ABSTRACT

This application report shows how the RX Sniff Mode can be configured for different TX configurations and discusses how average power consumption can be estimated for a receiver implementing this mode. You are expected to have basic knowledge on how the enhanced Wake on Radio (eWOR) and the RX Sniff Mode are implemented in TI's Performance Line Sub-1 GHz RF family of devices [1], [2].

Project collateral associated with this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/swra428.
1 Introduction

The RX Sniff Mode is a novel feature enabled by the TI Performance Line WaveMatch technology. The receiver only needs 4 bits of preamble for settling, as opposed to legacy receivers that often need 3-4 bytes. In a typical RF protocol where several preamble bytes are transmitted, TI Performance Line can autonomously duty cycle the receiver while waiting for a packet. Therefore, the RX Sniff Mode can be used fully transparent to you, while offering greatly reduced average current without sacrificing RF performance. The RX Sniff Mode is enabled by using the eWOR timer together with the RX termination based on carrier sense (CS) or preamble quality threshold (PQT).

Table 1. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGC</td>
<td>Automatic Gain Control</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CS</td>
<td>Carrier Sense</td>
</tr>
<tr>
<td>eWOR</td>
<td>Enhanced Wake on Radio</td>
</tr>
<tr>
<td>FIFO</td>
<td>First-In-First-Out</td>
</tr>
<tr>
<td>GPIO</td>
<td>General-purpose input/output</td>
</tr>
<tr>
<td>OSC</td>
<td>Oscillator</td>
</tr>
<tr>
<td>PQT</td>
<td>Preamble Quality Threshold</td>
</tr>
<tr>
<td>RC</td>
<td>Resistor-Capacitor</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received Signal Strength Indicator</td>
</tr>
<tr>
<td>RX</td>
<td>Receive Mode</td>
</tr>
</tbody>
</table>
2 Notation

Throughout this document, mapped register values are used. m(REGISTER_NAME.REGISTER_FIELD) equals the value in the description field that matches the bit pattern. If a register is not part of a m()-construct, it means that the bit pattern value is used. After a reset, m(WOR_CFG1.EVENT1) = 4 while WOR_CFG1.EVENT1 = 0.

Table 2. WOR_CFG1 - eWOR Configuration Reg. 1

<table>
<thead>
<tr>
<th>Bit No</th>
<th>Name</th>
<th>Reset</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:0</td>
<td>EVENT1</td>
<td>0x00</td>
<td>R/W</td>
<td>Event 1 timeout</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EVENT1</th>
<th>WOR_EVENT1</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>4</td>
</tr>
<tr>
<td>001</td>
<td>6</td>
</tr>
<tr>
<td>010</td>
<td>8</td>
</tr>
<tr>
<td>011</td>
<td>12</td>
</tr>
<tr>
<td>100</td>
<td>16</td>
</tr>
<tr>
<td>101</td>
<td>24</td>
</tr>
<tr>
<td>110</td>
<td>32</td>
</tr>
<tr>
<td>111</td>
<td>48</td>
</tr>
</tbody>
</table>

3 Configuring the Radio for RX Sniff Mode

This section explains all the different parameters and register fields involved when configuring the radio for the RX Sniff Mode. Table 3 shows the different register fields discussed and in which section they are covered.

Table 3. Register Fields Used by RX Sniff Mode

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Register Field</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>WOR_CFG1</td>
<td>WOR_RES</td>
<td>Section 3.2 and Section 3.8</td>
</tr>
<tr>
<td>WOR_CFG1</td>
<td>WOR_MODE</td>
<td>Section 3.10 and Section 3.11</td>
</tr>
<tr>
<td>WOR_CFG1</td>
<td>EVENT1</td>
<td>Section 3.3</td>
</tr>
<tr>
<td>WOR_CFG0</td>
<td>DIV_256HZ_EN</td>
<td>Section 3.12</td>
</tr>
<tr>
<td>WOR_CFG0</td>
<td>EVENT2_CFG</td>
<td>Section 3.1 and Section 3.4</td>
</tr>
<tr>
<td>WOR_CFG0</td>
<td>RC_MODE</td>
<td>Section 3.5</td>
</tr>
<tr>
<td>WOR_CFG0</td>
<td>RC_PD</td>
<td>Section 3.5</td>
</tr>
<tr>
<td>WOR_EVENT0_MSB</td>
<td>EVENT0_15_8</td>
<td>Section 3.2 and Section 3.7</td>
</tr>
<tr>
<td>WOR_EVENT0_LSB</td>
<td>EVENT0_7_0</td>
<td>Section 3.2 and Section 3.7</td>
</tr>
<tr>
<td>SETTLING_CFG</td>
<td>FS_AUTOCAL</td>
<td>Section 3.6</td>
</tr>
<tr>
<td>RFEND_CFG1</td>
<td>RXOFF_MODE</td>
<td>Section 3.10 and Section 3.11</td>
</tr>
<tr>
<td>RFEND_CFG1</td>
<td>RX_TIME</td>
<td>Section 3.7</td>
</tr>
<tr>
<td>RFEND_CFG1</td>
<td>RX_TIME_QUAL</td>
<td>Section 3.7</td>
</tr>
<tr>
<td>RFEND_CFG0</td>
<td>TERM_ON_BAD_PACKET</td>
<td>Section 3.10, Section 3.10.2, and Section 3.11</td>
</tr>
<tr>
<td>RFEND_CFG0</td>
<td>ANT_DIV_RX_TERM_CFG</td>
<td>Section 3.9, Section 3.9.1, and Section 3.9.2</td>
</tr>
<tr>
<td>AGC_CS_THR</td>
<td>AGC_CS_THRESHOLD</td>
<td>Section 3.9.1</td>
</tr>
<tr>
<td>AGC_CFG1</td>
<td>AGC_WIN_SIZE</td>
<td>Section 3.9.1</td>
</tr>
<tr>
<td>AGC_CFG1</td>
<td>AGC_SETTLE_WAIT</td>
<td>Section 3.9.1</td>
</tr>
<tr>
<td>AGC_CFG0</td>
<td>RSSI_VALID_COUNT</td>
<td>Section 3.9.1</td>
</tr>
</tbody>
</table>
Table 3. Register Fields Used by RX Sniff Mode (continued)

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Register Field</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREAMBLE_CFG0</td>
<td>PQT_EN</td>
<td>Section 3.9.2</td>
</tr>
<tr>
<td></td>
<td>PQT</td>
<td>Section 3.9.2</td>
</tr>
<tr>
<td>FIFO_CFG</td>
<td>CRC_AUTOFLUSH</td>
<td>Section 3.10.1</td>
</tr>
<tr>
<td>PKT_CFG1</td>
<td>CRC_CFG</td>
<td>Section 3.10.1</td>
</tr>
<tr>
<td>PKT_CFG0</td>
<td>LENGTH_CONFIG</td>
<td>Section 3.10.2</td>
</tr>
<tr>
<td>PKT_LEN</td>
<td>PACKET_LENGTH</td>
<td>Section 3.10.2</td>
</tr>
<tr>
<td>IOCFGx</td>
<td>GPIOx_CFG</td>
<td>Section 3.1 and Section 3.10</td>
</tr>
<tr>
<td>MDMCFG1</td>
<td>CARRIER SENSE_GATE</td>
<td>Section 3.2</td>
</tr>
<tr>
<td>SYNC_CFGx (1)</td>
<td>PQT_GATING_EN</td>
<td>Section 3.2</td>
</tr>
</tbody>
</table>

(1) PQT_GATING_EN is in the SYNC_CFG1 register for CC112x and SYNC_CFG0 register for CC120x.

3.1 Event 0, Event 1 and Event 2

The eWOR timer has three events: Event 0, Event 1, and Event 2. The relationship between Event 0 and Event 1 is shown in Figure 1. Event 2 is not used in the RX Sniff Mode (see Section 3.4) and is, therefore, not shown in the figure.

All three events can be monitored on the general-purpose input/output (GPIO) pins by setting IOCFGx.GPIOx_CFG = WOR_EVENT0/1/2 (54/55/56). If IOCFGx.GPIOx_CFG = WOR_EVENT2 (56), WOR_CFG0.EVENT2_CFG must be ≠ 00b. Event 2 is not shown in Figure 1 as it will not be used in most RX Sniff Mode applications. For more details, see Section 3.4.

3.2 Event 0 and t_{Event0}

The RX Sniff Mode Event 0 is the event used when the crystal oscillator and the digital regulator are turned on (when the radio is in SLEEP state). The time between two Event 0’s are called t_{Event0} and there are several factors to take into account when determining this time. It is recommended that both MDMCFG1.CARRIER SENSE_GATE and SYNC_CFGx.PQT GATING_EN = 0 when using the RX Sniff Mode to simplify how t_{Event0} should be determined.

- Termination based on CS (see Section 3.9.1)

Before the radio can detect a sync word, a minimum of 4-bit preamble is needed for AGC settling (including frequency offset compensation). This means that the maximum time between two Event 0’s can be calculated as shown in Equation 1 when RX is terminated based on CS.

\[
 t_{Event0 [Desired]} = \frac{\text{# of Preamble Bits} - 4}{\text{Data Rate [bps]}} \quad [s]
\]  

(1)
• Termination based on PQT (see Section 3.9.2)

If termination based on PQT is used, the radio needs a maximum of 10 bits of preamble to be sure that a preamble is detected (AGC settling and frequency offset compensation is included). The preamble quality estimator uses an 8-bit wide correlation filter to detect a preamble and 2 extra bits might be necessary to align the transmitter and receiver. In addition, the radio needs some time, T0, before starting to look for the preamble. For more information on how to calculate T0, see the device-specific user’s guides, [1] and [2]. This means that the max time between two Event 0’s can be calculated as shown in Equation 2 when RX is terminated based on PQT. Equation 2 is not valid when using OOK, feedback to PLL, or when TOC_LIMIT ≠ 0.

\[ t_{\text{Event0}}^{\text{Desired}} = \frac{\text{# of Preamble Bits} - 10}{\text{Data Rate [bps]}} - T0[\text{s}] \]  

(2)

\( t_{\text{Event0}} \) is given by the WOR_CFG1.WOR_RES, WOR_EVENT0_MSB, and WOR_EVENT0_LSB registers together with the frequency of the low-power RC oscillator as shown in Equation 2. For more details on WOR_RES, see Section 3.8.

\[ t_{\text{Event0}}^{\text{Programmed}} = \frac{1}{f_{\text{RCOSC}}} \cdot EVENT0 \cdot 2^5 \cdot \text{WOR}_\text{RES} [\text{s}] \]  

(3)

The RC oscillator has a tolerance of 0.1% after calibration (see [3] or [4]). This means that when programming \( t_{\text{event0}} \), Equation 4 should be used.

\[ t_{\text{Event0}}^{\text{Programmed}} = \frac{100}{100.1} \cdot t_{\text{Event0}}^{\text{Desired}} \]  

(4)

### 3.3 Event 1 and \( t_{\text{Event1}} \)

\( t_{\text{Event1}} \) is the time between Event 0 and Event 1. If \( t_{\text{Event1}} \) is larger than the crystal start-up time, an \( \text{SRX} \) strobe is issued on this event. Setting \( t_{\text{Event1}} \) larger than the crystal start-up time is useful in applications where the transmitter and receiver are in sync and one needs to put the radio in RX mode at a known time. In a typical RX Sniff Mode application, \( t_{\text{Event1}} \) should be set shorter than the crystal start-up time. In these cases, the \( \text{SRX} \) strobe will be issued as soon as the crystal is stable (CHIP_RDY_n is asserted). This way the radio enters RX mode as fast as possible, reducing the power consumption. \( t_{\text{Event1}} \) can be calculated as shown in Equation 5.

\[ t_{\text{Event1}} = \frac{1}{f_{\text{RCOSC}}} \cdot m(\text{WOR}_\text{CFG1.WOR}_\text{EVENT1})[\text{s}] \]  

(5)

### 3.4 Event 2 and \( t_{\text{Event2}} \)

At Event 2, the radio can autonomously be taken out of SLEEP mode to perform RC oscillator calibration and improve the accuracy of the eWOR timer. The time between two Event 2’s are called \( t_{\text{Event2}} \) and is given by Equation 6.

\[ t_{\text{Event2}} = \frac{2^m(\text{WOR}_\text{CFG0.WOR}_\text{EVENT2})}{f_{\text{RCOSC}}} [\text{s}] \]  

(6)
When enabling calibration at Event 2 by setting \( WOR\_CFG0.WOR\_EVENT2 \neq 0 \), \( t_{\text{Event0}} \) must be greater than \( t_{\text{Event2}} \) \[1\], \[2\]. Using the RX Sniff Mode does in most cases mean that \( t_{\text{Event0}} \) is much smaller than the minimum \( t_{\text{Event2}} \) (~1 s when \( f_{\text{RCOSC}} = 32 \) kHz and ~0.82 s when \( f_{\text{RCOSC}} = 40 \) kHz), therefore, \( WOR\_EVENT2 \) should in most RX Sniff Mode applications be set to 0.

### 3.5 RC Oscillator

To run the RX Sniff Mode, the internal RC oscillator must be enabled by setting \( WOR\_CFG0.RC\_PD = 0 \). In order to keep the frequency as accurate as possible, the RC oscillator needs to be calibrated. How often the RC oscillator should be calibrated is controlled through the \( WOR\_CFG0.RC\_MODE \) register field.

In an RX Sniff Mode application, the radio typically wakes up several times every second; it is recommended to do an initial calibration at start-up and then turn off the RC oscillator calibration (\( RC\_MODE = 0 \)). Re-calibration can then be initialized by the MCU.

The function found in Example 1 can be used to run a single RX oscillator calibration. In Example 1, \( \text{cc112x} \) should be replaced with \( \text{cc120x} \) in the function calls if the radio is a CC120X, and the register prefixes should be CC120X instead of CC112X.

**Example 1. RC Oscillator Calibration (CC112x)**

```c
/***********************
* @fn calibrateRcOsc
* @brief Calibrates the RC oscillator used for the eWOR timer. When this
* function is called, WOR_CFG0.RC_PD must be 0
* @param none
* @return none
*/
static void calibrateRCOsc (void) {
    uint8 temp;
    // Read current register value
    cc112xSpiReadReg(CC112X_WOR_CFG0, &temp,1);
    // Mask register bit fields and write new values
    temp = (temp & 0xF9) | (0x02 << 1);
    // Write new register value
    cc112xSpiWriteReg(CC112X_WOR_CFG0, &temp,1);
    // Strobe IDLE to calibrate the RCOSC
    trxSpiCmdStrobe(CC112X_SIDLE);
    // Disable RC calibration
    temp = (temp & 0xF9) | (0x00 << 1);
    cc112xSpiWriteReg(CC112X_WOR_CFG0, &temp, 1);
}
```

### 3.6 Frequency Synthesizer Calibration

The internal on-chip FS characteristics varies with temperature and supply voltage changes as well as the desired operating frequency and must be calibrated regularly. Calibration can be done automatically when going to or from active states (RX and TX) by setting \( SETTLING\_CFG.FS\_AUTOCAL \neq 1 \). Since the radio goes from IDLE to RX (and back to IDLE) several times a second when using the RX Sniff Mode, it is recommended to disable auto calibration \( FS\_AUTOCAL = 0 \) and do a manual calibration instead (when needed) to reduce current consumption. A manual calibration is performed by issuing an \( \text{SCAL} \) strobe command (for CC112x, see \[7\]).
3.7 RX Timeout

When using the RX Sniff Mode, RX is terminated when there is no carrier on the air or when no preamble is present, depending on the RFEND_CFG0.ANT_DIV_RX_TERM setting. For more details, see Section 3.8. Assume the radio wakes up in the beginning of the preamble and detects that a signal is present (CS or PQT). This means that ideally it should stay awake for a minimum time given by the preamble length plus the sync word length to be able to detect a sync word. The wakeup period, $t_{\text{Event0}}$, is programmed to be 4 bits or $T_0 + 10$ bits shorter than the preamble length (see Section 3.2). It is not possible to program the RX timeout to be larger than $t_{\text{Event0}}$, therefore, the RX timeout must be disabled when running the RX Sniff Mode ($\text{RFEND\_CFG1.RX\_TIME} = 111\text{b}$).

The RX\_TIME\_QUAL bit in the RFEND\_CFG1 register determines what happens when the RX timer expires. Since the RX Sniff Mode does not implement the RX timeout, this bit is don't care.

In a noisy environment, the MCU can be programmed to wake up if the radio detects a signal (CARRIER\_SENSE or PQT\_REACHED asserted). It can then start a timer before going back to sleep and wake up when the timer expires. If no sync is found, the MCU should put the radio back in SLEEP Mode by issuing an SIDLE strobe followed by an SWOR strobe (see Figure 2). Note that the radio needs to reach the IDLE state before the SWOR strobe can be issued.

Case A shows the format of the desired packet while case B shows how the MCU forces the radio back to SLEEP after a timeout equal to the preamble + sync length of the desired packet. In Case A, RX is terminated automatically after a packet is received ($\text{RFEND\_CFG1.RXOFF\_MODE} = 0$).

![Figure 2. MCU Controlled RX Timeout](image-url)
Configuring the Radio for RX Sniff Mode

### 3.8 eWOR Timer Resolution

WOR\_CFG1.WOR\_RES is used to configure the eWOR timer resolution and the resolution of $t_{\text{Event0}}$ (see Section 3.2).

<table>
<thead>
<tr>
<th>WOR_RES</th>
<th>Resolution</th>
<th>Max $t_{\text{Event0}}$</th>
<th>Resolution</th>
<th>Max $t_{\text{Event0}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31.25 µs</td>
<td>2.048 s</td>
<td>25 µs</td>
<td>1.638 s</td>
</tr>
<tr>
<td>1</td>
<td>1 ms</td>
<td>65.536 s</td>
<td>0.8 ms</td>
<td>52.429 s</td>
</tr>
<tr>
<td>2</td>
<td>32 ms</td>
<td>2097.152 s</td>
<td>25.6 ms</td>
<td>1677.722 s</td>
</tr>
<tr>
<td>3</td>
<td>1.024 s</td>
<td>67108.864 s</td>
<td>819.2 ms</td>
<td>53687.091 s</td>
</tr>
</tbody>
</table>

#### Table 4. Resolution and Max $t_{\text{Event0}}$ for Different Values of WOR\_RES

3.9 RX Termination

When implementing the RX Sniff Mode, the radio should terminate RX as fast as possible if there is no signal on the air to minimize the current consumption. The radio can terminate the RX mode in lack of a carrier (RFEND\_CFG0.ANT\_DIV\_RX\_TERM\_CFG = 1) or in lack of preamble (RFEND\_CFG0.ANT\_DIV\_RX\_TERM\_CFG = 100). Detecting a carrier takes less time compared to detecting a preamble and $t_{\text{Event0}}$ is shorter when RX termination is based on PQT compared to CS. Which RX termination to use (CS or PQT) depends on the system requirements and the environment the system is operating in. Consider the scenario shown in Figure 3. When the radio never triggers on noise, termination on CS gives the lowest total RX current. Figure 4 shows a scenario where noise is present in the channel and terminating on PQT gives the lowest RX current even if detecting a preamble takes longer than detecting a carrier. A third case is also shown in Figure 4. Here the MCU terminates RX after a timeout equal to the length of the preamble and sync word (the sync word is in this case half the length of the preamble). As seen from the figure, this leads to an even lower current consumption on the radio, but the MCU will draw some more current compared to case A and case B.

![Figure 3. Total RX Consumption in a Noise Free Environment](image-url)
3.9.1 Termination Based on CS

When termination based on CS is implemented (RFEND_CFG0.ANT_DIV_RX_TERM_CFG = 1), the CS threshold has to be set to a proper value. The CS threshold is programmed through the AGC_CS_THRESHOLD register field found in the AGC_CS_THR register. AGC_CS_THRESHOLD is a two's complement number with 1 dB resolution and is given by Equation 7. For details regarding the RSSI offset, see the device-specific user’s guides [1], [2].

\[
CS\text{ Threshold} = AGC_{\text{CS}\_\text{THR}} + \text{RSSI Offset}
\]  

(7)

Setting the optimal threshold is a trade-off between sensitivity and current consumption. The CS threshold determines the sensitivity limit of the application as only packets with a signal strength above the threshold will be received. The CS threshold should be set as low as possible to achieve good sensitivity (close to the sensitive limit given in the radio’s data sheet [3], [4], [5], [6]). However, setting the threshold too low causes more wake-ups due to noise and interference, and the current consumption will increase.
The carrier sense response time is the time it takes from when the radio enters RX mode until it can determine if there is a signal on the air or not. There are several registers that determines the CS response time and they are all discussed in the RSSI section of the user’s guides [1], [2]. Two Excel sheets are available for calculating the CS response time for both CC112x and CC120x [9], [10]. It is recommended to use typical settings from SmartRF™ Studio [8] when implementing the RX Sniff Mode and then set AGC_CFG1.AGC_SETTLE_WAIT, AGC_CFG1.AGC_WIN_SIZE, and AGC_CFG0.RSSI_VALID_COUNT to 0. When doing this, there will be a trade-off between CS response time and accuracy. Note that changing registers from the recommended settings to reduce the CS response time might reduce the sensitivity, so testing must be done to assure satisfying RF performance.

### 3.9.2 Termination Based on PQT

When termination based on PQT is implemented (RFEND_CFG0.ANT_DIV_RX_TERM_CFG = 100), the preamble detector must be enabled by setting PREAMBLE_CFG0.PQT_EN = 1. The preamble threshold is configured with the register field PREAMBLE_CFG0.PQT. The PQT response time is the time it takes from the radio enters RX mode until it can determine if there is a valid preamble on the air or not. For more info on PQT, see the Preamble Detection section in the device-specific user’s guide [1], [2].

### 3.10 Termination Due to Bad Packets

What happens after a bad packet has been received is determined by the TERM_ON_BAD_PACKET bit in the RFEND_CFG0 register (in this context, a bad packet is a packet with the wrong length, wrong address, or with CRC error). When TERM_ON_BAD_PACKET = 0, the radio stays in RX regardless of the RFEND_CFG1.RXOFF_MODE when a bad packet has been discarded. When TERM_ON_BAD_PACKET = 1, the radio will go back to SLEEP after rejecting a bad packet given that WOR_CFG1.WOR_MODE = 0 or 1. For more details, see Section 3.11. When a good packet is received, the radio will enter the state indicated by the RFEND_CFG1.RXOFF_MODE setting.

When making a low power system, it is not just the radio that should minimize its current consumption. The MCU running the application should also reduce its current consumption as much as possible. One thing that can be done is to only wake up the MCU when there is a good packet in the RX FIFO and let the radio go back to SLEEP automatically otherwise. This means that TERM_ON_BAD_PACKET should be 1.

Since the radio only wants to wake up the MCU when there is a good packet in the RX FIFO, the PKT_CRC_OK should be used to wake up the MCU (IOCFGx.GPIOx_CFG = PKT_CRC_OK (19)). On the CC112x, the GPIO2 pin is the only GPIO pin that will be 0 also in SLEEP state when configured as PKT_CRC_OK, while on the CC120x, both GPIO0 and GPIO2 will be low in SLEEP state.

#### 3.10.1 CRC Filtering

Since the radio is configured to go back to SLEEP if a packet is discarded (see Section 3.10), CRC filtering needs to be enabled. This is done by setting FIFO_CFG.CRC_AUTOFLUSH = 1. When this bit is set, it is important that the CRC check is enabled (PKT_CFG1.CRC_CFG ≠ 0).

#### 3.10.2 Maximum Packet Length Filtering

If the application uses variable packet length (PKT_CFG0.LENGTH_CONFIG = 1), maximum packet length filtering must be enabled. Maximum packet length filtering keeps the radio from entering RXFIFO_OVERFLOW state and also makes sure that packets that are too big for the RX FIFO are discarded so that the radio goes back to SLEEP (since RFEND_CFG0.TERM_ON_BAD_PACKET = 1).

The maximum allowed packet length must allow the complete packet, including optional status bytes, room in the RX FIFO. If PKT_CFG1.APPEND_STATUS = 0, max packet length is 127 and if APPEND_STATUS = 1 max packet length is 125. The maximum allowed packet length is written to the PKT_LEN.PACKET_LENGTH register field.

In an application where the radio does not expect packets longer than 20 bytes (length byte + payload), PACKET_LENGTH should be set to 19 even if there are room for longer packet in the RX FIFO.
3.11 eWOR Mode

There are five different eWOR modes to select from, but only two that can be used if the radio should return to SLEEP automatically on the reception of a bad packet, as discussed in Section 3.10. These two are Feedback Mode (\( \text{WOR\_CFG1.WOR\_MODE} = 0 \)) and Normal Mode (\( \text{WOR\_CFG1.WOR\_MODE} = 1 \)). For both modes, the radio enters the state indicated by the \( \text{RFEND\_CFG1.RXOFF\_MODE} \) setting when a good packet is received, and enter SLEEP if a bad packet is received (since \( \text{RFEND\_CFG0.TERM\_ON\_BAD\_PACKET\_EN} = 1 \)). Using Feedback Mode, the radio enters IDLE instead of SLEEP if 16 bad packets are received in a row. For more details, see [1] and [2].

3.12 eWOR Clock Division

Setting \( \text{WOR\_CFG0.DIV\_256HZ\_EN} = 1 \) lowers the current consumption in SLEEP mode, but when this bit is set, the radio should not be woken from SLEEP by pulling CSn low. The setting of this bit depends on the application’s need to manually wake up the radio.

4 Code Examples

There is one code example for the CC1120 and one for the CC1200 to supplement this application report. Both code examples can be downloaded from the web [11], [12].

4.1 CC1120 Code Example

The CC1120 code example uses the 1.2 kbps, 50 kHz RX filter BW typical settings from SmartRF Studio [8], version 1.13.0 as a starting point (see Figure 5).

![Figure 5. Typical Settings (CC1120, 1.2 kbps, 50 kHz RX filter BW)](image_url)
The register settings obtained by using the code export feature is shown in Example 2.

Example 2. Code Export (CC1120, 1.2 kbps, 50 kHz RX filter BW)

```c
// RX filter BW = 50.000000
// Address config = No address check
// Packet length = 255
// Symbol rate = 1.2
// PA ramping = true
// Performance mode = High Performance
// Carrier frequency = 868.000000
// Bit rate = 1.2
// Packet bit length = 0
// Whitening = false
// Manchester enable = false
// Modulation format = 2-FSK
// Packet length mode = Variable
// Device address = 0
// TX power = 15
// Deviation = 20.019531
// RF settings for CC1120
static const registerSetting_t preferredSettings[] = {
    {CC112X_IOCFG3, 0xB0},
    {CC112X_IOCFG2, 0x06},
    {CC112X_IOCFG1, 0xB0},
    {CC112X_IOCFG0, 0x40},
    {CC112X_SYNC_CFG1, 0x0B},
    {CC112X_DEVIATION_M, 0x48},
    {CC112X_MODCFG_DEV_E, 0x05},
    {CC112X_DCFILT_CFG, 0x1C},
    {CC112X_IQIC, 0x00},
    {CC112X_CHAN_BW, 0x04},
    {CC112X_AGC_CFG2, 0x05},
    {CC112X_AGC_CFG1, 0xA9},
    {CC112X_AGC_CFG0, 0xCF},
    {CC112X_FIFOCFG, 0x00},
    {CC112X_SETTLING_CFG, 0x03},
    {CC112X_FS_CFG, 0x12},
    {CC112X_PKT_CFG0, 0x20},
    {CC112X_PKT_LEN, 0xFF},
    {CC112X_PKT_CFG1, 0x00},
    {CC112X_FIFOCFG, 0x00},
    {CC112X_FREQOFF_CFG, 0x22},
    {CC112X_FREQ1, 0x6C},
    {CC112X_FREQ2, 0x80},
    {CC112X_FREQ0, 0x00},
};
```
Three different configurations (see Figure 6) are available for this example and they are discussed in the following sections (Section 4.1.1, Section 4.1.2, and Section 4.1.3).

### Configuration for RX Sniff Mode

#### RX Config. 1

In this configuration, three preamble bytes are used since this is the preamble configuration given by SmartRF Studio (see Figure 5).

- **tEvent0** Configuration
  
  Equation 8 shows \( t_{\text{Event0}} \) as maximum 16.67 ms.

  \[
  t_{\text{Event0}} = \frac{\text{# of Preamble Bits} - 4}{\text{Data Rate [bps]}} = \frac{3 \cdot 8 - 4}{1200} = 16.67 \cdot 10^{-3} \text{ s}
  \]  

  (8)

  This means that \( \text{WOR\_RES} \) should be 0 (see Table 4).

  Solving Equation 9 with respect to EVENT0, you get EVENT0 to be 532.80.

  \[
  \text{EVENT0} = t_{\text{Event0}}[\text{Programmed}] \times f_{\text{RCOSC}} = 100.1 \times t_{\text{Event0}}[\text{Desired}] \times f_{\text{RCOSC}} = 100.1 \times 16.67 \cdot 10^{-3} \times 32 \cdot 10^{-3} = 532.80
  \]  

  (9)

  \( \text{EVENT0} = 532.80 \) means that \( \text{EVENT0\_15\_8} = 0x02 \) and \( \text{EVENT0\_7\_0} = 0x14 \).

  In addition, \( \text{MDMCFG1.CARRIER\_SENSE\_GATE} = 0 \) and \( \text{SYNC\_CFG1.PQT\_GATING} = 0 \) as discussed in Section 3.2.

- **tEvent1** Configuration
  
  According to the CC1120 data sheet [3], the start-up time of the crystal is typical 400 µs, therefore, \( \text{EVENT1} \) should be less than 100\( _{\text{b}} \). In the CC1120 RX Sniff Mode code example [11] \( \text{EVENT1} = 000_{\text{b}} \).

- **tEvent2** Configuration
  
  Since \( t_{\text{Event0}} \) is 16.67 ms, \( t_{\text{Event0}} \) is less than the minimum \( t_{\text{Event2}} \) (1 s) and \( \text{WOR\_EVENT2\_CFG} \) should be set to 0 (see Section 3.4).

- **RC Oscillator Configuration**
  
  \( \text{WOR\_CFG0.RC\_PD} \) and \( \text{WOR\_CFG0.RC\_MODE} \) both set to 0 as described in Section 3.5.

- **FS Calibration**
  
  \( \text{SETTLING\_CFG.FS\_AUTOCAL} = 0 \) as stated in Section 3.6.

- **RX Timeout Configuration**

Figure 6. Code Configurations
RFEND_CFG1.RX_TIME = 111, and RFEND_CFG1.RX_TIME_QUAL = x as described in Section 3.7 (RX_TIME_QUAL = 1 in the code example [11] since this is the default value of this bit). No timeout is implemented, meaning the radio will stay in RX as long as there is a signal on the air.

• **RX Termination Configuration**

In the code example [11], termination based on carrier sense is implemented. This means that RFEND_CFG0.ANT_DIV_RX_TERM_CFG = 1. Because of this, PREAMBLE_CFG0.PQT_EN and PREAMBLE_CFG0.PQT are don’t care and the default values are being used (PREAMBLE_CFG0.PQT_EN = 1 and PREAMBLE_CFG0.PQT = 1010b)

When ANT_DIV_RX_TERM_CFG = 1, the CS threshold must be set to something other than default. The sensitivity limit for the chosen RF settings is -114 dBm [3]. In the code example [11], CS threshold is set 1 dB higher, at -113 dBm, to avoid too many wake-ups due to noise.

The RSSI offset is -102 dBm [1]. To get a CS threshold of -113 dBm, AGC_CS_THR must be set to 0xF5 (see Equation 10).

\[
\text{CS Threshold} = \text{AGC_CS_THR} + \text{RSSI Offset} \\
\text{AGC_CS_THR} = \text{CS Threshold} - \text{RSSI Offset} = -113 - (-102) = -11
\] (10)

To reduce the CS response time, AGC_CFG1.AGC_WIN_SIZE, AGC_CFG1.AGC_SETTLE_WAIT and AGC_CFG0.RSSI_VALID_COUNT are all set to 0.

• **“Bad Packet” Termination**

To save as much power as possible, the radio is configured to return to SLEEP if a bad packet is received and to only wake up the MCU if a good packet is received, therefore, RFEND_CFG0.TERM_ON_BAD_PACKET = 1 and IOCFG2.GPIO2_CFG = 010011b (see Section 3.10). (1)

In addition, FIFO_CFG.CRC_AUTOFLUSH must be 1 and PKT_CFG1.CRC_CFG ≠ 0. In the code example [11], CRC_CFG = 01b, as this is the default value of this register field.

RFEND_CFG1.RXOFF_MODE is set to 0 (IDLE) making the radio enter IDLE state when a good packet is received.

Since the radio should only wake up the MCU when a good packet is received, the radio must be configured to avoid entering RX FIFO ERROR state. The settings given in Example 2 indicates variable packet length mode PKT_CFG0.LENGTH_CFG = 01, and PKT_CFG1.APPEND_STATUS = 1, therefore, PKT_LEN should be set to 125 (0x7D) (see Section 3.10.2)

• **eWOR Mode**

To make the receiver SW as simple as possible, Normal mode (WOR_CFG1.WOR_MODE = 1) is selected for this code example [11].

• **eWOR Clock Division**

WOR_CFG0.DIV_256HZ_EN = 1 to make the current consumption as low as possible.

(1) IOCFG0.GPIO0_CFG = 000110b and is used by the TX to determine when a packet has been sent.
### Table 5. Register Fields to be Changed When Running RX Sniff Mode (RX config. 1)

<p>| Register Name | Register Field                | Value | Register Value |
|---------------|-------------------------------|-------|----------------|----------------|
| WOR_CFG1      | WOR_RES                       | 00b   | WOR_CFG1 = 0x08|
|               | WOR_MODE                      | 001b  |                |
|               | EVENT1                        | 000b  |                |
| WOR_CFG0      | WOR_CFG_NOT_USED              | 00b   | WOR_CFG0 = 0x20|
|               | DIV_256HZ_EN                  | 1b    |                |
|               | EVENT2_CFG                    | 0b    |                |
|               | RC_MODE                       | 0b    |                |
|               | RC_PD                         | 0b    |                |
| WOR_EVENT0_MSB| EVENT0_15_8                   | 0x02  | WOR_EVENT0_MS = 0x02|
| WOR_EVENT0_LSB| EVENT0_7_0                    | 0x15  | WOR_EVENT0_LSB = 0x15|
| SETTLING_CFG  | SETTLING_CFG_NOT_USED         | 00b   | SETTLING_CFG = 0x03|
|               | FS_AUTOCAL                    | 00b   |                |
|               | LOCK_TIME                     | 01b   |                |
|               | FSREG_TIME                    | 1b    |                |
| RFEND_CFG1    | RFEND_CFG1_NOT_USED           | 0b    | RFEND_CFG1 = 0x0F|
|               | RXOFF_MODE                    | 0b    |                |
|               | RX_TIME                       | 11b   |                |
|               | RX_TIME_QUAL                  | 1b    |                |
| RFEND_CFG0    | RFEND_CFG0_NOT_USED           | 0b    | RFEND_CFG0 = 0x09|
|               | CAL_EN_WAKE_UP_EN             | 0b    |                |
|               | TXOFF_MODE                    | 00b   |                |
|               | TERM_ON_BAD_PACKET            | 1b    |                |
|               | ANT_DIV_RX_TERM_CFG           | 001b  |                |
| AGC_CS_THR    | AGC_CS_THRESHOLD              | 0xF5  | AGC_CS_THR = 0xF5|
| AGC_CFG1      | AGC_SYNC_BEHAVIOUR            | 101b  | AGC_CFG1 = 0xA0|
|               | AGC_WIN_SIZE                  | 000b  |                |
|               | AGC_SETTLE_WAIT               | 0b    |                |
| AGC_CFG0      | AGC_HYST_LEVEL                | 11b   | AGC_CFG0 = 0xC3|
|               | AGC_SLEWRATE_LIMIT            | 00b   |                |
|               | RSSI_VALID_COUNT              | 00b   |                |
|               | AGC_ASK_DECAY                 | 11b   |                |
| PREAMBLE_CFG0 | PREAMBLE_CFG0_NOT_USED        | 0b    | PREAMBLE_CFG0 = 0x2A|
|               | PQT_EN                        | 1b    |                |
|               | PQT_VALID_TIMEOUT             | 0b    |                |
|               | PQT                           | 101b  |                |
| FIFO_CFG      | CRC_AUTOFLUSH                 | 1b    | FIFO_CFG = 0x80|
|               | FIFO_THR                      | 000000b |            |</p>
<table>
<thead>
<tr>
<th>Register Name</th>
<th>Register Field</th>
<th>Value</th>
<th>Register Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKT_CFG1</td>
<td>PKT_CFG1_NOT_USED</td>
<td>0b</td>
<td>PKT_CFG1 = 0x05</td>
</tr>
<tr>
<td></td>
<td>WHITE_DATA</td>
<td>0b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDR_CHECK_CFG</td>
<td>00b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CRC_CFG</td>
<td>01b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BYTE_SWAP_EN</td>
<td>0b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APPEND_STATUS</td>
<td>1b</td>
<td></td>
</tr>
<tr>
<td>PKT_CFG0</td>
<td>PKT_CFG0_RESERVED7</td>
<td>0b</td>
<td>PKT_CFG0 = 0x20</td>
</tr>
<tr>
<td></td>
<td>LENGTH_CONFIG</td>
<td>01b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PKT_BIT_LEN</td>
<td>000b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART_MODE_EN</td>
<td>0b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART_SWAP_EN</td>
<td>0b</td>
<td></td>
</tr>
<tr>
<td>PKT_LEN</td>
<td>PACKET_LENGTH</td>
<td>0x7D</td>
<td>PKT_LEN = 0x7D</td>
</tr>
<tr>
<td>IOMUX</td>
<td>GPIO2_ATRAN</td>
<td>0b</td>
<td>IOMUX = 0x13</td>
</tr>
<tr>
<td></td>
<td>GPIO2_INV</td>
<td>0b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GPIO2_CFG</td>
<td>010011b</td>
<td></td>
</tr>
<tr>
<td>MDMCFG1</td>
<td>CARRIER_SENSE_GATE</td>
<td>0</td>
<td>MDMCFG1 = 0x46</td>
</tr>
<tr>
<td></td>
<td>FIFO_EN</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MANCHESTER_EN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INVERT_DATA_EN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COLLISION_DETECT_EN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DVGA_GAIN</td>
<td>11b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SINGLE_ADC_EN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SYNC_CFG1</td>
<td>SYNC_CFG1_RESERVED7</td>
<td>0</td>
<td>SYNC_CFG1 = 0x0B</td>
</tr>
<tr>
<td></td>
<td>PQT_GATING_EN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC_CFG1_RESERVED5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC_THR</td>
<td>1011b</td>
<td></td>
</tr>
</tbody>
</table>
Example 3. RX Sniff Mode Settings (CC1120, 1.2 kbps, 50 kHz RX filter BW, RX config. 1)

```c
// RX filter BW = 50.000000
// Address config = No address check
// Packet length = 125
// Symbol rate = 1.2
// PA ramping = true
// Performance mode = High Performance
// Carrier frequency = 868.000000
// Bit rate = 1.2
// Packet bit length = 0
// Whitening = false
// Manchester enable = false
// Modulation format = 2-FSK
// Packet length mode = Variable
// Device address = 0
// TX power = 15
// Deviation = 20.019531
// Rx settings for CC1120
static const registerSetting_t preferredSettings[] = {
    {CC112X_IOCFG3, 0xB0},
    {CC112X_IOCFG2, 0x13},
    {CC112X_IOCFG1, 0xB0},
    {CC112X_IOCFG0, 0x06},
    {CC112X_SYNC_CFG1, 0x0B},
    {CC112X_DEVIATION_M, 0x48},
    {CC112X_MODCFG_DEV_E, 0x05},
    {CC112X_DCFILT_CFG, 0x1C},
    {CC112X_IQIC, 0x00},
    {CC112X_CHAN_BW, 0x04},
    {CC112X_MDMCFG0, 0x05},
    {CC112X_AGCCS_THR, 0xF5},
    {CC112X_AGCCFG1, 0xA0},
    {CC112X_SETTLING_CFG, 0x03},
    {CC112X_FS_CFG, 0x12},
    {CC112X_MOR_CFG0, 0x20},
    {CC112X_MOR_EVENT0_MSB, 0x02},
    {CC112X_MOR_EVENT0_LSB, 0x14},
    {CC112X_PKT_CFG0, 0x20},
    {CC112X_RFEND_CFG0, 0x09},
    {CC112X_PKT_LEN, 0x7D},
    {CC112X_FREQOFF_CFG, 0x22},
    {CC112X_FREQ2, 0x6C},
    {CC112X_FREQ1, 0x80},
    {CC112X_FS_DIG1, 0x00},
    {CC112X_FS_DIG0, 0x5F},
    {CC112X_FS_CALL, 0x40},
    {CC112X_FS_CAL0, 0x0E},
    {CC112X_FS_DIVTWO, 0x03},
    {CC112X_FS_DSM0, 0x33},
    {CC112X_FS_DVC0, 0x17},
    {CC112X_FS_PPD, 0x50},
    {CC112X_FS_PRE, 0x6E},
    {CC112X_FS_REG_DIV_CML, 0x14},
    {CC112X_FS_SPARSE, 0xAC},
    {CC112X_FS_VCO0, 0xB4},
    {CC112X_XOSC5, 0x0E},
    {CC112X_XOSC1, 0x3},
};
```
4.1.1.1 Measurements and Estimates (RX Config. 1)

Using the CC112x_RX_Sniff_Mode Excel sheet [9] to estimate the CS response time, you get ~540 μs (see Figure 7).

![Figure 7. CS Response Time (CC1120, 1.2 kbps, 50 kHz RX filter BW)](image)

Figure 7. CS Response Time (CC1120, 1.2 kbps, 50 kHz RX filter BW)

Figure 8 shows the current profile when running the CC1120 RX Sniff Mode code. The RX time is measured to be 540 μs, the same as the estimated CS response time found using the Excel sheet [9] (see Figure 7).

![Figure 8. Current Profile (CC1120, 1.2 kbps, 50 kHz RX filter BW)](image)
Figure 9 shows the measurement of the average current consumption when running the RX Sniff Mode (RX config. 1). The average current consumption (when there is no data on the air) is 0.905 mA (905 μV measured over a 1Ω resistor). This is almost the same as estimated in CC112x_RX_Sniff_Mode.xlsx [9] (see Figure 10).

![Figure 9. Average Current Consumption (CC1120, 1.2 kbps, 50 kHz RX Filter BW, RX config. 1)](image)

Figure 9. Average Current Consumption (CC1120, 1.2 kbps, 50 kHz RX Filter BW, RX config. 1)

![Figure 10. CC112x_RX_Sniff_Mode.xlsx Dashboard (RX config. 1)](image)

Figure 10. CC112x_RX_Sniff_Mode.xlsx Dashboard (RX config. 1)
4.1.2 RX Config. 2

Many existing protocols use 4 bytes of preamble and 4 bytes sync. When increasing the preamble from 3 (RX config. 1) to 4 bytes, the average current consumption is reduced from 0.904 mA to 0.646 mA as shown in Figure 11, as $t_{\text{Event0}}$ can be increased from 16.63 ms to 23.28 ms.

![Figure 11. CC112x_RX_Sniff_Mode.xlsx Dashboard (RX config. 2)](image)

4.1.3 RX Config. 3

Assume a system using 4 bytes of preamble and a 4 bytes long sync word. The CC112x (and CC120x) can be configured to look for an 11 bits sync word. This means that the receiver can sleep for part of the sync word as well (see Figure 12), increasing $t_{\text{Event0}}$ and reducing the current consumption.

![Figure 12. 4 Bytes Sync Word vs. 11 Bits Sync Word](image)

In this case, the effective preamble is 4 bytes + (32 – 11 bits) = 53 bits = 6.625 bytes and the average current consumption is estimated to be 0.369 mA (see Figure 13).

![Figure 13. CC112x_RX_Sniff_Mode.xlsx Dashboard (RX config. 3)](image)
4.2 **CC1200 Code Example**

The CC1200 code example [12] uses the 38.4 kbps typical settings from the SmartRF Studio [8] version 1.13.0 as a starting point (see Figure 14). The register settings will be altered to achieve a 1.5 mA average current consumption.

![Figure 14. Typical Settings (CC1200, 38.4 kbps, ETSI standard)](image-url)
The register settings obtained by using the code export feature is shown in Example 4.

Example 4. Code Export (CC1200, 38.4 kbps, ETSI standard)

```c
// RX filter BW = 104.166667
// Address config = No address check
// Packet length = 255
// Symbol rate = 38.4
// Carrier frequency = 867.999878
// Bit rate = 38.4
// Packet bit length = 0
// Whitening = false
// Manchester enable = false
// Modulation format = 2-GFSK
// Packet length mode = Variable
// Device address = 0
// Deviation = 19.989014
// Rf settings for CC1200
static const registerSetting_t preferredSettings[] = {
    {CC120X_IOCFG2, 0x06},
    {CC120X_SYNC_CFG1, 0xA9},
    {CC120X_MDCFG_DEV_E, 0x0B},
    {CC120X-prependable_CFG0, 0x8A},
    {CC120X_IQIC, 0xC8},
    {CC120X_CHAN_BW, 0x10},
    {CC120X_MDMCFG1, 0x42},
    {CC120X_MDMCFG0, 0x05},
    {CC120X_SYMBOL_RATE2, 0x8F},
    {CC120X_SYMBOL_RATE1, 0x75},
    {CC120X_SYMBOL_RATE0, 0x10},
    {CC120X_AG5_CFG, 0x27},
    {CC120X_AG5_CS_THR, 0xEE},
    {CC120X_AG5_CFG1, 0x11},
    {CC120X_AG5_CFG0, 0x94},
    {CC120X_FIFO_CFG, 0x00},
    {CC120X_FS_CFG, 0x12},
    {CC120X_PKT_CFG2, 0x0B},
    {CC120X_PKT_CFG1, 0x01},
    {CC120X_PKT_LEN, 0xFF},
    {CC120X_IF_MIX_CFG, 0x1C},
    {CC120X_TOC_CFG, 0x03},
    {CC120X_MDMCFG2, 0x02},
    {CC120X_FREQ2, 0x56},
    {CC120X_FREQ1, 0x56},
    {CC120X_FB_CFG, 0x00},
    {CC120X_FB_DIG1, 0x00},
    {CC120X_FB_DIG0, 0x00},
    {CC120X_FB_CAL1, 0x00},
    {CC120X_FB_CAL0, 0x00},
    {CC120X_FB_DVTO, 0x00},
    {CC120X_FB_DSM0, 0x17},
    {CC120X_FB_PFD, 0x00},
    {CC120X_FB_PRE, 0x05},
    {CC120X_FB_REG_DIV_CML, 0x1C},
    {CC120X_FB_SPARE, 0x94},
    {CC120X_FB_VCMO, 0x15},
    {CC120X_FB_VC0, 0x0E},
    {CC120X_FB_VCO, 0x09},
    {CC120X_XOSC5, 0x0E},
    {CC120X_XOSC1, 0x03},
};
```
AGC_WIN_SIZE, AGC_SETTLE_WAIT, and RSSI_VALID_COUNT were all set to 0 (see Section 3.9.1) and FS auto calibration was turned off (SETTLING_CFG.FS_AUTOCAL = 0). The register values were input to the CC120x_RX_Sniff_Mode Excel sheet [10] as shown in Figure 15. Figure 15 also shows that the number of preamble bytes is too low to be able to run the RX Sniff Mode. (1)

![Figure 15. CC120x_RX_Sniff_Mode.xlsx Dashboard (3 bytes preamble)](image)

To be able to run the RX Sniff Mode and achieve an average current consumption of ~1.5 mA, the number of preamble bytes transmitted must be increased to 24 bytes, as shown in Figure 16. It can easily be seen that reducing the current on the RX side means increased current consumption on the TX side.

![Figure 16. CC120x_RX_Sniff_Mode.xlsx Dashboard (24 bytes preamble)](image)

The sensitivity limit for the chosen RF settings is -110 dBm (see the CC1200 data sheet [6]). In the code example [12], the CS threshold is set 1 dB higher, at -109 dBm, to avoid too many wake-ups due to noise.

The RSSI offset is -81 dBm (see the CC120x user’s guide [2]). To get a CS threshold of -109 dBm, AGC_CS_THR must be set to 0xE4 (use Equation 11).

\[
CS\ Threshold = AGC\_CS\_THR + RSSI\ Offset \\
AGC\_CS\_THR = CS\ Threshold - RSSI\ Offset = -109 - (-81) = -28
\]

(11)

The rest of the relevant register fields were changed in the same manner as explained for the CC1120 example [11] (see Section 4.1) and the registers settings used by the CC1200 RX Sniff Mode code example [12] are shown in Example 5.

(1) A solution might be to use the RX Duty Cycle Mode as the RX duty cycle timer provides smaller timeouts on the time not spent in RX, but this application report does not go into details on this topic. The CC120x user’s guide [2] provides more info on this mode.
Example 5. RX Sniff Mode Settings (CC1200, 38.4 kbps, ETSI standard)

// RX filter BW = 104.166667
// Address config = No address check
// Packet length = 125
// Symbol rate = 38.4
// Carrier frequency = 867.999878
// Bit rate = 38.4
// Packet bit length = 0
// Whitening = false
// Manchester enable = false
// Modulation format = 2-GFSK
// Packet length mode = Variable
// Device address = 0
// Deviation = 19.989014
// Rf settings for CC1200
static const registerSetting_t preferredSettings[] = {
    {CC120X_IOCFG2, 0x13},
    {CC120X_IOCFG0, 0x06},
    {CC120X_SYNC_CFG1, 0xA9},
    {CC120X_MODCFG_DEV_E, 0x0B},
    {CC120X_PREAMBLE_CFG1, 0x30},
    {CC120X_PREAMBLE_CFG0, 0x8A},
    {CC120X_IQIC, 0xC8},
    {CC120X_CHAN_BW, 0x10},
    {CC120X_MDMCFG1, 0x42},
    {CC120X_MDMCFG0, 0x05},
    {CC120X_SYMBOL_RATE2, 0x8F},
    {CC120X_SYMBOL_RATE1, 0x75},
    {CC120X_SYMBOL_RATE0, 0x10},
    {CC120X_AGC_REF, 0x27},
    {CC120X_AGC_CS_THR, 0xE4},
    {CC120X_AG_CCFG1, 0x00},
    {CC120X_AG_CCFG0, 0x90},
    {CC120X_SETTLING_CFG, 0x03},
    {CC120X_FS_CFG, 0x12},
    {CC120X_MOR_CFG0, 0x20},
    {CC120X_MOR_EVENT0_LSB, 0xC3},
    {CC120X_PKT_CFG2, 0x00},
    {CC120X_PKT_CFG0, 0x09},
    {CC120X_PKT_LEN, 0x7D},
    {CC120X_IF_MIX_CFG, 0x1C},
    {CC120X_TOC_CFG, 0x03},
    {CC120X_MDMCFG2, 0x02},
    {CC120X_FREQ2, 0x56},
    {CC120X_FREQ1, 0x0C},
    {CC120X_FREQ0, 0x0C},
    {CC120X_RFEND_CFG0, 0x09},
    {CC120X_PKT_LEN, 0x7D},
    {CC120X_XOSC5, 0x0E},
    {CC120X_XOSC1, 0x33},
};
**Example 5. RX Sniff Mode Settings (CC1200, 38.4 kbps, ETSI standard) (continued)**

Figure 17 shows the current profile when running the CC1200 RX Sniff Mode code [12]. The average current consumption is measured to be 1.50 mA, very close to the estimated current consumption found using the 120x_RX_Sniff_Mode Excel sheet [10] (see Figure 16).

![Current Consumption Graph](image)

**Figure 17. Average Current Consumption (CC1200, 38.4 kbps, ETSI standard, 24 bytes preamble)**
5 Conclusion

This application report has shown which registers to configure when implementing the RX Sniff Mode and also provides two Excel sheets, which provide a good estimation of the average current consumption when using this mode. The code examples can be used as starting points when implementing the RX Sniff Mode in a real-life application.

From the examples one can see that the average current consumption decreases when the preamble length increases. For very long preamble sequences it is recommended to use SmartPreamble. This concept is described in details in [13].

6 References

1. CC112X/CC1175 Low-Power High Performance Sub-1 GHz RF Transceivers/Transmitter User’s Guide (SWRU295)
2. CC120x Low-Power High Performance Sub-1 GHz RF Transceivers User’s Guide (SWRU346)
3. CC1120 High Performance RF Transceiver for Narrowband Systems Data Sheet (SWRS112)
4. CC1121 High Performance Low Power RF Transceiver Data Sheet (SWRS111)
5. CC1125 Ultra-High Performance RF Narrowband Transceiver Data Sheet (SWRS120)
6. CC1200 Low Power, High Performance RF Transceiver Data Sheet (SWRS123)
7. CC112x/CC1175 High Performance RF Transceiver/Transmitter for Narrowband Systems Errata Note (SWRZ039)
8. SmartRF Studio (SWRC176)
9. CC112x_RX_Sniff_Mode.xlsx (SWRA428)
10. CC120x_RX_Sniff_Mode.xlsx (SWRA428)
11. CC112x Software Examples (SWRC253)
12. CC120x Software Examples (SWRC274)
13. CC112x/CC120x RX Sniff Mode With Smart Preamble (SWRA438)
Appendix A Revision History

This document has been revised from SWRA428 to SWRA428A because of the following technical change(s).

Table 6. SPNU428A Revisions

<table>
<thead>
<tr>
<th>Location</th>
<th>Additions, Deletes, and Edits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3</td>
<td>Added two registers (MDMCFG1 and SYNC_CFGx)</td>
</tr>
<tr>
<td>Section 3.2</td>
<td>Added info on how to determine $t_{\text{Event0}}$ when using PQT termination. Also added info on how the RC oscillator tolerance should be accounted for.</td>
</tr>
<tr>
<td>Section 3.7</td>
<td>Added info on $t_{\text{Event0}}$ when PQT termination is enabled</td>
</tr>
<tr>
<td>Section 3.9</td>
<td>Added info saying that $t_{\text{Event0}}$ is shorter when PQT termination is enabled compared to when using CS termination.</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Added case C showing the current consumption for the radio when MCU is used to terminate RX (and CS termination is enabled on the radio).</td>
</tr>
<tr>
<td>Section 3.9.2</td>
<td>Added explanation of PQT response time and reference to user’s guides.</td>
</tr>
<tr>
<td>Section 4.1</td>
<td>SmartRF Studio version changed from 1.12.0 to 1.13.0.</td>
</tr>
<tr>
<td>Section 4.1.1.1</td>
<td>New value for EVENT0 since the RC oscillator tolerance is accounted for in the equation.</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Added MDMCFG1 and SYNC_CFG1</td>
</tr>
<tr>
<td>Example 3</td>
<td>Changed value of WOR_EVENT0_LSB</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Updated with new measurements</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Updated due to new revision of excel sheet</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Updated due to new revision of excel sheet</td>
</tr>
<tr>
<td>Section 4.1.2</td>
<td>Changes numbers in the text related to Figure 11</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Updated with new measurements</td>
</tr>
<tr>
<td>Example 4</td>
<td>Changed value FS_REG_DIV_CML</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Updated due to new revision of excel sheet</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Updated due to new revision of excel sheet</td>
</tr>
<tr>
<td>Example 5</td>
<td>Changed value FS_REG_DIV_CML</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Updated with new measurements</td>
</tr>
<tr>
<td>Section 5</td>
<td>Added reference to [13]</td>
</tr>
</tbody>
</table>
Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have not been so designated is solely at the Buyer’s risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have not been so designated is solely at the Buyer’s risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have not been so designated is solely at the Buyer’s risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have not been so designated is solely at the Buyer’s risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.