

Clear Channel Assessment Using TI MSP430 and CC2500 Radio

Randy Wu

Technical Staff

ABSTRACT

This application report describes a 2.4-GHz anti-jamming frequency-hopping system consisting of a designated Transmitter and Receiver, each with distinct functions to avoid RF-channel interference. The Transmitter board emulates the control device (e.g., remote control) by sending commands to be processed on the Receiver side (e.g., TV, stereo, set top box, or gaming console). Since many popular consumer devices use the 2.4-GHz RF spectrum (e.g., cordless phones, wireless routers, microwaves, and PC/Bluetooth® accessories), these systems need to implement a proprietary frequency-hopping scheme in the event that the channel being used by the Transmitter/Receiver pair is already being used by other wireless systems in the vicinity. The specific radio device used for this demo is the TI CC2500 (2.4 GHz) RF transceiver module. The TI portfolio also includes the CC1100 RF transceiver (315/433/868/915-MHz ISM/SRD bands), which shares a common set of configuration registers as those used in the CC2500. This makes it easy to switch between these two RF transceiver devices, so the radio-specific portion of this demo can be used for both types of TI CCxxxx devices.

Contents

1	Introduction	2
2	Hardware Evaluation Platform	2
3	Application Function and Usage	4
4	Theory of Operation	6
5	Software Architecture	10
6	Conclusion	12
7	References	12
Appendix A MSP430 + CCxxxxEMK Demo Board Configuration		13
Appendix B Software Flow Charts		14

List of Figures

1	MSP430FG4618 Experimenter Board	2
2	LCD Display Icons	5
3	Receiver Function Flow Chart	6
4	Transmitter Function Flow Chart	7
5	Repeated Data Packets Write Burst Timing	8
6	ACK Reception During Burst Write Sequence	9

1 Introduction

The system described in this application report implements a channel-hopping scheme in which the Receiver continuously scans a predetermined set of RF channels for clarity and jumps to the clearest channel, while the Transmitter continuously cycles through the RF channels until it receives an acknowledgement from the Receiver to maintain a continuous wireless RF link. This implementation of "link maintenance" allows both the Transmitter and Receiver to be continuously in sync with each other and ready to transmit data in both directions, waiting for an event (e.g., user button press) on the Transmitter side while the Receiver needs to be on the clearest channel to ensure that messages sent back and forth have the highest probability of success of reaching their final destinations.

The low cost and ultralow-power MSP430 plus CCxxxx-based RF demo described in this application report leverages existing development boards manufactured by Texas Instruments. With this implementation, the MSP430 remains in Low Power Mode 3 (LPM3) most of the time where the chip itself consumes an average of only 0.8 μ A! The CCxxxx on the Transmitter side features a Wake-On-Radio (WOR) feature, whereby the device periodically goes to sleep and wakes up to catch potential incoming packets. This conserves power by reducing the time it stays in a fully active Receiver mode, since setting any active transmit or receive mode for any RF transceiver device consumes more power vs. an idle mode state. In most cases, the average current consumption (for the radio itself) configured for WOR mode is approximately 100 μ A!

For ease of development, the SmartRF™ Studio graphical user interface tool suite can be downloaded and used to automatically generate CCxxxx configuration register settings based on any combination of desired operation configurations.

2 Hardware Evaluation Platform

2.1 Texas Instruments MSP430FG4618 Experimenter Board

The MSP430FG4618/F2013 Experimenter Board is a comprehensive development target board that can be used for a number of applications. Wireless communication is possible through an expansion header which is compatible with all TI wireless evaluation modules. In addition, interfaces to a 4-mux LCD, UART connection, microphone, audio output jack, buzzer, and capacitive touch pad enable the development of a variety of applications. Communication between the two onboard MSP430 devices is also possible. All pins of the MSP430FG4618 device are made available either via headers or interfaces for easy debugging. Basic sample code for this board is available online at www.msp430.com.

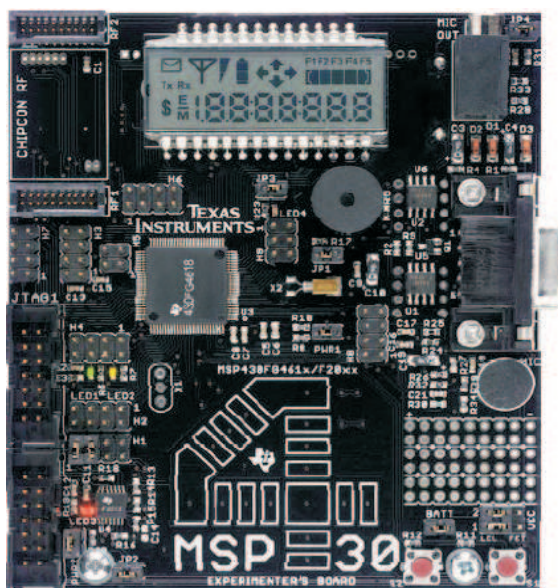


Figure 1. MSP430FG4618 Experimenter Board

2.2 TI CCxxxxEMK Daughterboard

Interface to the wireless world is accomplished via the wireless evaluation module header supporting the CCxxxxEMK boards from TI. The transceiver modules are connected to the USART of the MSP430FG4618 configured in SPI mode via software. Software libraries that interface the MSP430 to these transceivers are available at www.msp430.com. The demo application implemented in this application report relies on these easy-to-use libraries as the foundation for the low-level interactions between the MSP430 and CCxxxx devices.

2.3 Benefits of Using the TI CCxxxx RF Transceiver

The CCxxxx is a low cost, true single-chip RF transceiver designed for very low power wireless applications. The circuit is intended for the ISM (Industrial, Security, and Medical) and SRD (Short Range Device) frequency bands. The RF transceiver is integrated with a highly configurable baseband modem. The modem supports various modulation formats and has a configurable data rate of up to 500 kbps. The CCxxxx provides extensive hardware support for effective RF communications, such as packet handling, data buffering, burst transmissions, clear channel assessment, link quality indication, and WOR. The main operating parameters and the 64-byte transmit and receive buffer FIFOs can be controlled via standard 3-wire SPI interface. Only a few passive components are required to interface the CCxxxx with the MSP430 ultralow-power microcontroller. By using the TI CCxxxx device, the following features greatly benefit the system with respect to RF communications:

- Automated packet handling – automates training, frame synchronization, addressing, and error detection
- Wake-On-Radio (WOR) – achieves automatic low-power periodic Receiver polling
- Tx-If-CCA™ feature (CCA) – prevents multiple Transmitter collisions and automatically arbitrates the data direction for listen-before-talk systems
- Forward error correction (FEC) with interleaving – reduces gross bit error rate when operating near the sensitivity limit
- Carrier sense (CS) – enables detection of busy channels used by other potentially interfering systems (e.g., wireless router, cordless phone)
- Link quality indicator (LQI) – estimates how easily a received signal can be demodulated; can be used as a relative measurement of the link quality
- Automatic frequency compensation (AFC) – align own frequency to the received center frequency
- Manchester encoding – provides an additional level of data integrity for packet transmissions
- Deep 64-byte Tx FIFO data buffer – allows MCU to store sufficient amounts of data pending to be transmitted by the CCxxxx at any time, regardless of what the CCxxxx is doing
- Deep 64-byte Rx FIFO data buffer – allows CCxxxx to store sufficient amounts of received data and save it for the MCU to be read at any time
- Efficient standard SPI interface – all registers can be programmed with one "burst" transfer
- Digital RSSI output – allows the MCU to quickly read (in real-time) a simple 8-bit byte representing the Received Signal Strength Indicator of the last received data packet

3 Application Function and Usage

3.1 Receiver Board

When the Receiver board is first powered up, it parks itself on frequency #1 (denoted as **F1** on the LCD display). Pressing **SW2** once increments the RF channel to the next frequency until the last (highest numbered channel) preset frequency has been reached. The next button press enables the Receiver board's AUTO scan mode (AUTO is displayed on the LCD just below the RF channel markers). Once the Receiver is in this AUTO mode, it periodically scans all of the RF channel presets and jump to the clearest channel if communications with the Transmitter board has been lost. Pressing the **SW2** button while the Receiver board is in AUTO scan mode sets the board's frequency back to **F1**, and AUTO scan mode is disabled until the subsequent **SW2** button press, once the board is set on the highest-numbered preset RF channel.

When the Receiver board is parked (due to manual channel change or auto scan mode) on a preset RF channel, the Antenna icon on the LCD display stays on, and when the Receiver board is scanning through a round of the preset channels, the Antenna icon is cleared during this brief amount of time. During the channel scan sequence, in the event that the Receiver happens to receive a broadcast beacon from the Transmitter, an acknowledge is not sent back until the channel scan has been completed. This way, the Receiver does not mistakenly send back an acknowledge while the Transmitter beacon that is sent out during a normal scan period coincides with the Receiver's channel scan sequence.

Whenever the Receiver board receives a valid packet and determines that the packet contains its own hard-coded address, the Mailbox/Envelope icon on the LCD display flashes momentarily (approximately one second).

Pressing **SW1** on the Receiver board sends a command to toggle **LED1** on the Transmitter board. Since the Transmitter board is most likely configured for WOR mode, the delay between the pressing of **SW1** and the toggling of **LED1** is more noticeable to the human eye.

3.2 Transmitter Board

When the Transmitter board is first powered up, it parks itself on frequency #1 (denoted as **F1** on the LCD display). Pressing **SW1** and **SW2** once sends a command to toggle **LED1** or **LED2** on the Receiver board, respectively.

When the Transmitter board is parked on one of the preset RF channels, the Antenna icon on the LCD display stays on, and whenever the Transmitter board is scanning each RF channel looking for the Receiver board, the Antenna icon is cleared during this time. The Transmitter board periodically sends out a broadcast beacon to the Receiver. If the Receiver sends back an acknowledge to the beacon request, the Transmitter board remains on its currently-selected preset RF channel. If no response to the beacon is received, the Transmitter cycles through each preset RF channel until it receives an acknowledge to the beacon request, and at that time the Transmitter parks itself on the currently-selected channel.

Whenever the Transmitter board receives a valid packet and determines that the packet contains its own hard-coded address, the Mailbox / Envelope icon on the LCD display flashes momentarily (approximately one second).

3.3 User Interfaces

3.3.1 LCD Display

The LCD display on each board is used as an interface for user functions and feedback:

- **Tx or Rx** – denotes Transmitter or Receiver configuration (determined at project build time)
- **Antenna** – denotes the board is currently settled on an RF channel; disappears whenever Receiver is scanning channels in AUTO mode and whenever Transmitter is cycling through channels looking for the Receiver
- **Mailbox/Envelope** – blinks for approximately (at most) one second whenever a valid packet (with a packet ID which matches the hard-coded address of the board receiving the message) has been decoded
- **AUTO** – displayed in the lower right-hand corner whenever channel-hopping auto scan mode is enabled via SW2 (applies to Receiver mode configurations only)
- **F1 F2 F3 F4 F5** – shows which preset RF channel the board is currently parked on; highest channel number depends on the amount of preset channel frequencies programmed in the application software (only one of these channel presets can be selected at any one time)

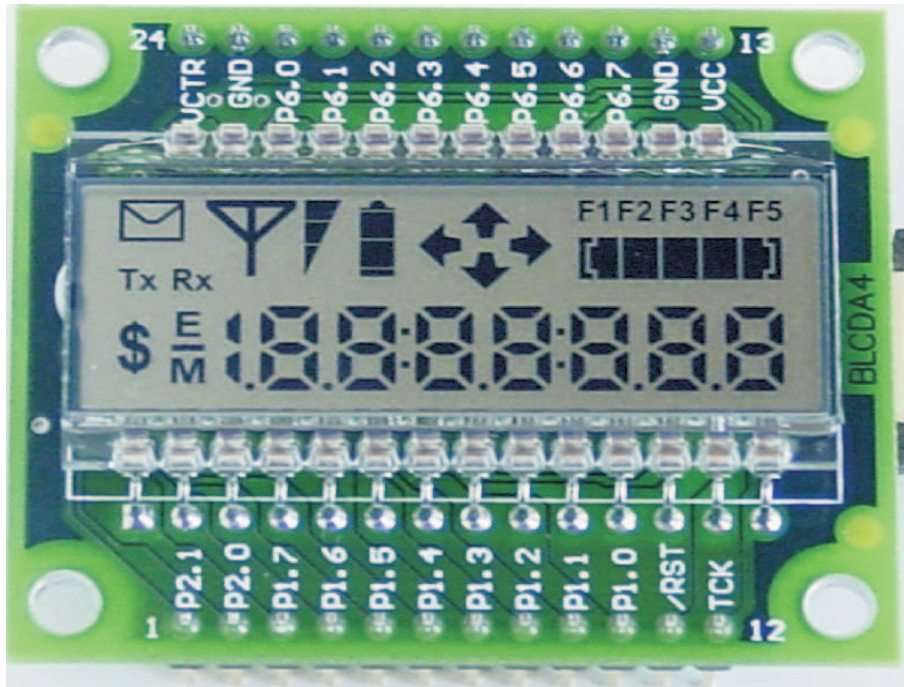


Figure 2. LCD Display Icons

3.3.2 LED Functions

The LEDs are also used to provide feedback on the interactions between the demo boards:

- **LED1** – mapped to the other board's Switch #1 (**SW1**) for **LED1** Toggle commands
- **LED2** – mapped to the other board's Switch #2 (**SW2**) for **LED2** Toggle commands
- **LED4** – turns on whenever a transmit burst sequence is initiated; turns off whenever the corresponding acknowledge has been received (used to determine ACK turnaround time)

4 Theory of Operation

4.1 Receiver Operation

The main responsibilities of the Receiver consist of the following:

- Scan all preprogrammed RF channels at periodic intervals
- Detect RF carrier on each channel during scan mode and maintain incrementing "carrier sense busy counters" for each channel
- Assess which channel is the clearest and jump to that channel if communications with the Transmitter have been lost
- Decode incoming packets from the Transmitter, process the command as quickly as possible, and then generate an acknowledge packet to send back to the Transmitter to verify reception of the command packet

In this system, the Receiver acts as the "RF Master" in regards to determining which RF channel is to be used at any time. It has the responsibility of clearest channel assessment and determines which channel to park on to avoid other RF sources that may be jamming the currently-selected RF channel. Once the Receiver has parked, it assumes the Transmitter will find the Receiver during times of nonactivity on a periodic basis.

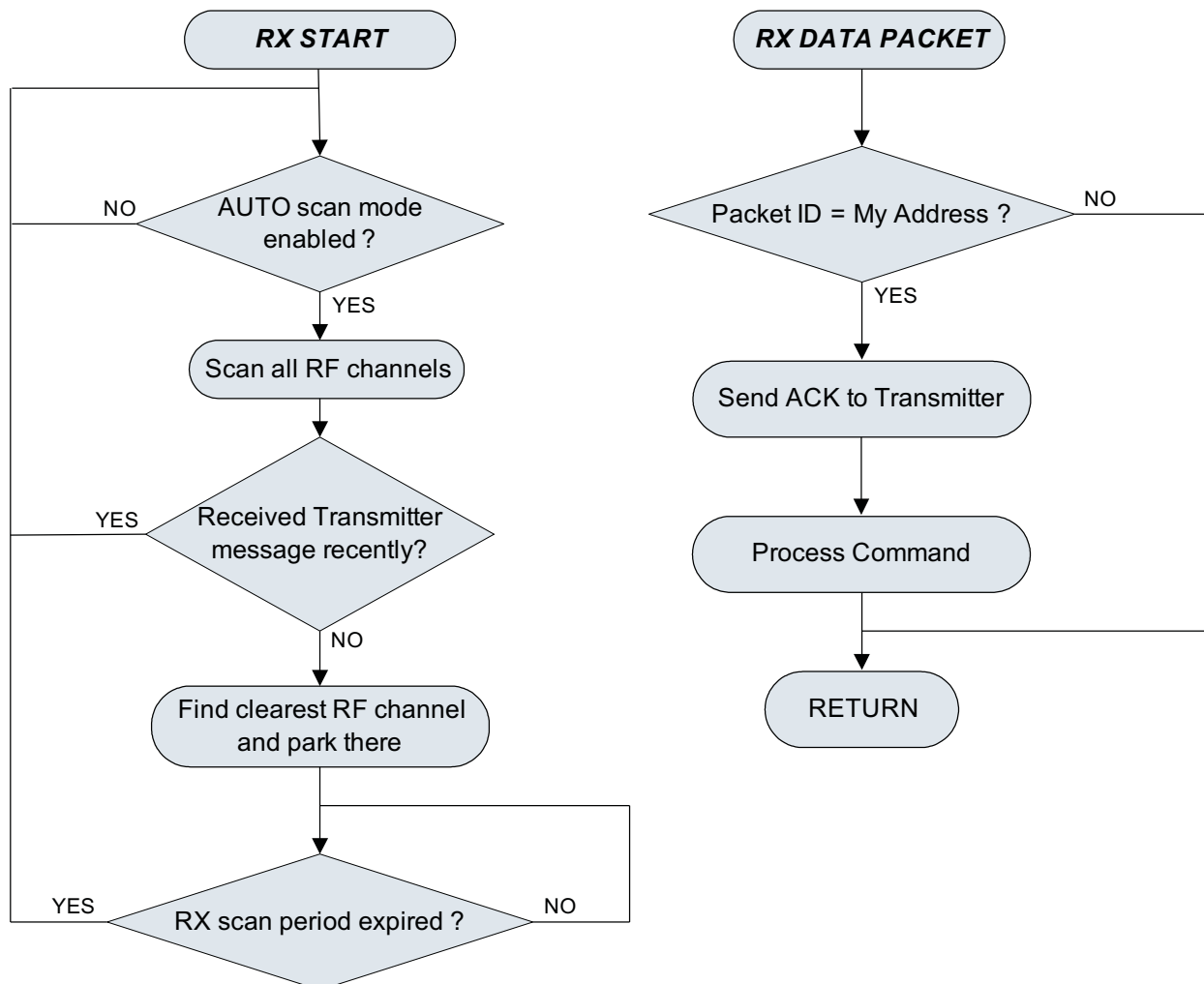


Figure 3. Receiver Function Flow Chart

4.2 Transmitter Operation

The main functions of the Transmitter consist of the following:

- Detect all user button presses and send command to the Receiver
- Send out periodic beacon packet to ensure Receiver is still on the same channel
- Cycle through the predetermined RF channels until the Receiver is found

In this system, the Transmitter acts as the "RF Slave" since it must track the Receiver and park itself on whichever channel the Receiver has assessed as the clearest channel to use. It has the responsibility of sending out a periodic beacon and making sure that the Receiver responds to the beacon with a simple acknowledge. If the Receiver does not respond to the Transmitters beacon, the Transmitter increments its channel once per second and sends out the beacon on each channel, until the Receiver responds to the beacon. Once the Receiver responds to the beacon, the Transmitter stays on the current channel and periodically sends out the beacon to ensure that the Receiver is still on the same channel.

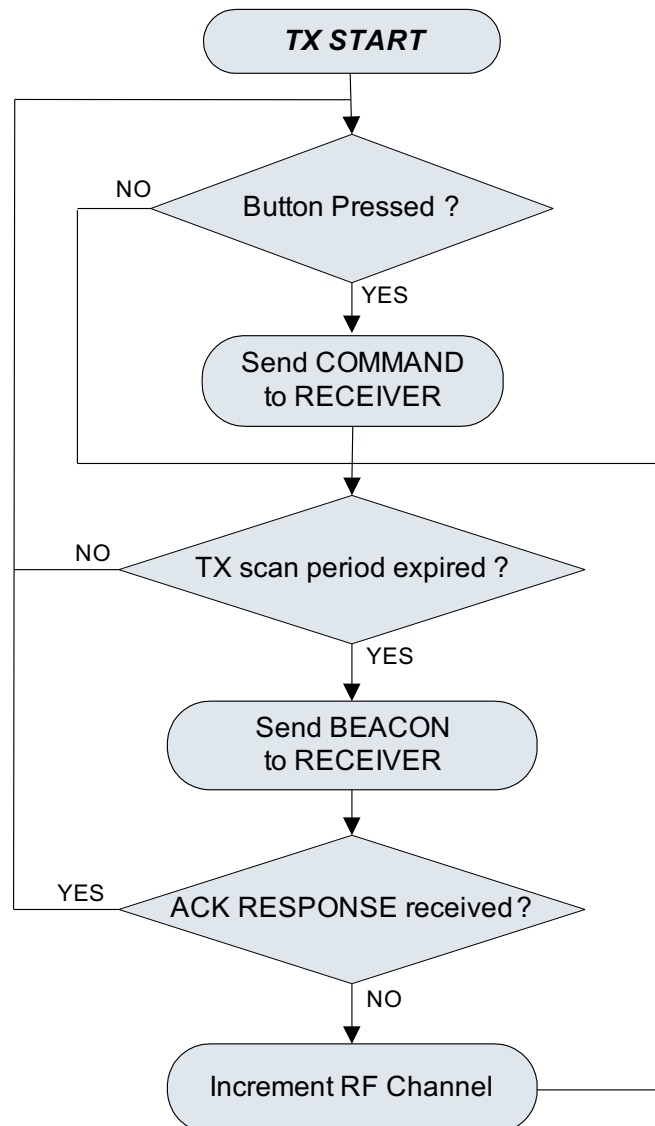


Figure 4. Transmitter Function Flow Chart

4.3 Transmit Burst Sequence (Repeating Data Packets)

Whenever the MSP430 needs to send a packet to the other end (the destination), a Transmitter data burst sequence is initiated between the MSP430 and CCxxxx. The MSP430 updates a specified number of bytes in its transmit buffer and then proceeds to write the sequence of bytes that make up the information to be stored in the packet. This sequence of bytes is then written to the CCxxxx TX FIFO via SPI interface where it is the job of the CCxxxx to packetize the data and send it off when the MSP430 issues a command for the CCxxxx to go into Transmit mode once the data packet information has been written into the CCxxxx TX FIFO buffer.

To maintain redundancy and data link integrity (and to also work with CCxxxx devices configured for the power-efficient WOR mode), each packet of information that needs to be sent is written multiple times to the CCxxxx TX FIFO buffer with a preset delay between each write operation. The software parameters that determine how many packets per burst are sent, how many milliseconds to delay between each packet in each burst, how many milliseconds to wait in between each burst for an acknowledge from the other end, and times to repeat the burst sequence are all easily configurable in the application software:

- **TXBURSTNUMPKTS** – number of data packets to write to the CCxxxx TX FIFO per burst
- **INTRPCKTDLYMSEC** – number of milliseconds to delay in between each data packet write
- **ACKTIMEOUTMSEC** – number of milliseconds to delay between each TX FIFO data write
- **TXRETRIESMAXCOUNT** – total number of burst sequences to execute

As a potential enhancement to the existing application report code, a Message ID field could be added to the packet structure so that when the Receiver decodes incoming packets, it can know if a specific packet is a duplicate repeated packet or if a new command packet was initiated on the Sender side. In this fashion, repeated data packets can be distinguished from new command packets caused by additional events that result in the same transmitted command code.

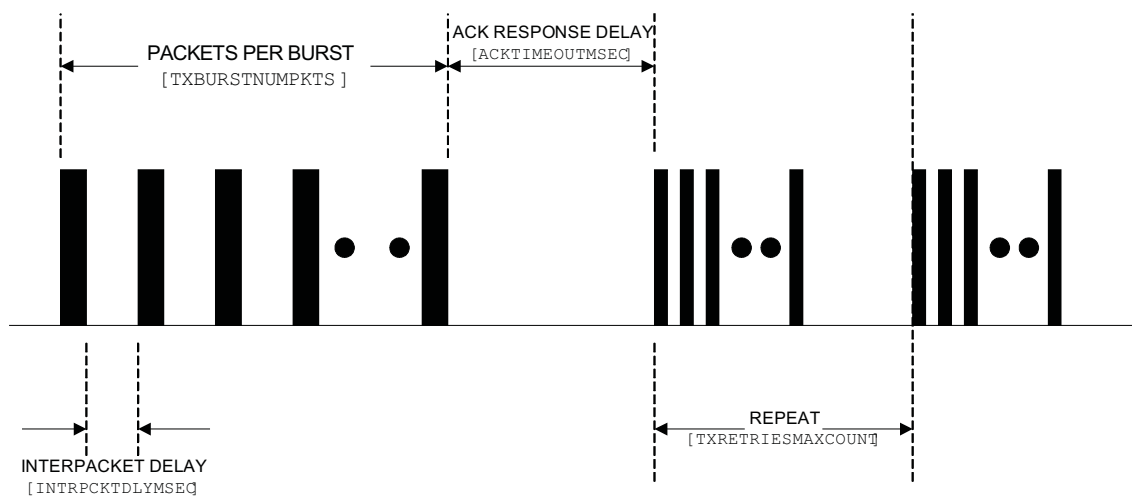


Figure 5. Repeated Data Packets Write Burst Timing

During the time the MSP430 is writing data packet information to the CCxxxx TX FIFO buffer, global interrupts are enabled and the CCxxxx is configured for standard Receive mode during all of the inter-packet write delays. The Tx-If-CCA™ feature of the CCxxxx RF transceiver is enabled, so outgoing Tx data packets are not transmitted if an acknowledge packet from the expected sender is being received. Should the CCxxxx receive the expected acknowledge data packet from the other end during the burst sequence of writes, the current burst write sequence is immediately aborted and no further duplicate packets are written to the CCxxxx TX FIFO.

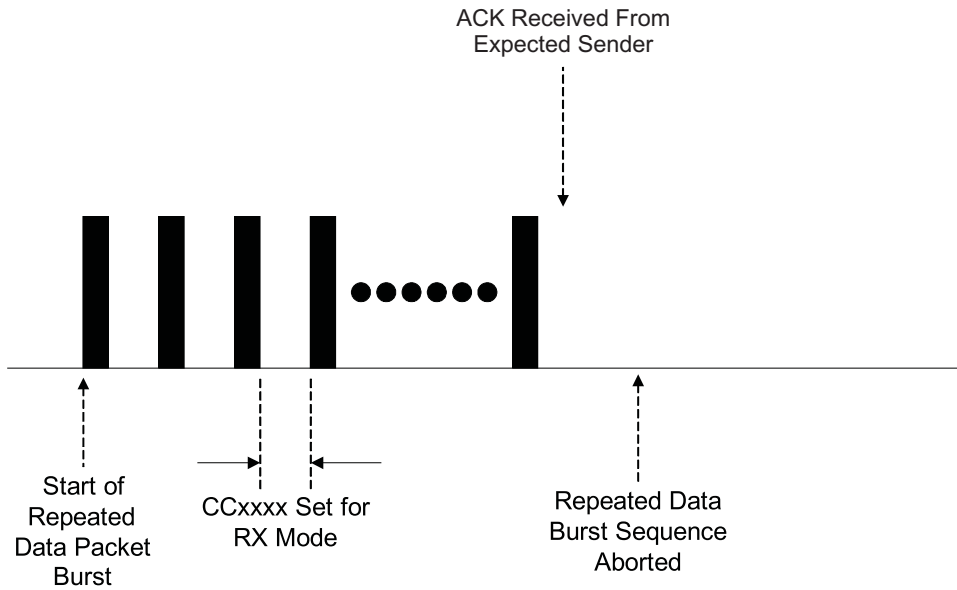


Figure 6. ACK Reception During Burst Write Sequence

5 Software Architecture

5.1 System Configuration Parameters

The main header file (`mcp430_cc2500_freqhop.h`) contains all of the critical parameters used to determine the build configuration for the Transmitter and Receiver modes. These parameters, which are all set at build time, control key functions, such as how many seconds are in a period for the Receiver to perform a channel scan or how many packets are sent by the Transmitter before it quits waiting for an acknowledge from the Receiver. The critical parameters that can be changed for optimum performance of the system are:

```

/* Transmission Parameters */
#define TXRETRIESMAXCOUNT    3 // Number of times to retry sending Tx burst to get
ACK
#define ACKTIMEOUTMSEC        100 // Number of milliseconds to wait for ACK
#define ACKTXNUM               15 // Number of ACK's to send out whenever new command
is rx'd
#define ACKPCKTDLYUSEC        5 // Number of usec to delay between each ACK
transmission
#define SCANCHANDLYMSEC        30 // Delay to allow CCA value to update
#define BUSYCHANCONSECMIN      3 // Consecutive times a channel must be busy before
jumping
#define BUFFERSIZE              4 // Size of buffer to store Tx and Rx FIFO data
#define RFNUMCHANS              4 // Number of channels in frequency hop table
#define TXPACKETLEN             3 // Number of Tx bytes to send in a single data
transmission
#define BUFFIDX_LEN             0 // Byte location for the PACKET LENGTH in the Tx
buffer
#define BUFFIDX_DST             1 // Byte location for the DEST ADDRESS in the Tx
buffer
#define BUFFIDX_CMD             2 // Byte location for the COMMAND field in the Tx
FIFO buffer

/* Command Codes */
#define CMD_NULL                0x00 // NULL Command code (No Action)
#define CMD_LED1CLR             0x01 // Command code for setting LED1
#define CMD_LED1SET             0x02 // Command code for clearing LED1
#define CMD_LED2CLR             0x03 // Command code for setting LED2
#define CMD_LED2SET             0x04 // Command code for clearing LED2
// *** Add more command codes as needed ***
#define CMD_ACK                 0xFF // Value corresponding to ACK

/* Miscellaneous System Settings */
#define RXSCANPERIODSEC         2 // Period for clear channel scan sequence
#define TXSCANPERIODSEC         5 // Period for Transmitter to ping the Receiver
#define SWDEBOUNCEMSEC          90 // Switch debounce delay in msec
#define LASTACKTIMEOUTSEC (TXSCANPERIODSEC/2) // Allow channel scan once absence of
ACK reaches this value (in seconds)

```

5.2 Transmitter/Receiver Board Configuration

Since the overall functions of the Transmitter and Receiver differ slightly, the program image must be built specifically for each configuration. One of these configurations must be defined in the "msp430_cc2500_freqhop.h" header file by virtue of the other configuration being commented out. For example, to build the Transmitter configuration, the lines of code must look like the following:

```
/* System Build Configuration */
#define _TRANSMITTER // [OPTION #1] Build variable for TX module
// #define _RECEIVER // [OPTION #2] Build variable for RX module
```

Similarly, to build the Receiver configuration, the previous lines of code must be changed:

```
/* System Build Configuration */
// #define _TRANSMITTER // [OPTION #1] Build variable for TX module
#define _RECEIVER // [OPTION #2] Build variable for RX module
```

5.3 TI CC2500 Receiver Mode Selection

While the TI CC2500 Receiver is active, the current consumption for the entire demo board can be as high as 17 mA. The Receiver board actually draws more current than the Transmitter (typically around 3 mA). Thus, the designated Receiver board will likely consume more current than the dedicated Transmitter board. It is important to shut off the Receiver when it is not in use or configure the Receiver for WOR mode with a duty cycle that is applicable for the end application. For a typical WOR application with a 37.5% Receiver on duty cycle, the average current consumption for the CC2500 itself can be as low as 100 μ A!

To select which CC2500 receive mode is to be used for each board, simply enable the #define (_SWOR or _SRX) that corresponds to the desired mode. For the default build configuration, the Transmitter operates in WOR mode while the Receiver operates in standard "always on" Receiver mode.

```
/* Operating Modes */
#define TXADDR 0x01 // Address of the designated Transmitter
#define RXADDR 0x02 // Address of the designated Receiver
#ifdef _TRANSMITTER
  #define ADDR_SELF TXADDR // Address of myself
  #define ADDR_OTHER RXADDR // Address of the other end
  #define _SWOR // Configure Receive mode for Wake-On-Radio (WOR) mode
  #define TXBURSTNUMPKTS 15 // Number of repeated bursts per data packet (min=2)
  #define INTRPCKTDLYMSEC 10 // Number of msec to delay between each burst
#endif /* _TRANSMITTER */
#ifdef _RECEIVER
  #define ADDR_SELF RXADDR // Address of myself
  #define ADDR_OTHER TXADDR // Address of the other end
  #define _SRX // Configure Receive mode for Standard active RX mode
  #define TXBURSTNUMPKTS 100 // Number of repeated bursts per data packet (min=2)
  #define INTRPCKTDLYMSEC 5 // Number of msec to delay between each burst
#endif /* _RECEIVER */
```

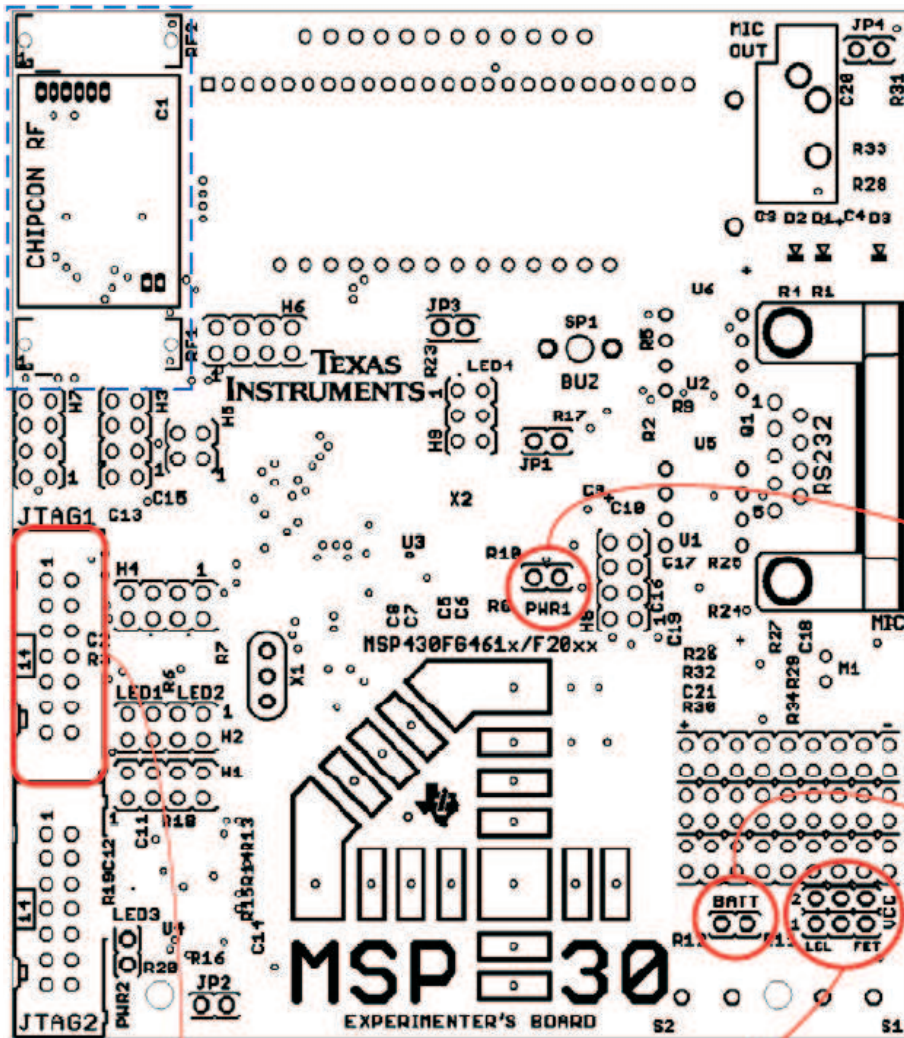
6 Conclusion

The purpose of this application report is to provide a highly adaptable hardware and software template for creating a customized application involving basic frequency hopping using the ultralow-power MSP430 and CCxxxx devices. The fully-functional and production-ready hardware and software footprints provided with this application report serve to jump start any RF design where high performance in conjunction with optimum ultralow power consumption is required. The basic frequency-hopping scheme can be expanded to incorporate more intelligence in the form of spectral analysis and more sophisticated channel-hopping and recovery algorithms. The additional logic can be stacked on top of the existing RF protocol layers to achieve a robust RF communications system. For portable applications where battery consumption is a major concern, the MSP430 low-power modes and the CCxxxx Wake-On-Radio (WOR) mode can be used for optimum battery life plus system performance. The built-in features indigenous to the CCxxxx RF transceiver alleviate the microcontroller from having to implement antijamming characteristics in software.

7 References

1. *MSP430x4xx Family User's Guide* ([SLAU056](#))
2. *MSP430xG461x Mixed Signal Microcontroller* ([SLAS508](#))
3. *MSP-EXP430FG4618/F2013 Experimenter's Board User's Guide* ([SLAU213](#))
4. *CC2500 Single-Chip Low-Cost Low-Power RF Transceiver* ([SWRS040](#))
5. *CC1100 CC2500 Wake On Radio* ([SWRA126](#))
6. *Software for CC1100/CC2500 and MSP430* ([SWRA141](#))

Appendix A MSP430 + CCxxxxEMK Demo Board Configuration



PWR1: Place to power FG4618. Use also to measure current.

BATT: Place to locally power board from 2xAAA batteries. In this case, use also to measure the total board current consumption.

JTAG1:
FET tool connector to program / debug the MSP430FG4618.

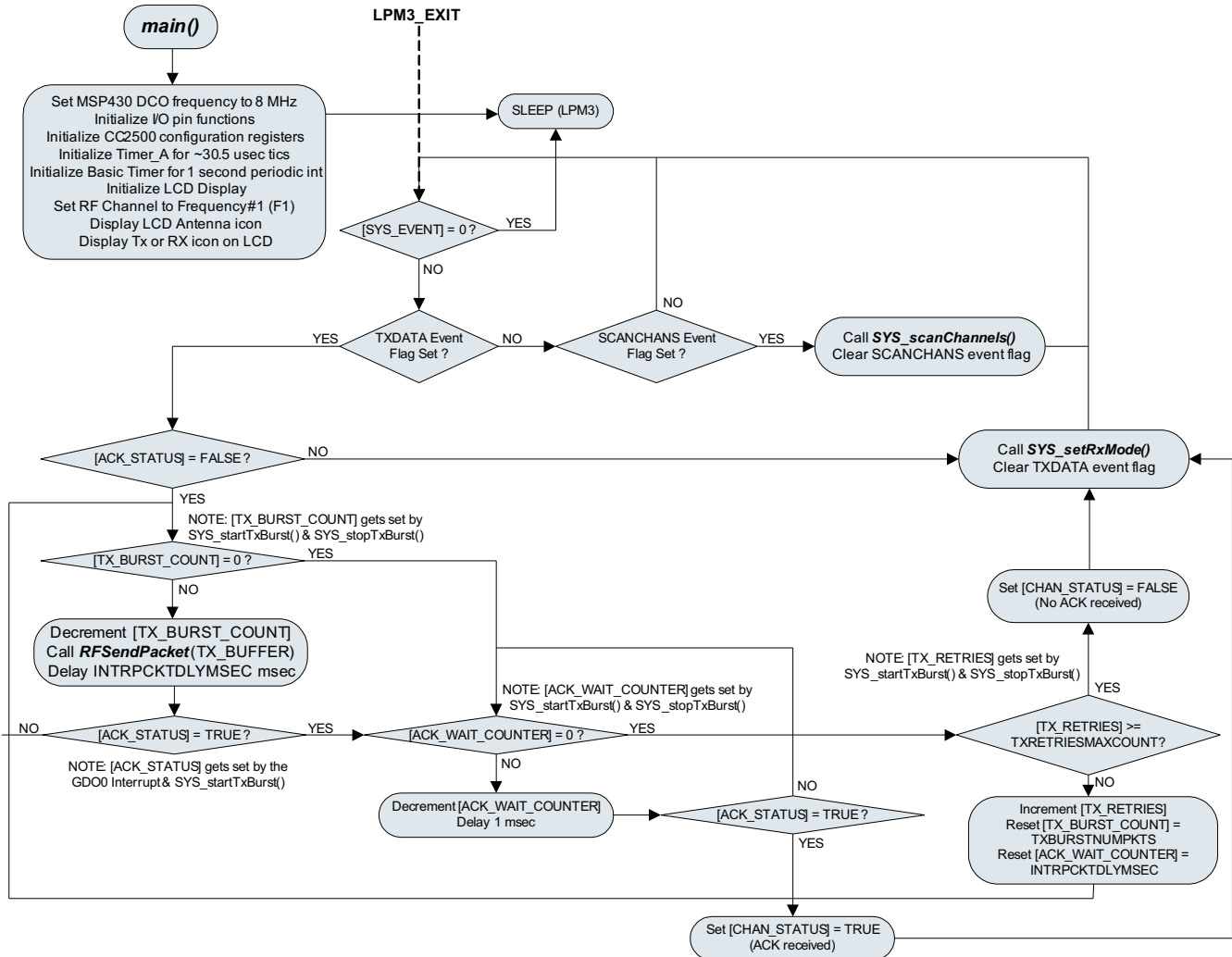


Board Power Supply Configuration
VCC_1: Lower 3 pins. Used for FG4618 / JTAG1.
VCC_2: Upper 3 pins. Used for F2013 / JTAG2.
 Jumper to "LCL" to provide local VCC to FET interface. (shown).
 Jumper to "FET" to power board from the FET interface. (In this case, the BATT jumper must not be set.)

Appendix B Software Flow Charts

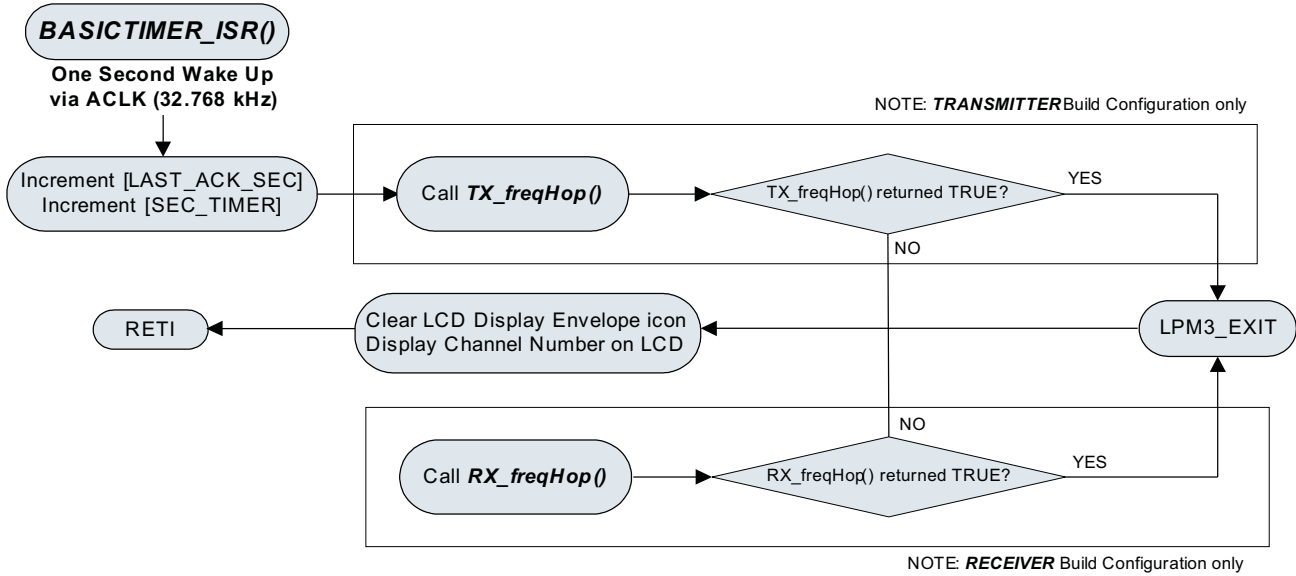
B.1 Background Loop (Main Function)

This function initializes all peripherals and then goes into LPM3. Whenever an interrupt occurs, this background loop wakes up out of LPM3 and processes the event(s). Once all event(s) have been processed, the MSP430 goes back to LPM3.



B.2 BASICTIMER_ISR()

This function executes once per second. Depending on whether the mode of operation is a Transmitter or Receiver, the corresponding frequency-hopping logic is executed.

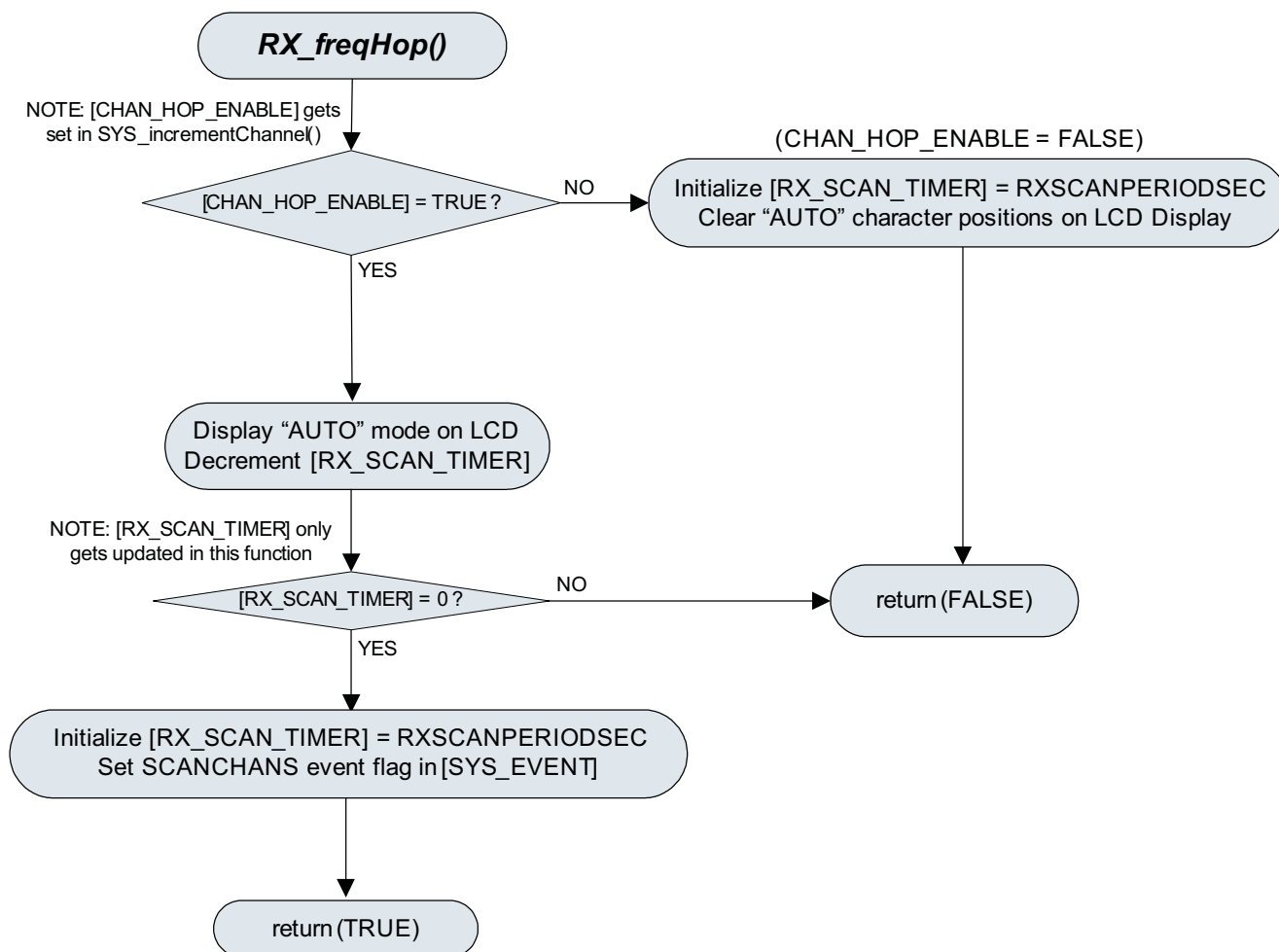


RX_freqHop()

B.3 *RX_freqHop()*

This function determines if it is time to scan the RF channels and changes frequency, if needed.

```
int RX_freqHop(void);
// Usage:      Determines if a channel scan and possible frequency hop needs to be
//             executed
// Parameters:  none
// Returns:    TRUE if channel scan needs to be initiated
```

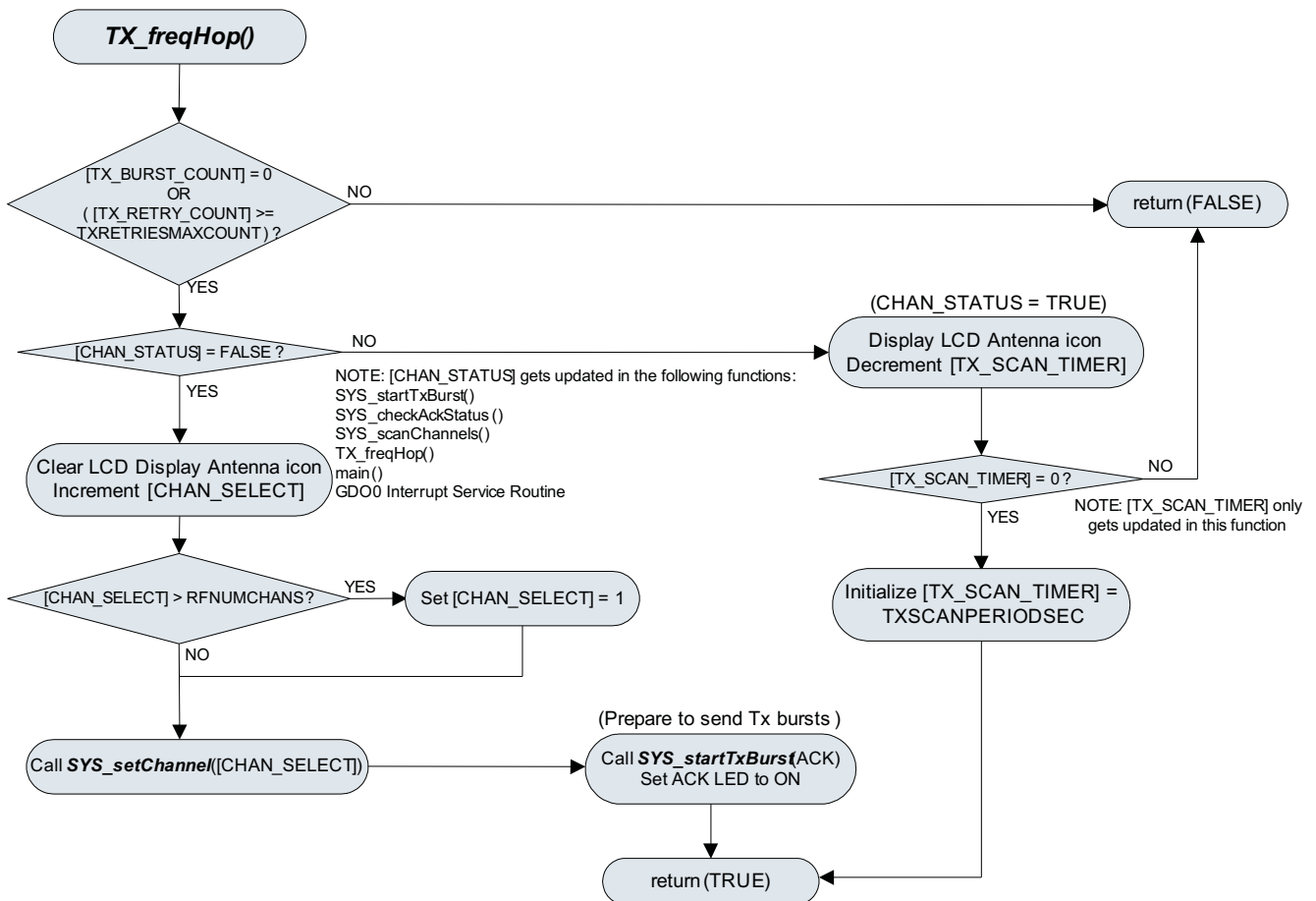


B.4 TX_freqHop()

This function determines if it's time to scan the RF channels and changes frequency, if needed. A transmit beacon is sent out to the Receiver; if no acknowledge is received within the specified time interval, cycle through the RF channels until a response is received for the transmit beacon.

```

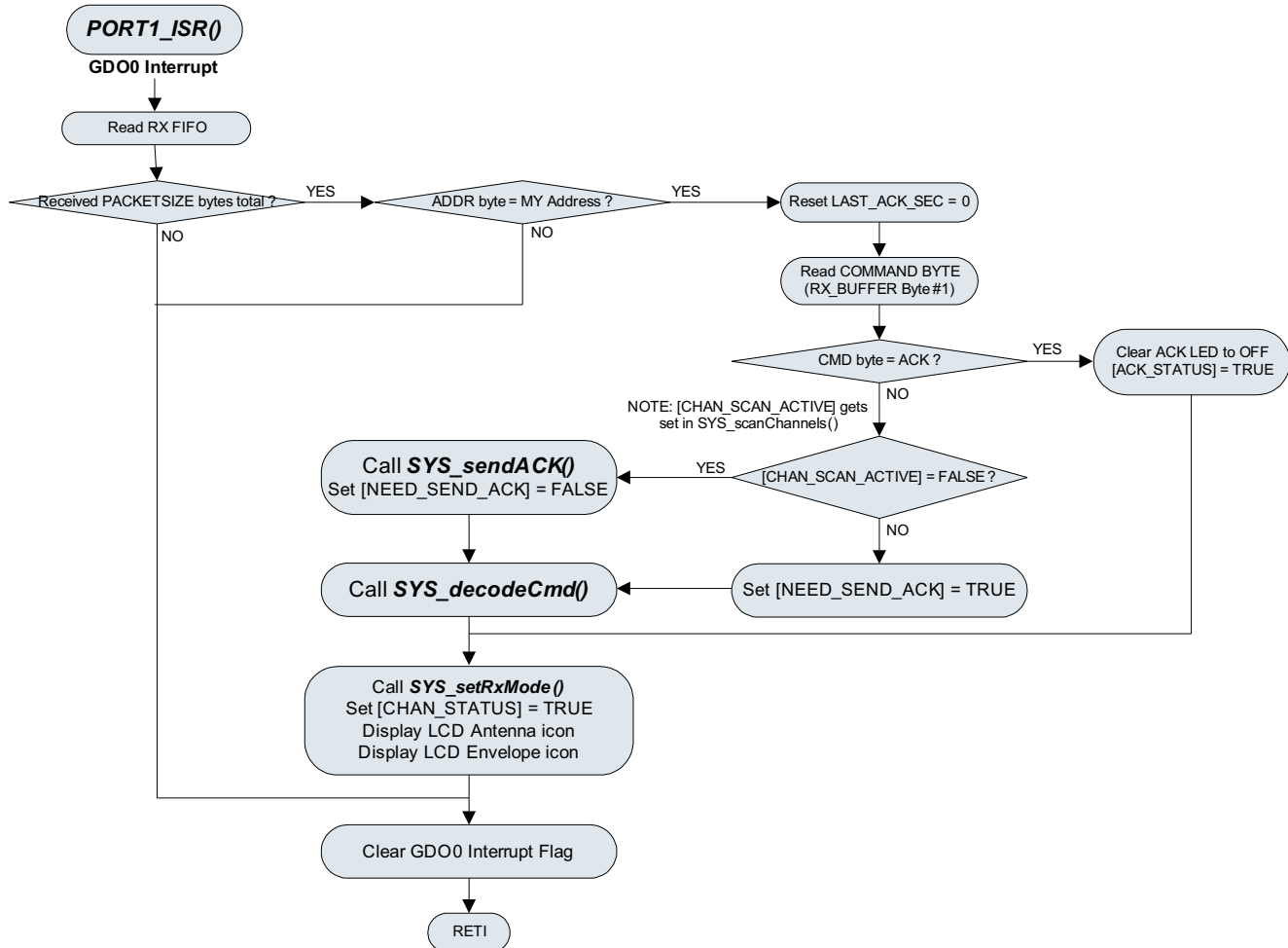
int TX_freqHop(void);
// Usage:      Sends out a beacon and determines if channel needs to be changed to
//             look for the Receiver
// Parameters: none
// Returns:    TRUE if channel needs to be changed to look for the Receiver
  
```



PORT1_ISR()

B.5 PORT1_ISR()

This function is executed whenever a Port1 interrupt (via CC2500 GDO output) has been triggered. The incoming data is scanned to see if it is a valid packet, containing the Receiver's address. If so, then process the command and send back an acknowledgement to the sender.

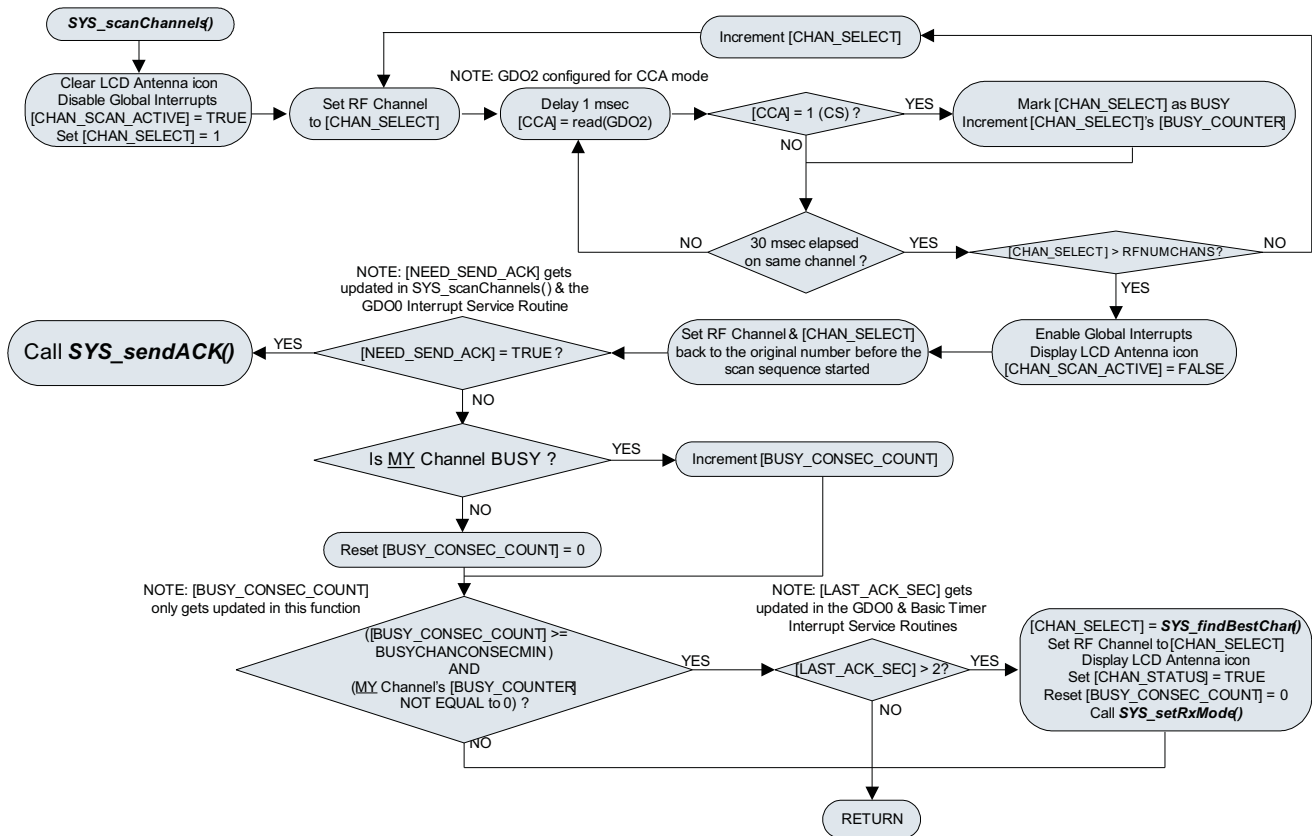


B.6 SYS_scanChannels()

This function scans all the predefined RF channels and determines which channels are clear.

```

void SYS_scanChannels(void);
// Usage:      Scans through all channels to assess Carrier Sense (RF activity) on
//              each one
// Parameters:  none
// Returns:    nothing
  
```



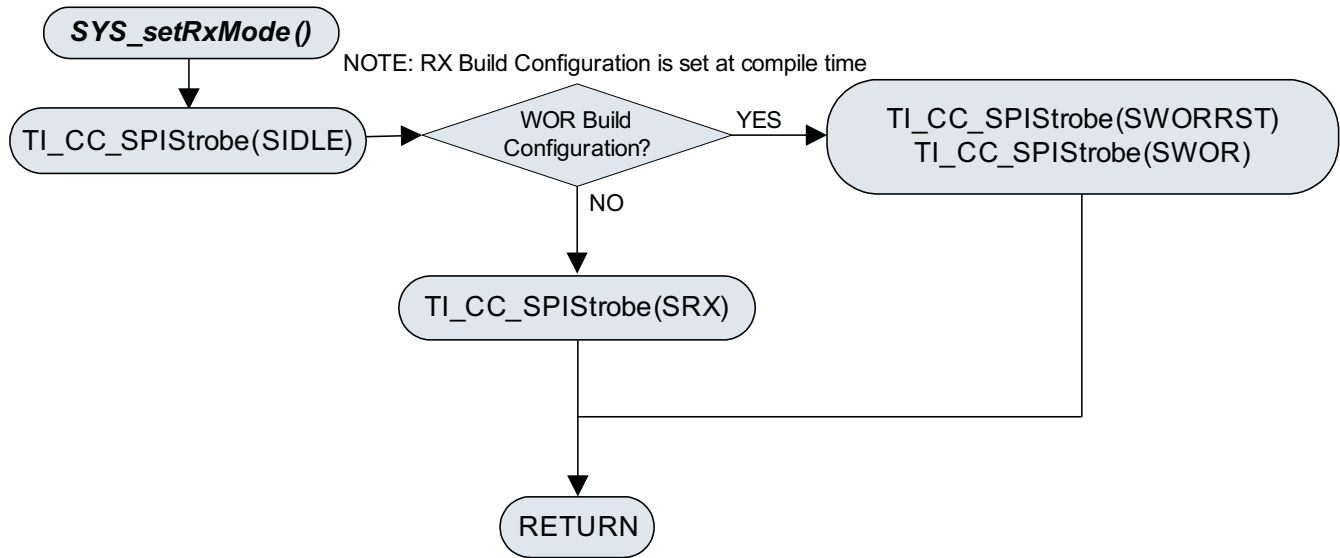
SYS_setRxMode()

B.7 *SYS_setRxMode()*

This function sets the TI CCxxx device to either WOR mode or standard RX mode, depending on the build configuration that was chosen.

```

void SYS_setRxMode(void);
// Usage:      Sets the radio to the pre-determined RX mode
// Parameters: none
// Returns:    nothing
  
```

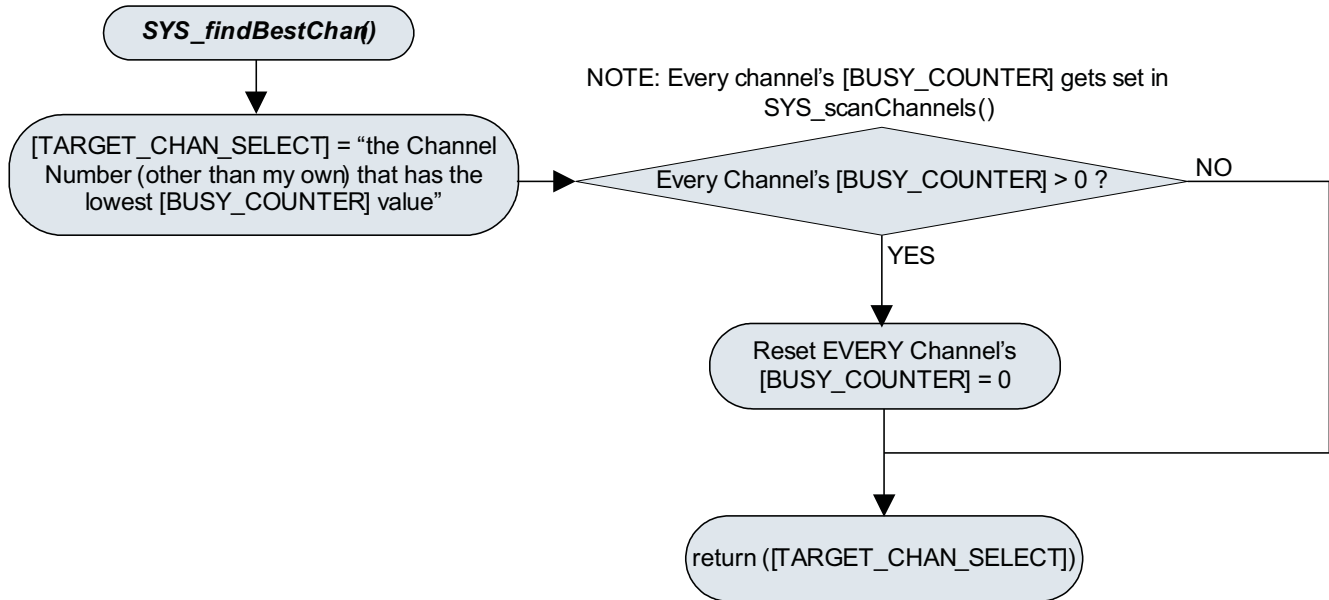


B.8 SYS_findBestChan()

This function looks for the channel with the smallest number of busy counts. If every channel's busy count is greater than zero, all busy counters are reset to zero.

```

int SYS_findBestChan(void);
// Usage:      Searches for the best channel to jump to, based on its history of
//             channel clarity
// Parameters:  none
// Returns:    Recommended channel number to jump to
  
```

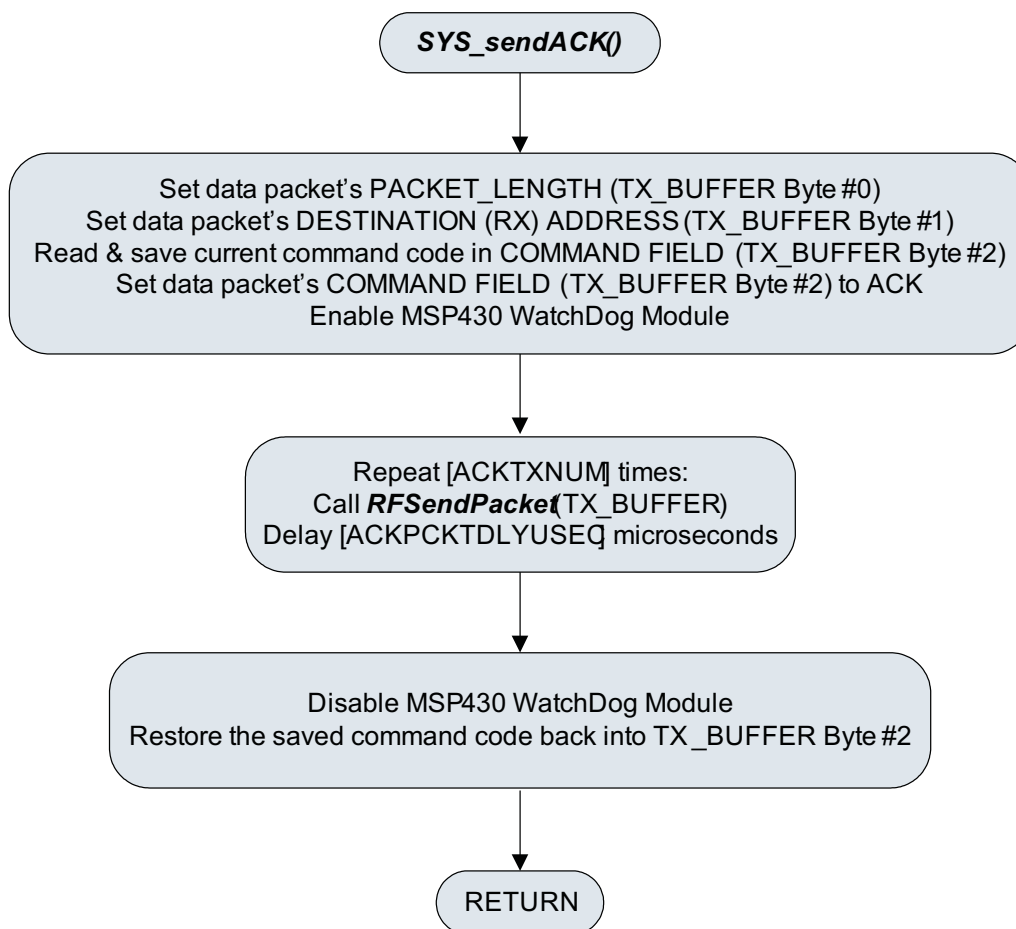


SYS_sendACK()

B.9 SYS_sendACK()

This function sets up the CC2500 to send an acknowledge packet back to the sender.

```
void SYS_sendACK(void);
// Usage:      Sets up the Tx Buffer to send out an ACK data packet multiple times
//             in a row
// Parameters:  none
// Returns:    nothing
```

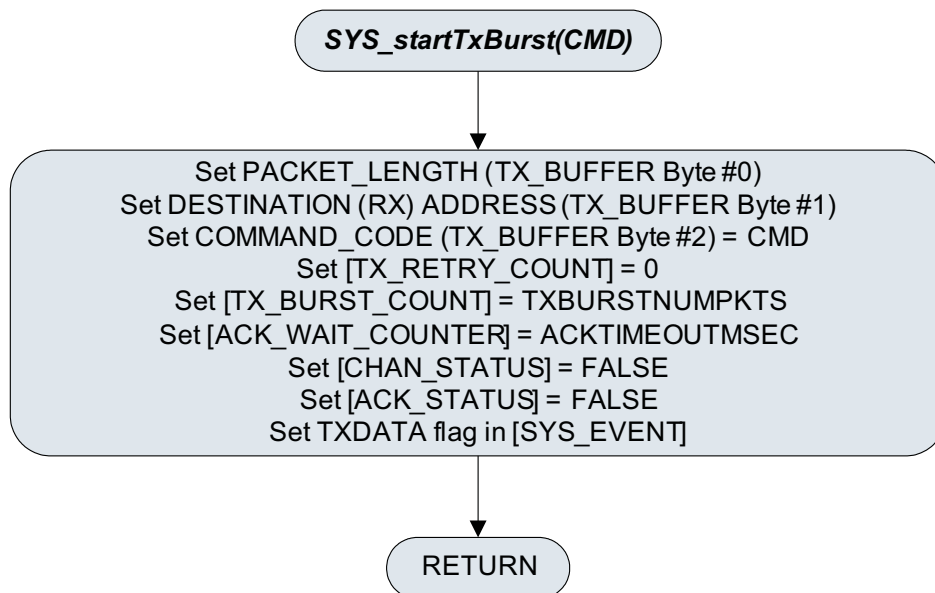


B.10 SYS_startTxBurst()

This function sends out a repeated burst transmission for a single data packet.

```

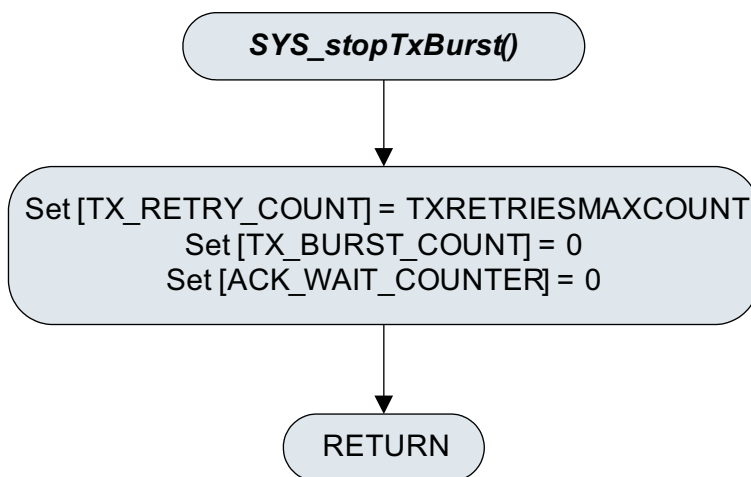
void SYS_startTxBurst(unsigned char dst, unsigned char cmd);
// Usage:      Prepares the TX Buffer & initializes variables required to start Tx
data bursts
// Parameters: [Destination (RX) address | Command code]
// Returns:    nothing
  
```



B.11 SYS_stopTxBurst()

This function resets all variables to stop any pending data burst transmission in progress.

```
void SYS_stopTxBurst(void);
// Usage:      Sets all variables to abort any Tx burst sequence that is currently
//             in progress
// Parameters:  none
// Returns:    nothing
```

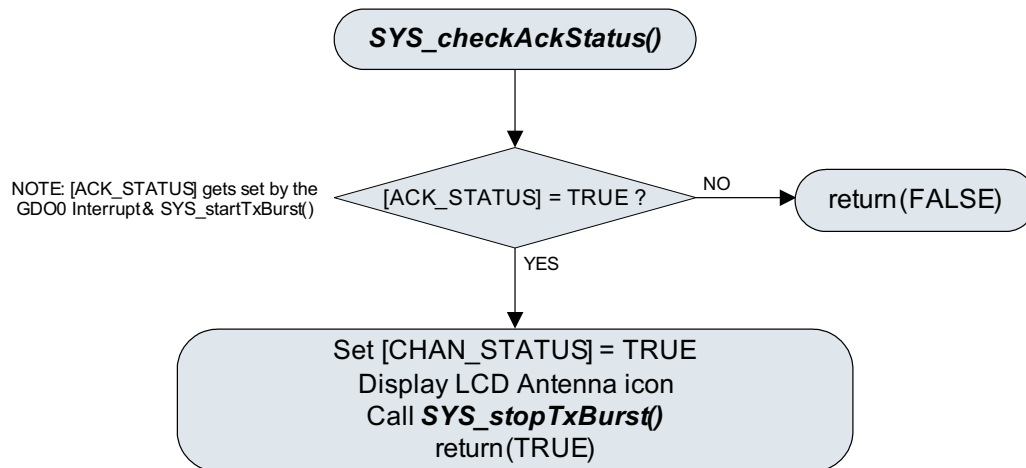


B.12 SYS_checkAckStatus()

This function checks to see if an acknowledge for the last transmitted data packet has been received.

```

int SYS_checkAckStatus(void);
// Usage:      Checks if an ACK has been recently received
// Parameters: none
// Returns:    TRUE if ACK has been received; FALSE if no ACK rx'd since last
              transmission
  
```



IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		Applications	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2007, Texas Instruments Incorporated