Application Report
SWRA438–October 2013

CC112x/CC120x RX Sniff Mode With SmartPreamble

Niklas Norin and Siri Johnsrud

ABSTRACT

This application report shows an innovative way to use the DualSync functionality of the CC112x/CC120x to handle a new kind of preamble. By using what is called SmartPreamble, the receiver is able to reliably detect exactly where it woke up in the preamble and use the eWOR timer to go back to sleep, and then wake up right before the sync word.

Contents

List of Figures

1 Introduction

Using the powerful RX Sniff Mode functionality of TI’s Performance Line Sub-1 GHz RF family may dramatically decrease the average power consumption of a receiver by using long preamble sequences. Long preamble allows the receiver to sleep for longer periods of time and still reliably receive the data payload. When the radio wakes up and goes into RX it uses CS or PQT termination to detect if there is a signal. If a signal is found, the radio waits in RX until the sync word is detected.

While using long preamble sequences allows the radio to sleep longer, it also means that if it wakes up early in the preamble it has to stay a long time in RX before sync is found. SmartPreamble allows the radio to sleep while waiting for sync, which greatly reduces the current consumption.

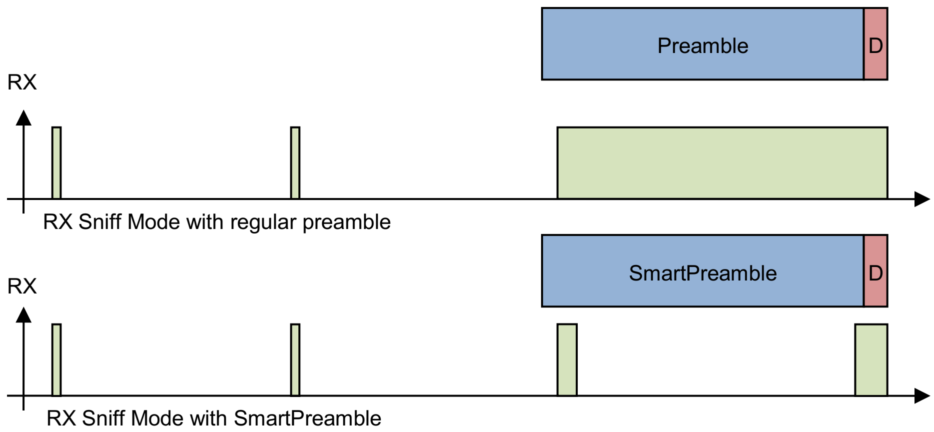

Table 1. Abbreviations

CS	Carrier Sense
EB	Evaluation Board
EM/EVM	Evaluation Module
GPIO	General-Purpose Input/Output
PQT	Preamble Quality Threshold
RX	Receive Mode

2 Prerequisites

The concepts involved are thoroughly documented in *CC112x/CC120x RX Sniff Mode* (SWRA428). It is assumed that the reader is familiar with [1] as well as the device-specific user's guides corresponding to the used radio transceiver [3], [4].

There is also a code example [2] that demonstrates the RX Sniff Mode with SmartPreamble on a TrxEB and CC1200EM. While it is not mandatory to have read the code, some figures, numbers, and code snippets are taken from that example.

3 RX Sniff Mode With SmartPreamble

When using the RX Sniff Mode with a long preamble sequence, the radio wakes up at a rate given by the preamble length to check if there is a signal on the air, either using CS or PQT detection and termination. When a signal is detected, the radio stays in RX until sync is found and the payload is received, as illustrated in Figure 1 (Case A).

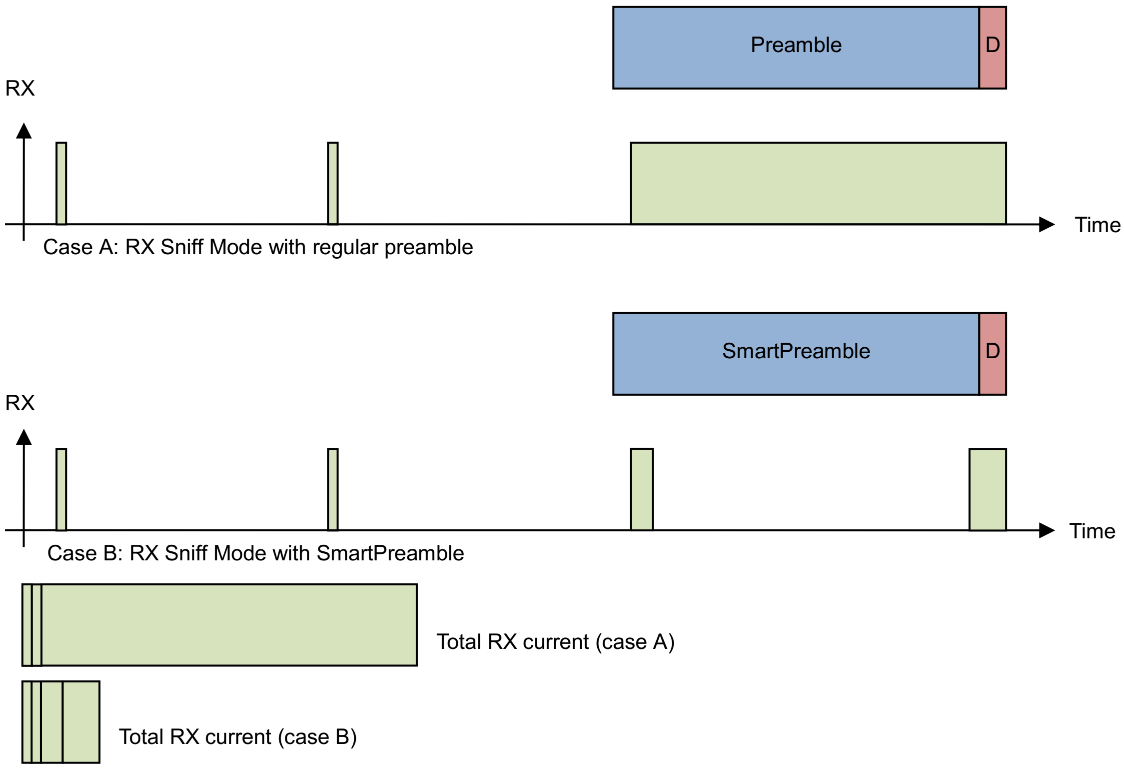

Figure 1. RX Sniff Mode With Regular Preamble and SmartPreamble

The time spent in RX while receiving the rest of the preamble may be considerable when using very long preambles, thus wasting a lot of energy. Since every byte of the regular preamble is the same, an alternating sequence of zeroes and ones, there is no way to determine where the radio woke up inside the preamble.

When using the SmartPreamble, the radio wakes up at set periods in the exact same way as with the RX Sniff Mode with a regular preamble. The difference is that when the radio wakes up inside the SmartPreamble, it has a way to determine where it is and how long it will be until the payload is on the air. This is done by making the transmitter encode timestamp information in the preamble. In fact, the preamble itself consists of a sequence of packages with a sync word that is different from the sync word in the payload packet.

Because of the DualSync support, there is no need for the MCU to analyze the payload data to determine what kind of packet was received. The radio will automatically detect packets with any of the two configured sync words and discard any others. When the radio signals that a packet has been received with the correct CRC, the MCU simply reads the general-purpose input/output (GPIO) pin configured as `SYNC_LOW0_HIGH1` to determine if it is a timestamp packet or a payload packet. If a timestamp packet is detected, instead of having to be awake for the entire length of the preamble, both the radio and the MCU can go back to sleep in microseconds waking up just in time for the payload (Case B).

Increasing the preamble to hundreds or thousands of bytes will increase the average current consumption of the transmitter. The SmartPreamble solution is more suitable for systems where saving power while receiving is far more important than saving power on the transmitter side.

3.1 Initial Register Configuration

The way that eWOR registers and most other registers are configured initially, when using SmartPreamble, is very similar to using a regular preamble. The only changes that need to be made to the radio registers exported from SmartRF™ Studio [5] software, in addition to the ones already mentioned in [1], is to enable DualSync, set the `SYNC1` and `SYNC0` registers, and make both `SYNC_LOW0_HIGH1` and `WOR_EVENT1` signals available on the GPIO's of the radio. For more details regarding the exact register configuration, see the customRegisterSettings array in the *cc120x_smartpreamble_rx.c* file [2].

The code example accompanying this document is written specifically for the CC120x, and while most of the register settings are similar or exactly the same as on the CC112x, there are some differences such as the sync related registers. If porting the example to the CC112x, see the CC112x User's Guide [3] and SmartRF Studio [5].

3.2 SmartPreamble Structure

The preamble used in RX Sniff Mode, which is described in [1], consists of a sequence of alternating zeros and ones and are inserted automatically by the transmitter. Preamble is needed to allow the AGC to settle before receiving the sync word.

When using SmartPreamble, the preamble not only consists of alternating zeros and ones but a sequence of packets instead. Each packet that constitutes the SmartPreamble, called a Timestamp packet, is a proper packet in itself with one byte preamble, a special sync word, length byte, timestamp, and CRC. The SmartPreamble consists of a sequence of Timestamp packets that are sent back-to-back. The structure of the SmartPreamble is shown in Figure 2. The payload packet containing the real data follows the SmartPreamble.

The most important part of the SmartPreamble is the timestamp. The timestamp is one byte and any value (0 - 255) is valid. The timestamp acts as a countdown towards the payload packet, therefore, the last timestamp packet will always have timestamp zero.

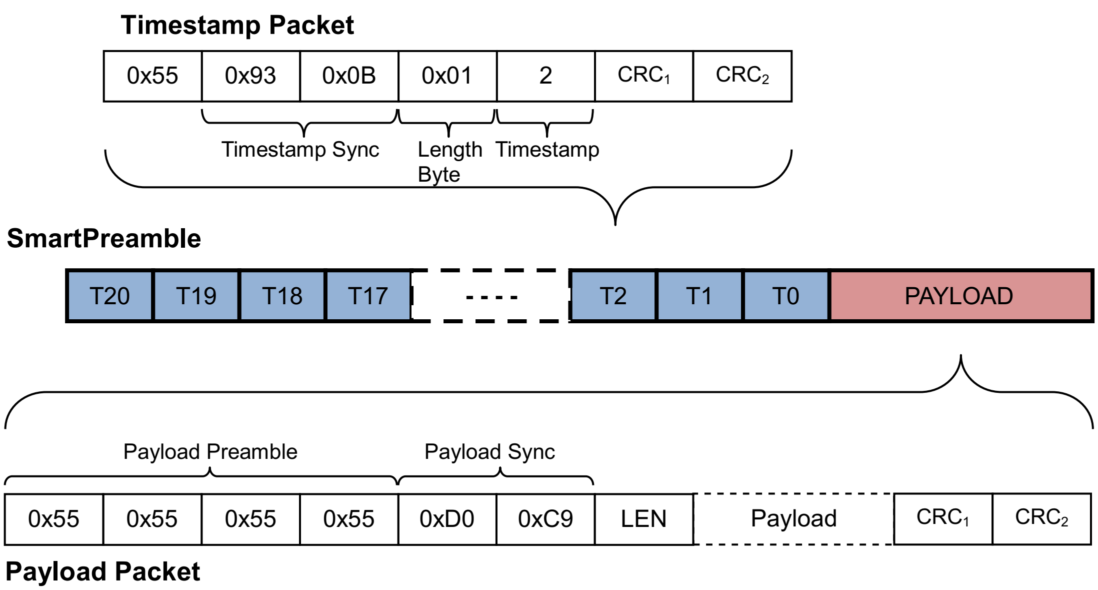

Figure 2. SmartPreamble Structure

3.3 Determining the Timing

The eWOR timer is used for two different purposes when using the SmartPreamble. It is used initially to detect if there is a signal on the air. This is called Search Mode and is the same as RX Sniff Mode. When a signal is found and the radio has received a timestamp packet, the eWOR timer is reset (`SWORRST`) and re-configured, using a lookup table, to go back to sleep and wake up right before the payload packet. This is called Look-up Mode.

3.3.1 Search Mode

In Search Mode, t_{Event0} is determined by the length of the SmartPreamble in the same manner as when using the ordinary Rx Sniff Mode, except there is no need to account for AGC settling as this can be done during the preamble of the payload packet. The `WOR_CFG1.WOR_EVENT0_MSB` and `WOR_EVENT0_LSB` is set according to Equation 1. For details on how to program t_{Event0}, see [1].

$$t_{Event0} = \frac{\#\text{ of Preamble Bits}}{\text{Data Rate [bps]}} \text{ [s]}$$

$$t_{Event0} = \frac{\#\text{ of Timestamp Packets} * 7 * 8}{\text{Data Rate [bps]}} \text{ [s]} \quad (1)$$

3.3.2 Look-Up Mode

As soon as the radio wakes up and receives a timestamp packet, and the timestamp itself has been read out, EVENT0 needs to be reset and reconfigured. The new value is calculated in such a way that the next time the radio wakes up it will be at the beginning of the preamble bytes of the payload packet.

An overview of the timing periods relevant for calculating the wakeup time is shown in Figure 3.

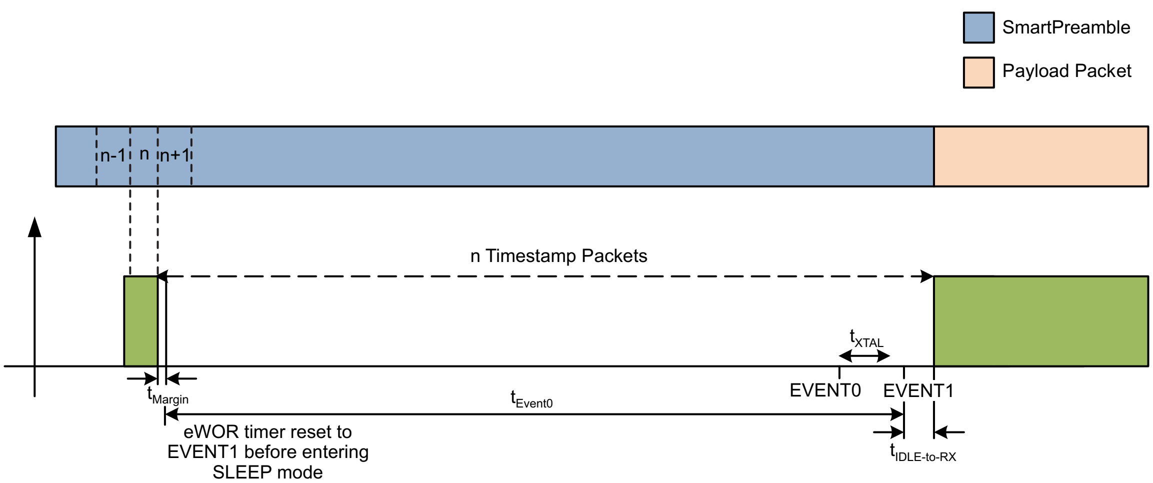

Figure 3. eWOR Timing

When calculating the time it takes when the radio enters SLEEP mode (after the timestamp packet was received) until it enters RX mode again, both the crystal start-up time, t_{XTAL}, as well as the time it takes to enter RX mode, $t_{IDLE\text{-}to\text{-}RX}$, needs to be taken into consideration. RX mode is entered when the crystal is stable and EVENT1 has occurred. Since the crystal start-up time can vary and it is important for the timing to be deterministic, t_{Event1} is set to be larger than t_{XTAL} (that means that the internal SRX strobe will be issued at EVENT1).

The time between two EVENT0's (or two EVENT1's) is called t_{Event0}. The eWOR timer is reset right before going back to SLEEP. This means that t_{Event0} can be calculated as shown in Equation 2.

$$t_{Event0} = \frac{\text{Timestamp } \# * 7 * 8}{\text{Data Rate [bps]}} - t_{IDLE\text{-}to\text{-}RX} - t_{Margin} \text{ [s]} \quad (2)$$

The margin, t_{Margin}, used in Equation 2 represents the time it takes for the MCU to handle a timestamp. In a real implementation, this is generally the time it takes from when the MCU receives the `CRC_OK` interrupt until the eWOR timer has been reset. The time margin from the code example [2] is approximately 273 μs, and the measurement can be seen in Figure 4, but this is entirely dependent on the implementation on the MCU and must be measured. Note that the delay between the last two strobes is an intentional delay between `SWORRST` and `SWOR`. The reason is that it takes up to two clock periods ($2/f_{XOSC}$) for the eWOR timer to reset; waiting two clock periods assures this has happened before entering SLEEP mode. Also note that t_{Event0} in Equation 2 does not take the 0.1 % RC oscillator tolerance into account, something that needs to be done when programming t_{Event0}. For details, see [1].

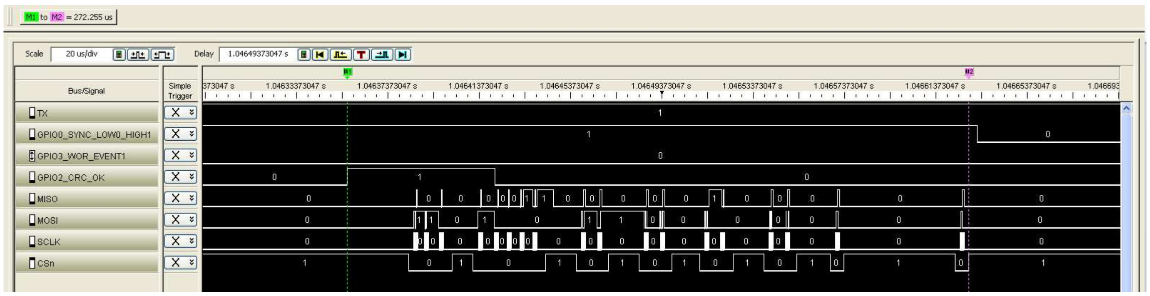

Figure 4. Example t_{Margin}

3.3.2.1 *Last Timestamp Packet*

If the receiver wakes up in the very last packet, with timestamp zero, then the next packet is the payload packet. Depending on the symbol rate and the payload preamble length, the receiver will not have time to go back to sleep and wake up again before the payload's sync word, and will go directly into RX state.

4 Implementation

All custom register settings needed to run the SmartPreamble transmitter and receiver may be found in the *customRegisterSettings* array in cc120x_smartpreamble_tx.c and cc120x_smartpreamble_rx.c, respectively [2]. The custom register settings in the code example are set up in such a way that it can be used together with recommended register settings exported from SmartRF™ Studio [5].

4.1 *Transmitter*

The transmitter software design is almost identical to the transmitter implemented in the infinite packet length mode example for the CC112x [6]. The transmitter uses infinite packet length mode combined with fixed packet length mode to send long packets. The main difference is that, instead of sending a sequence of the same preamble byte, a SmartPreamble is generated instead.

Example 1. SmartPreamble Generation

```
for (i = 0; i < NUM_TIMESTAMP_PACKETS; i++) {
     checksum = CRC_INIT;
     checksum = calcCRC (0x01, checksum) ;
     checksum = calcCRC ((NUM_TIMESTAMP_PACKETS – 1) – i, checksum);
     txBuffer[(i * 7)]         = 0x55;
     txBuffer[(i * 7) + 1]     = 0x93;
     txBuffer[(i * 7) + 2]     = 0x0B;
     txBuffer[(i * 7) + 3]     = 0x01;
     txBuffer[(i * 7) + 4]     = (NUM_TIMESTAMP_PACKETS – 1) – i;
     txBuffer[(i * 7) + 5]     = (uint8) (checksum >> 8 );
     txBuffer[(i * 7) + 6]     = (uint8) (checksum & 0x00FF);
```

Creating the SmartPreamble is straightforward as shown in Example 1. This loop creates as many timestamp packages as has been specified by the `NUM_TIMESTAMP_PACKETS` define. The structure of the timestamp packet should be recognized from Figure 2. Since no part of the SmartPreamble will have to be changed in between transmissions it may be generated once on startup, or hardcoded, and then reused.

The CRC calculation, normally done automatically by the radio, is a software implementation of CRC-16/CCITT and may be found in the code example [2].

4.2 Receiver

The SmartPreamble receiver is slightly more complex when compared to the regular RX Sniff Mode receiver. When using RX Sniff Mode, and a regular preamble, the radio will simply stay in RX if either a carrier or preamble is detected, depending on the configuration. As soon as the packet is fully received it can be handled. When using a SmartPreamble, the reception of a packet is normally divided into a two-step process, when waking up inside a timestamp. An exception to this is if the receiver wakes up directly in the preamble of the real payload packet, then the packet will simply be received in one step as with the regular RX Sniff Mode. Section 4.2.1 and Section 4.2.2 describe in detail how to implement a SmartPreamble receiver.

4.2.1 Signals

Three signals, `WOR_EVENT1`, `SYNC_LOW0_HIGH1` and `CRC_OK` need to be mapped to the GPIO's of the radio and made available on the MCU. `CRC_OK` is used to indicate that a packet has been received, both in the case of a timestamp and payload packet. `SYNC_LOW0_HIGH1` is then used to identify which sync word is found (timestamp or payload). In the case when a timestamp packet is received and the application waits for a payload packet, the `WOR_EVENT1` signal is used to indicate the start of a timeout period equal to the time it takes to receive a packet.

Something to keep in mind when using the radio GPIO signals is that `GPIO0` - `GPIO3` are normally hardwired to either high or low when the radio is in `SLEEP` state. The state of the signal in SLEEP is also dependent on which signal is assigned to that specific GPIO. CC112x and CC120x do not handle or implement this in the same manner. For more information, see the *GPIO* section of user's guides [3], [4].

4.2.2 Program Flow

As has been pointed out earlier, the reception of a payload using a SmartPreamble is normally a two-step process involving close interaction between the radio and the MCU. In this section, this process is described in detail. To easier understand the program flow see Figure 5, which illustrates the implementation using a flow chart and also the code example [2]. Note that though the radio and MCU flowcharts are shown side by side in Example 1 they are in no way synchronized from top to bottom.

When the MCU is first started, it initializes the radio with the static register settings needed for RX Sniff Mode with SmartPreamble. Next, t_{Event0} is generated both for Search Mode and for Look-Up Mode before the radio starts searching for the first valid SmartPreamble using RX Sniff Mode.

The MCU then waits until the `CRC_OK` signal is asserted to indicate the reception of either a payload packet or, most likely, a timestamp packet. To decide which kind of packet is received, the MCU reads the `SYNC_LOW0_HIGH1` signal. If it happened to wake up directly into the payload packet it reads out the data, let the application handle it, and goes back to wait for the next packet.

If the MCU wakes up and detects that it received a timestamp packet, then it enters the Timestamp processing part, shown in Example 1. It will read out the timestamp and check if it is the last timestamp, number zero. If it is, then it will go directly to RX. This is due to the fact that the time left until the payload packet start is shorter than it takes to go to sleep and wake back up again. If it is not the last timestamp packet, the application will enter Look-Up Mode. The MCU then waits for the `WOR_EVENT1` signal instead of `CRC_OK` because there is always a chance that the payload is corrupted in some way (CRC check fails) or the sync word of the payload packet is not detected by the radio. In that case, the MCU will timeout and go back to Search Mode. Note that the timeout has to be large enough to allow the longest used payload to be received at the current symbol rate.

After reading out the payload data, the MCU and radio goes back into Search Mode and the entire process starts over again.

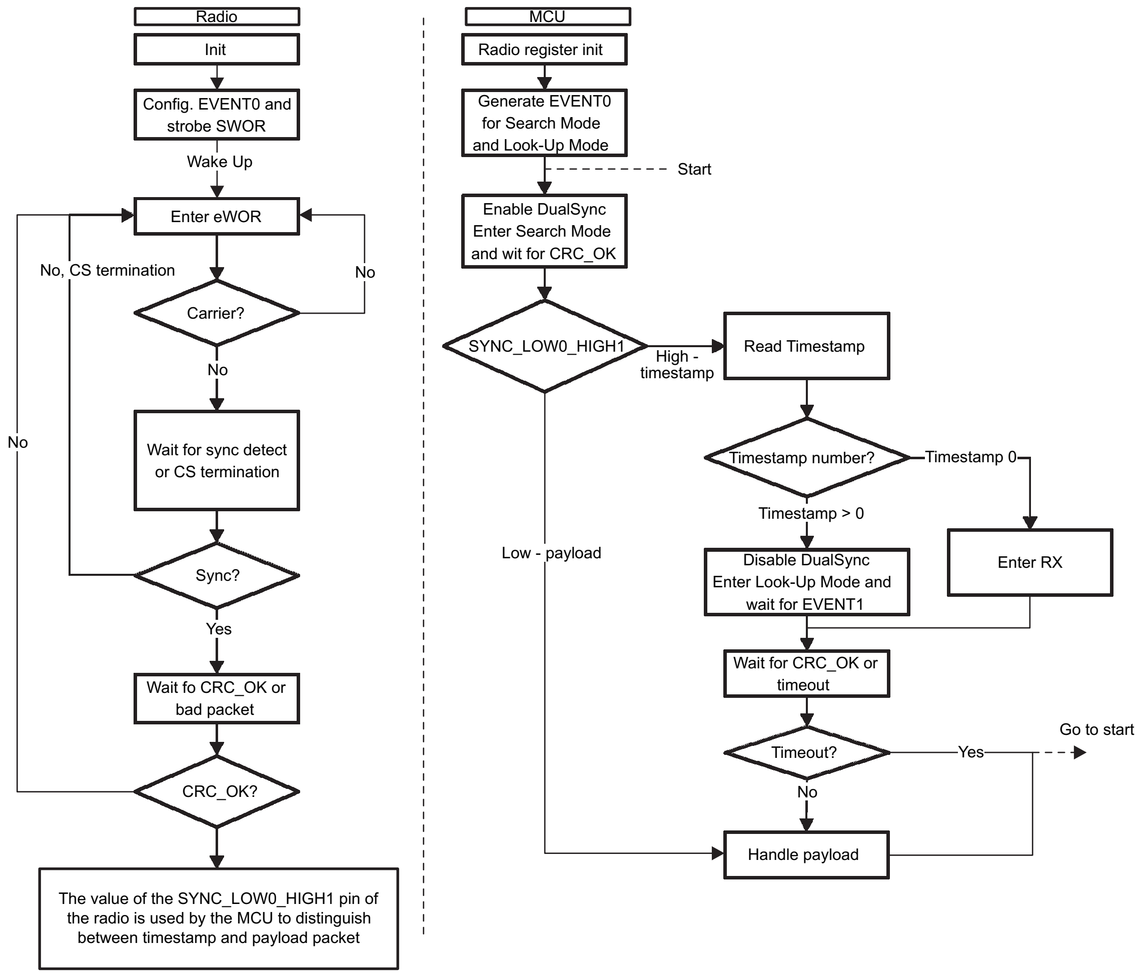

Figure 5. Radio and MCU Flowchart

5 Power Consumption

In the code example [2], a SmartPreamble consisting of 256 timestamp packets, each 7 bytes long, is implemented. Using the CC120x_RX_Sniff_Mode.xlsb Excel Sheet from [1], the average current consumption when looking for packets is as low as 20 µA (see Figure 6).

1	**Input Parameters:**	
2	Number of Preamble Bytes	1792
3	SYMBOL_RATE2	8F
4	SYMBOL_RATE1	75
5	SYMBOL_RATE0	10
6	Data Rate [sps]	38400
7	f_{XOSC} [MHz]	40
8	f_{RCOSC} [kHz]	40
9	MDMCFG2	2
10	MDMCFG0	5
11	IQIC	C8
12	SYNC_CFG0	3
13	CHAN_BW	10
14	DCFILT_CFG	4C
15	IF_MIX_CFG	1C
16	AGC_CFG1	0
17	AGC_CFG0	90
18	PREAMBLE_CFG0	8A
19	Termination Based on:	CS
20		
21	Average Current Consumption	0,020 mA
22	t_{Event0}	372,85 ms
23	EVENT0 (hex)	3A42

Figure 6. Average Current Consumption (1792 bytes SmartPreamble @ 38.4 kbps)

6 Conclusion

This application report shows how RX Sniff Mode with SmartPreamble, in a very simple and reliable way, may decrease the average current consumption of a receiver considerably by using TI's Performance Line Sub-1 GHz RF family. RX Sniff Mode with SmartPreamble utilizes the fact that, by increasing the length of the preamble and embedding information therein, it is possible for the receiver to sleep longer and act smarter when waking up. The SmartPreamble solution may be used to decrease the average current consumption of the receiver by orders of magnitude while still retaining a high data rate for transmissions.

7 References

1. *CC112x/CC120x RX Sniff Mode* (SWRA428)
2. *CC120X Software Examples* (SWRC274)
3. *CC112x/CC1175 Low-Power High Performance Sub-1 GHz RF Transceivers User's Guide* (SWRU295)
4. *CC120X Low-Power High Performance Sub-1 GHz RF Transceivers User's Guide* (SWRU346)
5. *SmartRF Studio 7 v1.14.0* (SWRC176)
6. *CC112x Software Examples* (SWRC253)

IMPORTANT NOTICE

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.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265