

# ***Asynchronous channel hopping system for FCC 15.247 compliance***

Thomas Almholt

## **ABSTRACT**

This application report shows how to use the CC112x and CC12xx series of transceivers in a Frequency Hopping Spread Spectrum system. This allows for transmit powers up to 1 W and utilization of narrow channels for optimum range at the same time.

Project collateral discussed in this application report can be downloaded from the following URL:  
<http://www.ti.com/lit/zip/swrc253>.

## **Contents**

1	Introduction .....	2
2	Review of Frequency Hopping Requirements per FCC 15.247 .....	2
3	Time Synchronized Channel Hopping versus Simple Asynchronous Channel Hopping .....	2
4	Algorithm for Implementing Simple Asynchronous Channel Hopping .....	7
5	Review of System Configuration Parameters .....	10
6	Using the Frequency Hopping System .....	13
7	System Test.....	15
8	References .....	17

## **List of Figures**

1	10 Channel Simple Asynchronous Channel Hopping Using a 2 ms Hop Time .....	3
2	Frequency Scanning When Using Pre-Calibration .....	5
3	Frequency Scanning When Using Min-Calibration .....	5
4	Flow Graph for Transmitting Node .....	7
5	Spectrum Analysis of Pseudo Random Hopping Sequence of a Narrow Band Hopping Transmitter.....	8
6	Flowchart of Interrupts Service Routine Implementing the RX Scanning Functionality .....	9
7	Flowchart of Main Loop of Demonstration Software.....	10
8	LCD: Showing the Information During Transmission of Packets .....	13
9	Measurement of the RF Burst Duration (Top) and “max-hold” Over 20 Seconds of All 50 Channels (Bottom) .....	14
10	LCD, Showing the Information Provided During Reception of Data.....	15
11	Results of Range Test, Plots by Daftlogic .....	16

## **List of Tables**

1	CC1120-Based Asynchronous Fast Frequency Hopping System Using @ 1.2 kbps and 25 kHz RX Channel Bandwidth.....	3
2	CS Response Time (CC1120, 1.2 kbps, 25 kHz RX filter BW) With Settings Taken From SmartRF Studio ...	4
3	Example of Time Consuming Elements in a Fast Frequency Hopping System Using @ 1.2 kbps and 12.5 kHz RX Channel Bandwidth.....	6
4	Narrow Band 2FSK Preamble Configurations Options .....	6

## Trademarks

MSP430, SmartRF are trademarks of Texas Instruments.  
All other trademarks are the property of their respective owners.

## 1 Introduction

FCC Section 15.249 restricts the fundamental radiated power to 50 mV/m at 3 meters distance (approximately -1 dBm EIRP) in the US 902-928 MHz ISM band. FCC CRF Section 15.247 specifies requirements that allow for up to 1 W transmit output power. High output power can be used if the system employs the Frequency Hopping Spread Spectrum (FHSS) or uses a digital modulation technique that gives a 6 dB bandwidth of minimum 500 kHz [1].

The low-duty cycle system is assumed in this application report, like home security sensors, gas and water meters systems, and other long-range low-data rate systems.

The design is based on the fast settling time of the frequency synthesizer in the transceiver to enable fast frequency scanning used to implement a simple asynchronous channel hopping system.

## 2 Review of Frequency Hopping Requirements per FCC 15.247

The FCC 15.247 defines the concept of frequency hopping for the operation of unlicensed wireless devices in the 902-928MHz ISM band for regions [1]. Included here are the two key excerpts from the FCC regulatory document:

- *Frequency hopping systems shall have hopping channel carrier frequencies separated by a minimum of 25 kHz or the 20 dB bandwidth of the hopping channel, whichever is greater. The system shall hop to channel frequencies that are selected at the system hopping rate from a pseudo randomly ordered list of hopping frequencies. Each frequency must be used equally on the average by each transmitter.*
- *For frequency hopping systems operating in the 902–928 MHz band: if the 20 dB bandwidth of the hopping channel is less than 250 kHz, the system shall use at least 50 hopping frequencies and the average time of occupancy on any frequency shall not be greater than 0.4 seconds within a 20 second period; if the 20 dB bandwidth of the hopping channel is 250 kHz or greater, the system shall use at least 25 hopping frequencies and the average time of occupancy on any frequency shall not be greater than 0.4 seconds within a 10 second period. The maximum allowed 20 dB bandwidth of the hopping channel is 500 kHz.*

By reviewing the statements in the regulatory document, the system needs to have the following for narrowband operation in the 902-928MHz band:

- 50 hopping channels
- 400 ms dwell time on each channel in a 20 second period
- 25 kHz minimum channel separation

This defines the system to have 50 channels and hopping at least every 400 ms and becomes the basis for the system requirements in [Section 3](#).

## 3 Time Synchronized Channel Hopping versus Simple Asynchronous Channel Hopping

In a typical Time Synchronized Channel hopping system, all the nodes in the system are synchronized using tight timing control that may require additional hardware in the form of RTC crystals and additional overhead in the form of special synchronization information to be exchanged between the nodes. For low-duty cycle, the overhead of maintaining a frequency hopping schedule increases the overall system power consumption and, in some cases, dominate the power consumption.

This section presents the concept on how the receiver and transmitters do not need to be synchronized in time or channel. This is achieved by letting the receiver perform fast frequency scanning of all channels used in the system and requiring that the transmitted burst contains a long enough preamble enabling the receiver to scan all channels at least once. The long preambles contain no data and enables the receiver to find active transmitter bursts of data before data loss occurs, which is called Simple Asynchronous Channel hopping system.

This requires a longer than normal preamble to work, but is considered an acceptable overhead. The length of the preamble becomes a compromise on how fast the receiver can hop and to what level of sophistication the receiver uses to determine whether to stay on a channel or proceed to the next.

Figure 1 depicts a 10 Simple Asynchronous Channel hopping and how it works by scanning each of the available channels until a preamble has been found. Then, it focuses on a channel until the entire packet has been received or a timeout occurs.

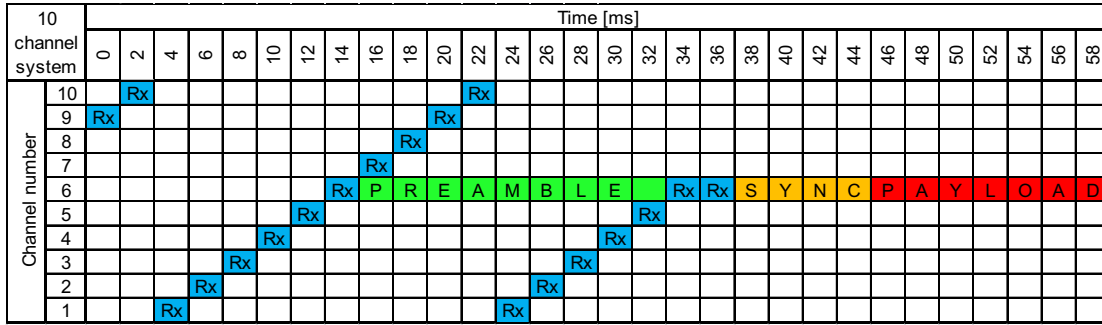


Figure 1. 10 Channel Simple Asynchronous Channel Hopping Using a 2 ms Hop Time

The blue “Rx” in Figure 1 indicates the receiver is performing a “sniff” operation on that specific channel and time. Green is the “preamble” of the transmitter’s data burst, yellow is the sync words and red is payload. For this system to work, the receiver needs to be fast enough to scan all channels and afford enough time to turn around to receive the packet. Therefore, the transmitted preamble time must be greater than scan\_time \* num\_channels.

The main time consuming components are detailed in Table 1; a compromise between scanning time and the PLL calibration method is shown.

In this configuration, the default and the most flexible PLL calibration method is called “pre-calibration”. the receiver is calibrated at each of the 50 channels; this calibration data is stored in memory and downloaded to the transceiver as needed.

The faster method is called “minimum calibration” and, in this configuration, the receiver is only calibrated at the center frequency of the entire band. This information is re-used for all 50 channels. This method only works for a relatively narrow band of operations (typically less than ±2MHz) from the calibrated frequency, but it saves time as the calibration data is not downloaded for each channel.

### 3.1 Timing Requirements of a Simple Asynchronous Channel Hopping System Using 25 kHz RX BW on a CC1120

Table 1 details the time consuming portions of a fast frequency scanning algorithm used to implement the Simple Asynchronous Channel hopping scheme. This section details the timing of a system operating with 25 kHz RX BW.

Table 1. CC1120-Based Asynchronous Fast Frequency Hopping System Using @ 1.2 kbps and 25 kHz RX Channel Bandwidth

1.2 kbps 25 kHz RX BW	Time With Pre-Calibration [µs]	Time With Minimum Calibration [µs]	Comments
RX – RX turnaround	50 µs	50 µs	Reset the RX chain (data sheet value)
RSSI VALID	990 µs	990 µs	CS response time (see Table 2)
SPI overhead	60 µs	140 µs	SPI traffic to enable reading of RSSI data, change RX channel, re-enable RX mode and optionally download 3 calibration values (compare Figure 2 and Figure 3)
Total	1100 µs	1180 µs	

The RX to RX turn-around is mostly determined by the overhead of the serial peripheral interface (SPI) port as it downloads new frequency variables, strobes the RX commands, and so forth.

The most significant amount of time is consumed waiting for the RSSI to become value, which is the same calculation as was done in the *CC112x/CC120x RX Sniff Mode* [10]. The same XLS document was used to calculate the “CS response” of the used parameters and measure exactly the same value. [Table 2](#) shows a screen grab of the excel documents showing the result of 990.2  $\mu$ s.

**Table 2. CS Response Time (CC1120, 1.2 kbps, 25 kHz RX filter BW) With Settings Taken From SmartRF Studio [9]**

Settings	
$f_{xosc}$ [MHz]	32
DCFILT_CFG.DCFILT_BW	4
DCFILT_CFG.DCFILT_FREEZE_COEFF	0
MDMCFG1.CARRIER_SENSE_GATE	0
CHAN_BW.BB_CIC_DECFACT	8
CHAN_BW_ADC_CIC_DECFACT	0
Decimation factor	20
CHAN_BW_CHFILT_BYPASS	0
AGC_CFG1.AGC_WIN_SIZE	0
AGC_CFG1.SETTLE_WAIT	0
AGC_CFG0.RSSI_VALID_CNT	0
# of AGC_UPDATE pulses	1
Results	
D0 [ $\mu$ s]	12.31 $\mu$ s
D1 [ $\mu$ s]	77.50 $\mu$ s
D2 [ $\mu$ s]	77.50 $\mu$ s
D3 [ $\mu$ s]	157.50 $\mu$ s
D4 [ $\mu$ s]	43.75 $\mu$ s
D5 [ $\mu$ s]	340.00 $\mu$ s
D6 [ $\mu$ s]	7.50 $\mu$ s
<b>T0 = D0 + D1 + D3 + D5</b>	587.31 $\mu$ s
T1	240.00 $\mu$ s
T2	81.44 $\mu$ s
C2 Response Time (# of AGC_UPDATE pulses is known)	990.19 $\mu$ s
Max CS Response Time	1230.19 $\mu$ s

With these two values, most of the time consumed during the hopping process was explained; the remaining time is the SPI overhead. In [Figure 2](#) and [Figure 3](#), the details of the timing of the implemented code running on an 8 MHz MSP430™ utilizing a 4 MHz SPI speed is shown.

The cyan trace is the chip-select line. Looking at that line, the additional register writes required to download the pre-calibrated PLL values becomes apparent, and the time it takes to writing to these registers is apparent.

The dark blue trace is the RSSI valid signal as defined in the device-specific user's guide of the CC112x or CC1200 series devices. [\[2\]](#), [\[3\]](#).

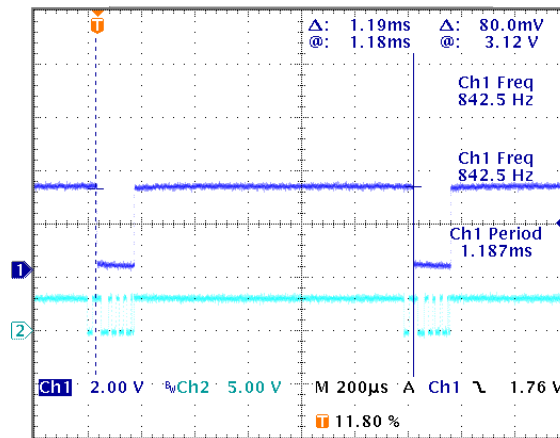


Figure 2. Frequency Scanning When Using Pre-Calibration

**NOTE:** The extra chip select transitions for every hop. (The dark blue is the RSSI valid indicator signal and the light blue is the chip select line.)

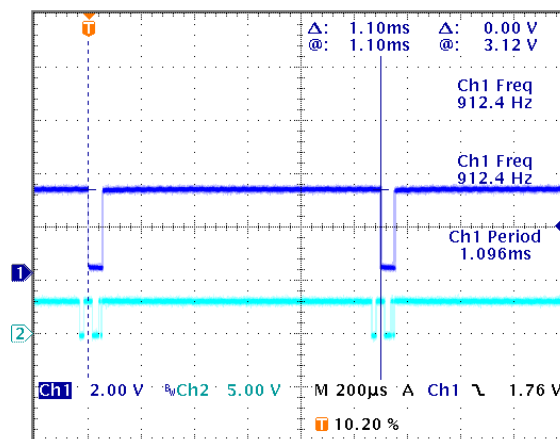


Figure 3. Frequency Scanning When Using Min-Calibration

**NOTE:** The fewer chip select transitions for every hop. (The dark blue is RSSI valid indicator signal and the light blue is the chip select line.)

The fastest scanning speed obtained with the current settings of 25 kHz RX BW and 1.2 kbps is 1.10 ms hop time, where 990 µs are spend waiting for the radio and 110 us are spend making decisions in the software or writing to the transceiver over SPI.

The speed will improve with increased SPI speed and MCU processing speed, but 8 MHz processing and 4 MHz SPI were chosen as a starting point for development as an example of low power and low cost MCU capabilities.

### 3.2 Timing Requirements of a Simple Asynchronous Channel Hopping System Using 12.5 kHz RX BW on a CC1120

Following the same analysis as in [Section 3.1](#), the scanning time is slower for the 12.5 kHz RX BW operation. Furthermore, as the CS response time increases, the overhead of downloading the calibration values for each frequency point becomes less significant.

**Table 3. Example of Time Consuming Elements in a Fast Frequency Hopping System Using @ 1.2 kbps and 12.5 kHz RX Channel Bandwidth**

1.2 kbps 12.5 kHz RXBW	Time With Pre-Calibration [µs]	Time With Minimum Calibration [µs]	Comments
RX – RX turnaround	50 µs	50 µs	Reset the RX chain (data sheet value)
RSSI VALID	2000 µs	2000 µs	CS response time (see <a href="#">Table 2</a> )
SPI overhead	60 µs	140 µs	SPI traffic to enable reading of RSSI data, change RX channel, re-enable RX mode and optionally download 3 calibration values (compare <a href="#">Figure 2</a> and <a href="#">Figure 3</a> )
Total	2100 µs	2180 µs	

### 3.3 Format of Transmit Packet for Simple Asynchronous Channel Hopping Systems

Based on the information in [Table 1](#) and [Table 3](#), it can be determined that the minimum required preamble times for a system are based on fast frequency scanning receiver architecture. The transceiver used in this document has two applicable options: 12 byte preamble and a 24 byte preamble. This creates the options described in [Table 4](#).

**Table 4. Narrow Band 2FSK Preamble Configurations Options**

Narrow Band 1.2 kbps 2-FSK Configuration						
Tested Configuration	Preamble	Sync Word	Payload	CRC	Total	
	24	2	0-30	2	30 – 60	Data [bytes]
	160	12.5	0-200	12.5	200-400	Time [ms]
Minimum Configuration	Preamble	Sync Word	Payload	CRC	Total	
	12	2	0-42	2	18 – 60	Data [bytes]
	80	12.5	0 - 280	12.5	120-400ms	Time [ms]

The default implementation is using a 12 byte preamble, which provides enough time for approximately 1 entire scanning loop including approximately 30 ms of spare time when using the 25 kHz RX BW setting. In the case of the 12.5 kHz RX BW setting, the scanning time is increased and an increased preamble length of 24 bytes is required.

## 4 Algorithm for Implementing Simple Asynchronous Channel Hopping

Both systems utilize the narrow band high performance settings of the CC112x series transceivers. The settings required for “Sniff mode” operation are virtually identical to the settings required for frequency scanning. Therefore, it is recommended to use the sniff mode panel for setting up custom system parameters in your frequency hopping system.

SmartRF™ Studio 7 [9] exports the recommended settings for each register. The only difference between the recommended sniff-mode settings and the frequency scanning settings are the “Wake on Radio” registers, which need to be removed effectively configuring the transceiver not to use the WOR feature.

To do this, identify the following four registers in the register export file and delete them.

- CC112X\_WOR\_CFG0
- CC112X\_WOR\_EVENT0\_MSB
- CC112X\_WOR\_EVENT0\_LSB
- CC112X\_RFEND\_CFG0

### 4.1 Transmitting Node

It is the responsibility of the transmit node to generate a pseudo random hopping sequence of 50 channels and make sure that the 50 channels are utilized per the FCC regulations.

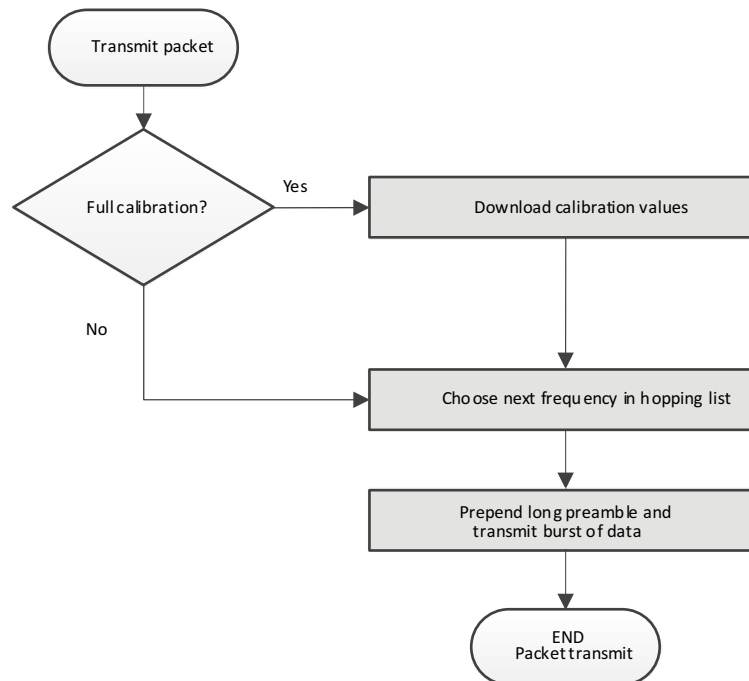
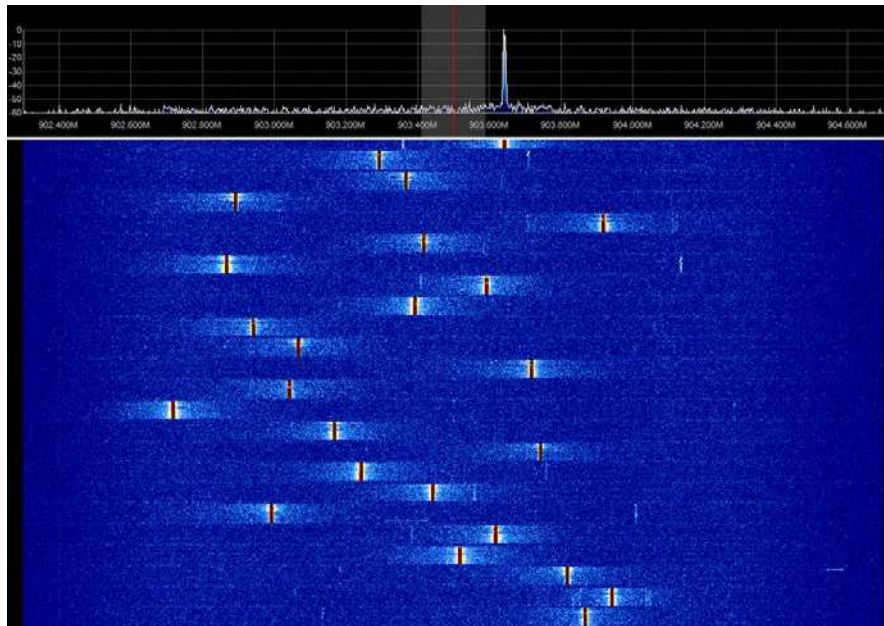


Figure 4. Flow Graph for Transmitting Node

The algorithm keeps track of the previously used channel and then selects the next channel in a list of available channels and uses this channel for transmission. This makes the transmit side very low power and very easy to implement. There is virtually no overhead in implemented a transmitting node with or without frequency hopping.

Figure 5 shows a spectral plot of how a frequency hopping transmits node looks over time. In this particular case, each burst is approximately 350 ms long and it hops at minimum 50 kHz to the next channel in the hopping sequence.



**Figure 5. Spectrum Analysis of Pseudo Random Hopping Sequence of a Narrow Band Hopping Transmitter**

## 4.2 Receiving Node

It is the responsibility of the receiving node to find a packet coming from a transmitting node at any time and on any frequency. This is accomplished by implementing a fast frequency scanning algorithm that runs the transceiver from channel to channel as fast as possible.

The concept of the scanning algorithm used by Simple Asynchronous Channel hopping system is pretty simple; it follows the flow described below:

1. Select next channel
2. Check for high RSSI on channel
  - a. If low RSSI → move back to line 1
  - b. If high RSSI → move to line 3
3. Packet reception on channel
  - Move to line 1

To get this system to actually work, timeouts to various stages of the flow need to be added creating a flow diagram that is more complex but still fairly simple to implement.

Consider that the flowchart shown in [Figure 6](#) runs inside an interrupt service routine that is tied to one of the timers inside the MSP430. This timer is programmed to provide a system tick and a tick rate of approximately 200  $\mu$ s has been chosen; this provides a good compromise between resolutions in time and consumed processing resources of the MSP430.

For each time the timer fires and interrupts, the flowchart starts from the beginning. The state of the state machine inside the interrupt service routine is maintained between each call because it is defined as a global variable in the entire software project.

The normal start of the system would be in “START” mode; this is setup externally before the interrupt service routine is enabled. This resets the timeout counters; reset the start frequency to 0 and finally update the state machine variable to the “CARRIER\_TO\_PKT” settings.

In “CARRIER\_TO\_PKT”, the state machine is looking for a carrier, it does this by looking for a valid RSSI from the transceiver core and then it compares the current RSSI to a moving average of historical RSSI values found. If the RSSI is good, then the system proceeds to the next state “PKT”. If not, it loops back and restarts the RX chain on a frequency from the list of possible frequencies.



In the "PKT", the state machine is looking for a valid Preamble, Sync word, and finally the full packet. If this does not happen within a specified amount of time, the state-machine automatically moves the receiver on a frequency and tries again.

At the end of the procedure, the interrupt service routine raises a flag to the calling program telling it that it is has found a packet and that someone should read the FIFO of the transceiver. This is done in the "SUCCESS" state and no further actions are taken here.

This is an interrupt service routine, it is perfectly okay to exit the routine at any time as the timer will just call the routine again shortly.

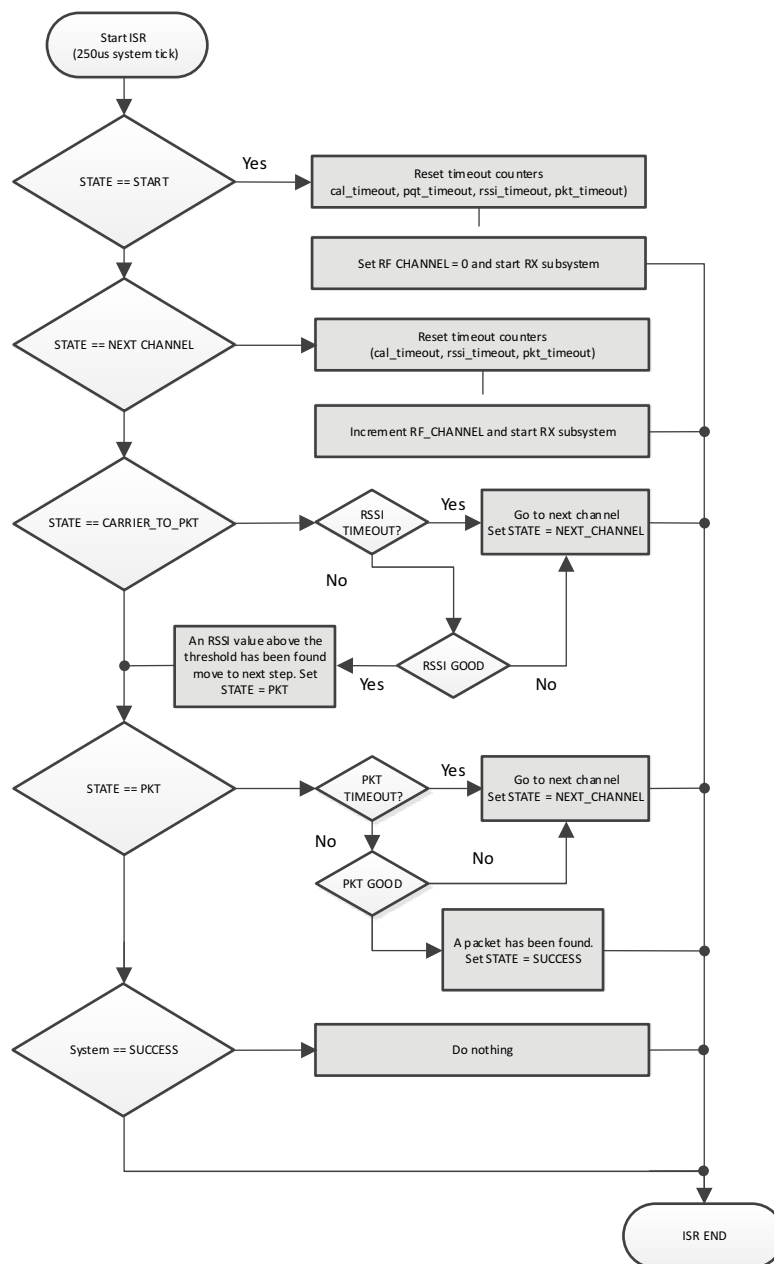


Figure 6. Flowchart of Interrupts Service Routine Implementing the RX Scanning Functionality

### 4.2.1 Architecture of Receiving Node

In an effort to enable users to easily port this code example to their platforms, all required functions are located in one source file called “radio\_scan\_drv.c”. All functions to implement both transmit and receive functionality are included in this file. The only requirement the file has is a working SPI driver and three ISR callback functions.

The transmitting function is very simple and has been implemented in the main loop of the demonstration software, however, the receive function is slightly more complex and has been implemented in a standalone ISR call back function that runs in the background of the host MSP430. By doing this, the main loop of the receive function remains very simple and easy to expand as *needed* by the end application.

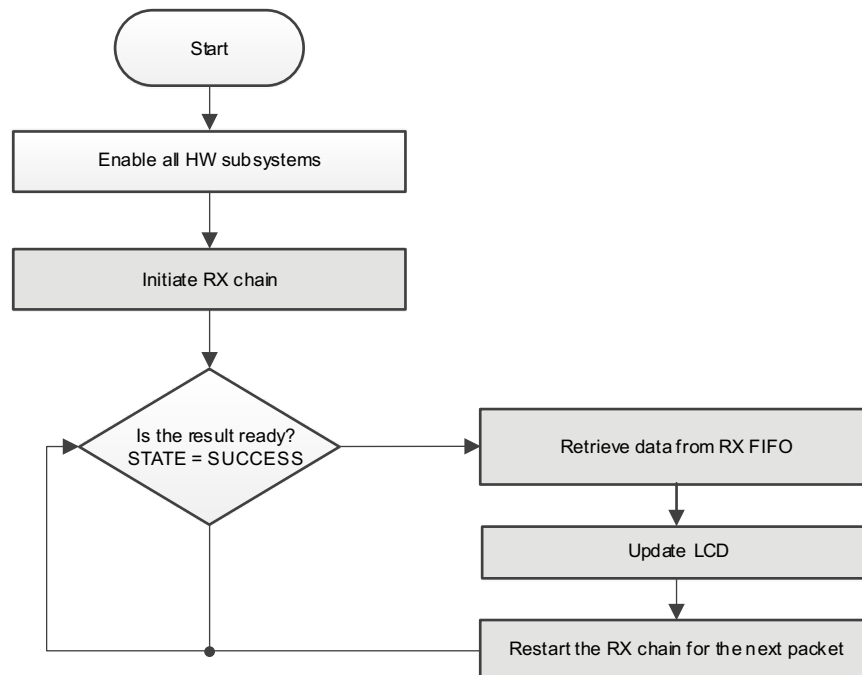


Figure 7. Flowchart of Main Loop of Demonstration Software

## 5 Review of System Configuration Parameters

The source code has been written so that it can easily be integrated into the final product software. The entire TX and RX algorithm is implemented in single source files. To implement the fast frequency scanning function used by the Simple Asynchronous Channel hopping system, the source file requires: one timer and two GPIO-based interrupts. The timer responsible for providing the system tick and the two GPIO-based interrupts are signals coming from the CC112x transceiver.

### 5.1 Configurations Options Frequency Hopping Algorithm

This section discusses each of the configuration settings possible in the source code.

#### 5.1.1 Enable optional range extender

It is possible to configure the system to use a range extender on either transmit, receive or both sides of the link.

```
#define ENABLE_RANGE_EXTENDER
```

By including this statement, the system will compile in support for the CC1190 range extender. The current consumption of the system increases on both sides of the link, however, in particular the TX side the current consumption will increase.

### 5.1.2 RX Bandwidth Configuration

It is possible to run this system using two different RX bandwidths: 12.5 and 25 kHz. Where 25 kHz is considered, the default operation (as this is the configuration used in SmartRF studio and 12.5 kHz) is considered the ultra-narrow band.

The main tradeoff between each ultra-narrow band operation and default operation is slower scanning times, but increased range.

```
#define ULTRA_NARROW_BAND
```

If ultra-narrow band is selected, the frequency accuracy of the two system being used must be within  $\pm 4$  kHz or approximately 5 ppm.

### 5.1.3 Calibration Configuration

There is an option to enable or disable full pre-calibration. If full pre-calibration is configured, the algorithm calibrates all channels individually as used by the system at the cost of slower scanning performance but at the advantage of full flexibility on frequency coverages.

```
#define RADIO_FULL_CAL
```

- “Pre calibration” can be used for all settings of the frequency steps and number of channels. The system simply performs a calibration at each and every chosen channel and stores this data for use each time the radio is configured to use that particular channel.
- “Minimum calibration” can only be used when the total frequency span from the lowest channel to the highest channel of operation is less than 4 MHz. In this case, a single RF calibration is performed at the center channel used for all channels.

Generally, if the total band of operation is less than 4 MHz wide, it is safe to disable full calibration mode.

### 5.1.4 Receiving Node

The algorithm that calculates the register values used to program the device for each of the frequency hopping channels needs to be configured with a couple of external constants in order to calculate the correct values.

The developer needs to supply the external reference frequency; for generality, the XTAL frequency with 1 kHz resolution is defined.

```
#define RF_XTAL_FREQ          32000    /* XTAL frequency, in 1 kHz steps */
```

Furthermore, the developer needs to configure the LO divider that is used during the normal operation of the device. For 868 and 915 MHz operation, the LO divider is always 4; however, for operations in other bands, this value needs to be changed. For more details, see the device-specific user's guide [\[2\]](#) [\[3\]](#).

```
#define RF_LO_DIVIDER        4        /* value of the hardware LO divider */
```

The next couple of defines simply configure the start, step size, and number of channels the system will operate in. Note that the system needs 6 bytes of RAM per frequency for systems using “full pre-calibration” and 3 bytes of RAM per frequency for systems using “minimum calibration”.

```
#define RF_START_FREQ      902750  /* start frequency defined in khz */
#define RF_STEPSIZE_FREQ   25       /* frequency step size defined in khz */
#define RF_NUM_CHANNELS    50       /* number of channels in system */
```

The following defines are used to configure the receiver timeouts. The first sets a timer “tick”, which is just a kind of heart beat for the system. This heartbeat is used to configure the timeouts for each section of the packet reception.

```
#define RF_SCAN_TIMER      8         /* set a timer tick to 200 us */
#define RF_CAL_TIMEOUT     2000000  /* approximately 500 s timeout */
#define RF_PACKET_TIMEOUT  2000     /* approximately 500 ms timeout */
#define RF_RSSI_TIMEOUT    40        /* approximately 10 ms timeout */
#define RF_PQT_TIMEOUT     120       /* approximately 30 ms timeout */
```

The “random seed” is used by the pseudo random number generator to generate the hopping sequence. Any integer number can be chosen and the algorithm will generate a unique sequence but repeatable sequence based on that value.

```
#define RANDOM_SEED        250       /* used to generate hop sequence */
```

The next couple of lines define the signal strength limits at which the radio considers the RSSI values above or under the noise floor. The receiver continuously monitors the background noise in each of the system channels and uses the `RSSI_LIMIT` as a starting value only after a few seconds of operation; the system will have found the actual noise floor on each channel and use the “above\_avg” define to setup a relative limit to the average noise floor.

```
#define RF_SCAN_RSSI_LIMIT -110      /* initial limit, used by moving avg */
#define RF_RSSI_ABOVE_AVG  8         /* use signal above moving average */
#define RSSI_OFFSET        102       /* constant defined in datasheet */
```

### 5.1.5 Transmitting Node

On the transmit side, the next couple of defines simply configure the start, step size and number of channels the system will operate in. Note that the system needs 6 bytes of RAM per frequency.

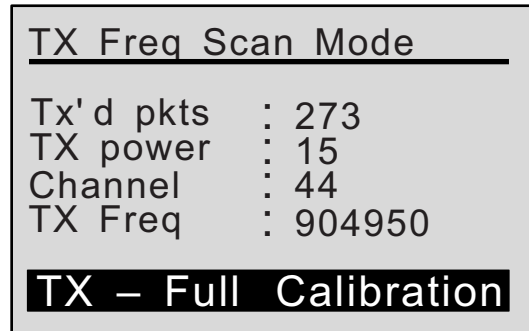
```
#define RF_START_FREQ      902750  /* start frequency defined in khz */
#define RF_STEPSIZE_FREQ   50       /* frequency step size defined in khz */
#define RF_NUM_CHANNELS    50       /* number of channels in system */
#define RF_BURST_TIMER     16384    /* send a packet every 500ms */
#define RF_CAL_TIMEOUT     500      /* approximately 500s */
#define RANDOM_SEED        250       /* used to generate hop sequence */
```

## 6 Using the Frequency Hopping System

### 6.1 Transmitting Node

After the TX configuration of the software has been downloaded to the TRXEB board and a CC1120 installed, the board will boot up and start transmitting messages.

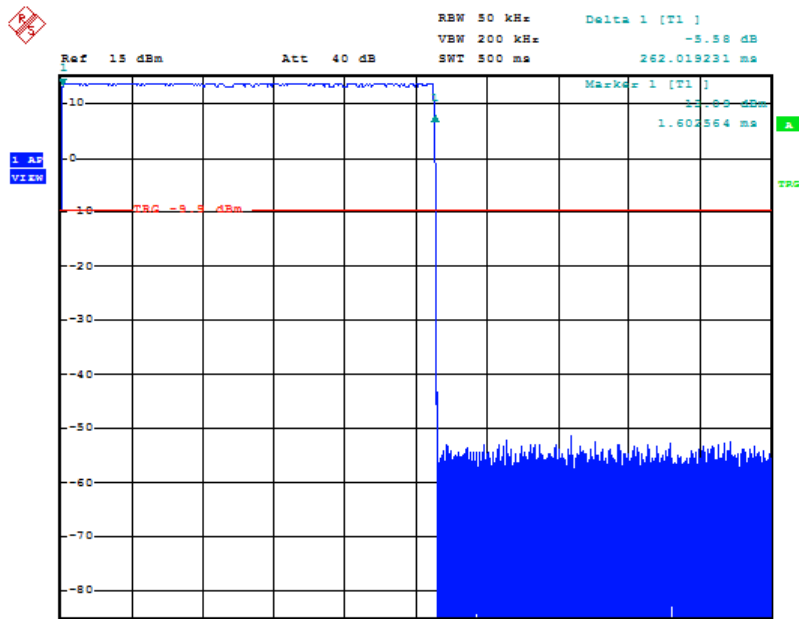
The display provides the total number of packets transmitted since it was last booted, it displays the active channel number (counted from 0) along with the corresponding frequency displayed in kHz.



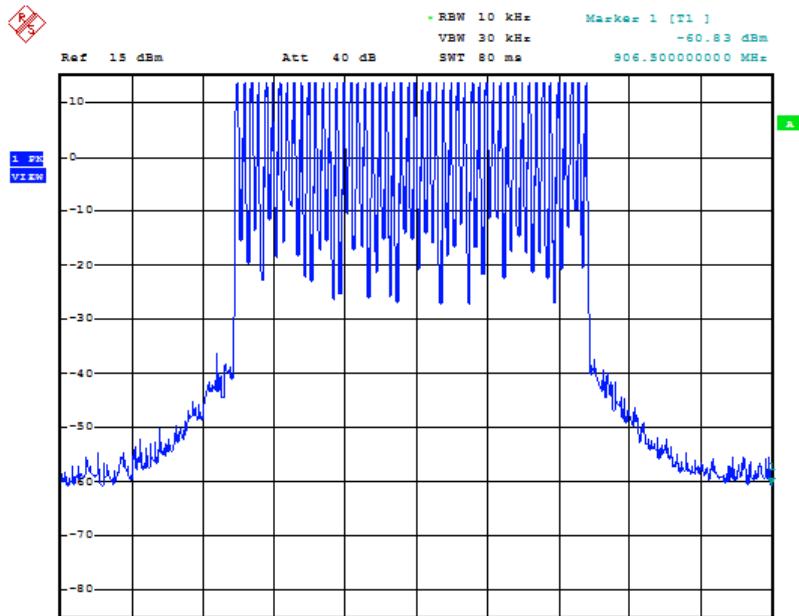
**Figure 8. LCD: Showing the Information During Transmission of Packets**

It is possible to correct the frequency error between the transmitting and receiving node by introducing a frequency offset on the transmitting side. This is done by carefully pushing the “UP” button for increase TX frequency in increments of 1 kHz and pushing the down button to decrease in the TX frequency in steps of 1 kHz. This is noted on the screen by the lowest digit on the TX frequency value displayed on the LCD.

In the default configuration, the transmitter creates an RF signal at one of the 50 channels starting from 902.750 MHz in steps of 50 kHz up to 905.25 MHz. This configuration complies with the FCC regulations outlined in the beginning of this application report [1].



Date: 19.MAR.2015 18:03:54



Date: 19.MAR.2015 17:51:42

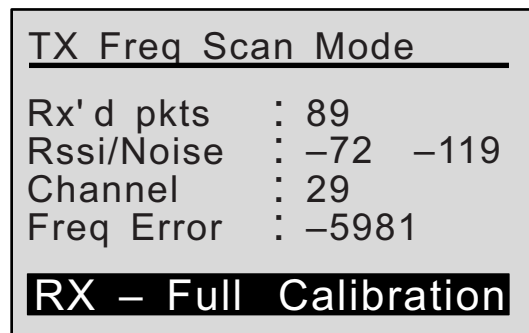
Figure 9. Measurement of the RF Burst Duration (Top) and “max-hold” Over 20 Seconds of All 50 Channels (Bottom)

## 6.2 Receiving Node

After the RX configuration of the software has been downloaded to the TRXEB board and a CC1120 installed, the board will boot up and start searching for messages. There is no configuration possible when the code is running, however, there is important information being displayed on the LCD.

The receiving node will display a total of five important pieces of information to use during the evaluation of the system.

- Rx'd pkts: displays the total number of packets received with a passing CRC.
- Rssi : the signal strength of the packet just received
- Channel: active channel number
- Noise: a moving average of the noise floor of the indicated channel
- Frequent Error: a moving average of the noise floor of the indicated channel



**Figure 10. LCD, Showing the Information Provided During Reception of Data**

As noted for the transmitting node, it is possible to correct the frequency error between the transmitting and receiving node by introducing a frequency offset on the transmitting side. For details, see [Section 6.1](#).

## 7 System Test

Range testing is one of the first things everyone wants to perform on a wireless system, but before range testing should start the two nodes should be calibrated to each other as the standard CC1120 development kits. Do not offer temperature compensated crystal oscillators (TCXO), but only a normal uncompensated crystal. So before starting the testing process, the indicated frequency error of the receiving node needs to be reduced to less than 8 kHz for the 25 kHz RXBW setting and less than 4 kHz for the 12.5 kHz setting. This is achieved by moving the transmitting node carrier frequency as described in [Section 6.2](#).

### 7.1 Long Distance Link Performance (residential area in Dallas, Texas)

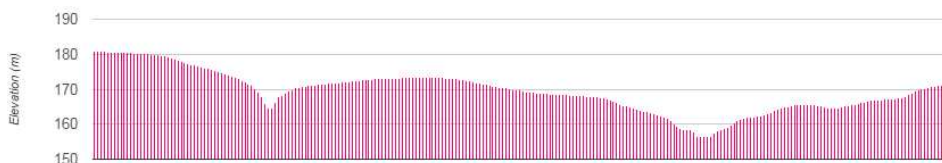
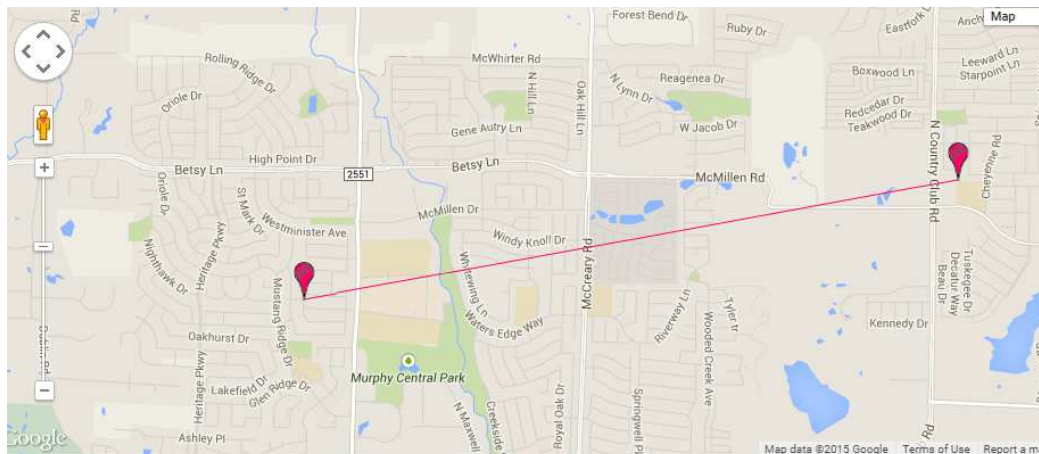
A range test was performed by leaving the transmitter indoors on the second story of a residential house and driving across a low lying area shown below on the elevation plot and parking at a school parking lot.

The mapping tool estimates the line of sight distance to be 4.6 km (2.9 miles). We measured a packet loss of ~20% and average signal strength of -108 dBm on the good packets. The back ground noise signal strength was indicated at -120 dBm by the system as described in [Section 6.2](#).

When comparing any long range test to any other long range test, it is very important to perform a review of the environment. There are a number of factors:

- Residential houses
- Large industrial complex building, including schools, big box stores, and so forth
- Wireless transmission towers (GSM/LTE/TV towers)

The most important and often most ignored is any elevation changes inside the range of the test. The signal attenuation of going into a valley is so severe that it will cut off any wireless signal at any strength unless it originates from a very high location (a tower).



**Figure 11. Results of Range Test, Plots by Daftlogic [11]**

Before this test was conducted, the frequency offset between the transmitting and receiving node was calibrated using the approach described in [Section 6.1](#). The average frequency error reported during the testing was under 2 kHz.

### 7.2 Close-In Link Performance (inside a house with range extender enabled)

As described earlier in [Section 3](#), the system is based on a sequentially scanning of all channels looking for a signal strength that is above the noise floor by a certain amount. This method has a side effect that causes issues when testing the system is close range testing.

From the device-specific data sheet, it is found that the adjacent channel rejection and second adjacent channel rejection is approximately 52 dB in the default configuration of the system. For close-in testing, the received signal strength is approximately -40 dBm (while testing at less than 10 meter or 30 feet range).

Subtract the adjacent channel rejection from the maximum signal strength and the result will be the apparent increase in noise floor at the channel of interest. For close-in range testing, this causes the system to spend time on the wrong channel looking for a preamble and in some cases not catching the burst on the right channel.



## 8 References

1. Electronic Code of Federal Regulations: [http://www.ecfr.gov/cgi-bin/text-id?SID=4d5c2199d21ed74af96d9bc69918adc7&mc=true&node=pt47.1.15&rgn=div5#\\_top](http://www.ecfr.gov/cgi-bin/text-id?SID=4d5c2199d21ed74af96d9bc69918adc7&mc=true&node=pt47.1.15&rgn=div5#_top)
2. Texas Instruments: [CC112X/CC1175 low-power high performance Sub-1 GHz RF transceivers/transmitter user's guide](#)
3. Texas Instruments: [CC120X low-power high performance Sub-1 GHz RF transceivers user's guide](#)
4. Texas Instruments: [CC1120 high-performance RF transceiver for narrowband systems data sheet](#)
5. Texas Instruments: [CC1121 high-performance low-power RF transceiver data sheet](#)
6. Texas Instruments: [CC1125 ultra-high performance RF narrowband transceiver data sheet](#)
7. Texas Instruments: [CC1200 low-power, high-performance RF transceiver data sheet](#)
8. Texas Instruments: [CC112x, CC1175 silicon errata](#)
9. Texas Instruments: [SmartRF Studio 7 v1.18.0 software \(http://www.ti.com/lit/zip/swrc176\)](#)
10. Texas Instruments: [CC112x/CC120x RX sniff mode](#)
11. Data Logic Distance Calculator: <http://www.daftlogic.com/projects-google-maps-distance-calculator.htm>

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale ([www.ti.com/legal/termsofsale.html](http://www.ti.com/legal/termsofsale.html)) or other applicable terms available either on [ti.com](http://ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2019, Texas Instruments Incorporated