi.com

For further product related or sales information please visit our website or contact your local representative.