



# **CC2530ZNP Mini Kit Sensor Monitor Network Sample Application**

Document Number: SWRA373

**Texas Instruments, Inc.**  
Rochester, Minnesota USA



| Version | Description     | Date       |
|---------|-----------------|------------|
| 1.0     | Initial release | 06/04/2011 |

## TABLE OF CONTENTS

|                                                           |           |
|-----------------------------------------------------------|-----------|
| <b>1. INTRODUCTION .....</b>                              | <b>1</b>  |
| <b>2. ABBREVIATIONS.....</b>                              | <b>2</b>  |
| <b>3. SENSOR MONITOR NETWORK APPLICATION .....</b>        | <b>3</b>  |
| 3.1 SENSOR MONITOR NETWORK APPLICATION DESCRIPTION.....   | 3         |
| 3.1.1 Startup and Configuration.....                      | 3         |
| 3.1.2 ZigBee Device Types.....                            | 4         |
| 3.1.3 Summary of LED States .....                         | 5         |
| 3.1.4 Sensor Monitor Network Application Role .....       | 5         |
| 3.2 SENSOR MONITOR NETWORK DETAILED CODE DESCRIPTION..... | 6         |
| 3.3 COMPILE OPTIONS .....                                 | 9         |
| 3.3.1 NV_Restore .....                                    | 9         |
| 3.3.2 HOST_MT.....                                        | 10        |
| 3.3.3 Z-TOOL .....                                        | 11        |
| <b>4. SENSOR MONITOR NETWORK GUI.....</b>                 | <b>15</b> |
| 4.1 SYS_PING.....                                         | 15        |
| 4.2 SENSOR MONITOR NETWORK MESSAGES.....                  | 16        |
| <b>5. UPGRADING CC2530 ZNP IMAGE .....</b>                | <b>16</b> |
| <b>6. FREQUENTLY ASKED QUESTIONS.....</b>                 | <b>18</b> |
| <b>7. REFERENCES .....</b>                                | <b>19</b> |

## LIST OF FIGURES

|                                                                                                                                                                                                                                                        |    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| FIGURE 1: (A) SINK AND (B) SOURCE DEVICES DEFINED BY THE SENSOR MONITOR NETWORK SAMPLE APPLICATION ....                                                                                                                                                | 3  |
| FIGURE 2: SUMMARIZES THE ZIGBEE DEVICE ROLES IN THE APPLICATION AND THE NETWORK. ....                                                                                                                                                                  | 4  |
| FIGURE 3: SHOWING NETWORK OPERATION AND OVER THE AIR MESSAGES (AFTER NETWORK JOIN OPERATION)<br>ILLUSTRATING BINDING AND THEN SOURCE DEVICES PERIODICALLY REPORTING THE DATA AND RECEIVING<br>ACKNOWLEDGEMENT FROM THE SINK DEVICE. ....               | 5  |
| FIGURE 4: STATE MACHINE DIAGRAM FOR SENSOR MONITOR NETWORK APPLICATION WITHOUT NV_RESTORE<br>COMPILE FLAG ENABLED .....                                                                                                                                | 8  |
| FIGURE 5: SHOWING CODE SNIPPET .....                                                                                                                                                                                                                   | 9  |
| FIGURE 6: STATE MACHINE DIAGRAM FOR SENSOR MONITOR NETWORK APPLICATION WITH NV_RESTORE COMPILE<br>FLAG ENABLED .....                                                                                                                                   | 10 |
| FIGURE 7: EZ430-CC2530ZNP MINI KIT BOARD CONNECTED TO THE PC .....                                                                                                                                                                                     | 11 |
| FIGURE 8: DEVICE MANAGER VIEW OF THE PC TO IDENTIFY COM PORT NUMBER OF CONNECTED DEVICES .....                                                                                                                                                         | 12 |
| FIGURE 9: STEPS ILLUSTRATING THE HOW TO CONFIGURE THE CONFIGURATION PARAMETERS FOR THE COM PORT FOR<br>COMMUNICATION WITH Z-TOOL APPLICATION. ....                                                                                                     | 13 |
| FIGURE 10: Z-TOOL APPLICATION SHOWING TWO DEVICES CONNECTED AND EXPOSED INTERFACES – SYSTEM, AF,<br>ZDO, SIMPLE API, UTIL, AND APP ON THE CC2530ZNP ON THE CC2530ZNP MINI KIT.....                                                                     | 13 |
| FIGURE 11: Z-TOOL APPLICATION WINDOW WITH TWO CONNECTED DEVICES ON COM50 AND COM31. DEVICE ON<br>COM50 REQUESTED IEEE ADDRESS FROM THE DEVICE AT ADDRESS 0x0000 (COORDINATOR) AND GETS THE<br>RESPONSE BACK OF TYPE IEEE_ADDR_REQ_SRSP. ....           | 14 |
| FIGURE 12: SENSOR MONITOR NETWORK GUI REPRESENTING COORDINATOR-SINK DEVICE AS RED CIRCLE, ROUTER-<br>SOURCE AS BLUE CIRCLE AND END-DEVICE-SOURCE AS YELLOW CIRCLE WITH TEMPERATURE, NETWORK<br>ADDRESS AND TIME STAMP OF THE LAST PACKET RECEIVED..... | 15 |
| FIGURE 13: FRAME FORMAT FOR SYS_PING_RESPONSE MESSAGE TO THE GUI.....                                                                                                                                                                                  | 15 |
| FIGURE 14: FRAME FORMAT FOR THE OTA SENSOR MONITOR NETWORK MESSAGES .....                                                                                                                                                                              | 16 |
| FIGURE 15: CC-DEBUGGER CONNECTED TO CC2530ZNP MINI KIT BOARD. GREEN LIGHT INDICATES PROPER<br>CONNECTION .....                                                                                                                                         | 16 |

|                                                                                                                     |    |
|---------------------------------------------------------------------------------------------------------------------|----|
| FIGURE 16: SMARTRF FLASH PROGRAMMER WINDOW .....                                                                    | 17 |
| FIGURE 17: SMARTRF FLASH PROGRAMMER WINDOW WITH STATUS MESSAGE SHOWING SUCCESSFUL OPERATION<br>STATUS MESSAGE ..... | 17 |

## LIST OF TABLES

|                                                                                                |   |
|------------------------------------------------------------------------------------------------|---|
| TABLE 1: LED STATES BASED ON DEVICE CONFIGURATION FOR SENSOR MONITOR NETWORK APPLICATION ..... | 5 |
|------------------------------------------------------------------------------------------------|---|

## 1. INTRODUCTION

This document explains the sensor monitor network application for the CC2530ZNP Mini Kit development platform. The sensor monitor network application is comprised of two types of devices: Sources and a single Sink, operating in potentially all three ZigBee roles: Coordinator, Router, and End Device. Sources provide temperature and battery voltage readings collected by the MSP430F2274 using its on-chip temperature sensor and A2D converter periodically to the Sink.

The sensor monitor network application can also optionally utilize the non-volatile restore feature, where a device can form/join a ZigBee network using the previous network configuration values in its non-volatile memory, for example in cases of power fluctuation, battery run-out condition, etc. For this functionality the NV\_Restore compile flag has to be enabled. This document first explains the application without the NV\_Restore compile flag's being enabled. Section 3.3.1 then explains the operation with NV\_Restore enabled.

The document also explains in section 3.3.3 how the Z-Tool application can be used with the CC2530ZNP Mini Kit development platform by enabling the compile flags "HOST\_MT" and "ZTOOL", using this sample application for ZigBee network development and debugging.

It is recommended to use the ZNP image built using the Z-Stack-2.5.0 release with this sample application and new design development. Section 5 explains how to program the CC2530 with a new ZNP image.

This document does not focus on the use of the accompanying sensor monitor network visualizer GUI or the ZNP interface. For this developers are recommended to refer to the GUI reference document [7] and ZNP Interface specification document [3].

IAR or Code Composer Studio can be used to download, test and debug this application. IARv5.20.1 was used for development of this sample application. Using an earlier version can cause linker errors, and it is recommended to use the IARv5.20.1 or later for development with this application.

You can order the CC2530ZNP-Mini-Kit from [5]

For additional details for the ZNP-Mini-Kit refer to [1] and for additional sample applications refer to [2].

## 2. ABBREVIATIONS

|        |                                                                |
|--------|----------------------------------------------------------------|
| AF-ZDO | Application Framework/Zigbee Device Object Interface           |
| APP    | Application                                                    |
| APSME  | Application Support Sub-Layer Management Entity                |
| APSD   | Application Support Sub-Layer Data Entity                      |
| APSSE  | Application Support Sub-Layer Security Entity                  |
| A2D    | Analog to Digital                                              |
| FFD    | Full Function Device (Coordinator or Router in Zigbee Network) |
| GUI    | Graphical User Interface                                       |
| HAL    | Hardware Abstraction Layer                                     |
| MT     | Monitor and Test                                               |
| MLME   | MAC Layer Management Entity                                    |
| MCPS   | MAC Common Part Service                                        |
| NV     | Non Volatile                                                   |
| NLME   | Network Layer Management Entity                                |
| NLDE   | Network Layer Data Entity                                      |
| NLSE   | Network Layer Security Entity                                  |
| OTA    | Over the Air                                                   |
| PD     | PHY Data                                                       |
| PLME   | PHY Layer Management Entity                                    |
| RFD    | Reduced Function Device (End Device in ZigBee Network)         |
| SAPI   | Simple Application Programming Interface                       |
| SAP    | Service Access Point                                           |
| ZNP    | Zigbee Network Processor                                       |

### 3. SENSOR MONITOR NETWORK APPLICATION

### 3.1 Sensor Monitor Network Application Description

### 3.1.1 Startup and Configuration

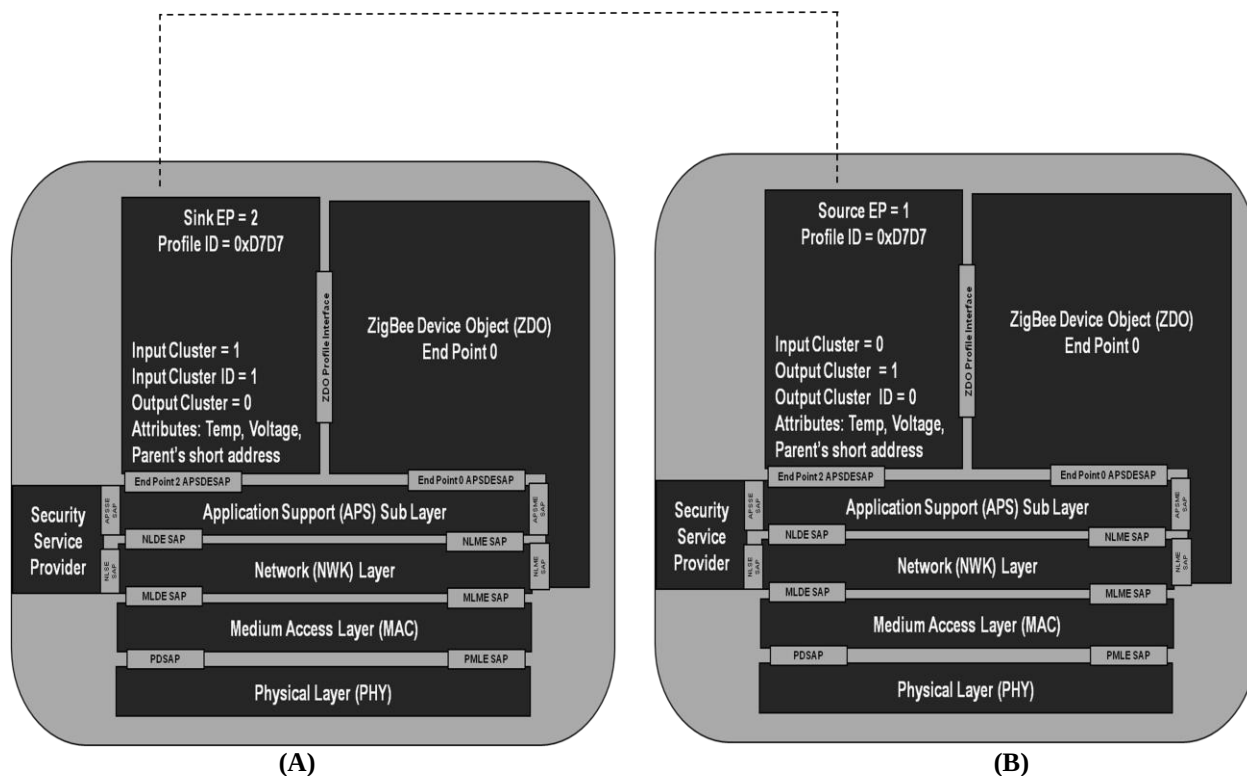
## Startup

Upon powering up the device will blink both the RED and GREEN LEDs awaiting configuration.

## Configuring the Device Type

A device can be configured in two ways: using one button press or two button presses (within a two second interval). Pressing the button one time in two seconds will first attempt to join the device as a Router-Source in an existing network and, if no network exists, it will then create a new network by configuring itself as the Coordinator-Sink. From the application perspective only a single Sink, the Coordinator, will reside in a network so that all Source reports are sent to a single device. Pressing the button two times in two seconds will configure the device as an End Device-Source.

Please make sure a Coordinator has successfully created a network before proceeding to configure additional devices.



**Figure 1: (A) Sink and (B) Source devices defined by the sensor monitor network sample application**

The sample application defines a single endpoint, endpoint two on the Sink and end-point one on the Source. Depending on the configuration, the application running on this endpoint provides the device with functionality of either the Sink or the Source device on the network. Figure 1 above describes the architecture of the two types of ZigBee devices on the network and details important parameters of the defined endpoints. Description of various layers and interfaces can be found in ZigBee specification document [8].



The sample application implements a Manufacturer Specific/Private Profile type network. It defines a example profile Id = 0xD7D7 for the example. Before field deployment of the network, developers need to obtain the profile Id from the ZigBee Alliance. For the defined profile, the sample application defines a single cluster, cluster ID = 0x0001 for the profile. On the sink device this cluster is of type input, while on source devices this is an output cluster. As described in section 3.1.4, when a source joins the network it binds its application endpoint to the Sink Endpoint. For the application, three attributes for the cluster have been defined – temperature, voltage and parent's short address. The values of these attributes are transmitted in each periodic report (message) from the source to the sink.

Please refer to section 3.2 for understanding the sample application code in detail.

### 3.1.2 ZigBee Device Types

#### Coordinator

The Coordinator, upon initialization, will display the RED LED as always on. The button on the Coordinator will allow the user to toggle whether joining is permitted, with the RED LED continuously ON when joining is allowed, and blinking at 1 Hz when joining is not allowed. The Coordinator will act as a Sink at the application layer.

#### Router

The Router, once joined to a network, will display the GREEN LED as always on. The button on the Router will allow the user to toggle whether joining is permitted, with the GREEN LED continuously ON when joining is allowed, and blinking at 1 Hz when joining is not allowed. The Router will act as a Source at the application layer.

#### End Device

The End Device, once joined to a network, will blink the GREEN LED at 1 Hz to show it is connected to the network. The End Device will act as a Source at the application layer.

Figure 2 also summaries how the network devices can be classified: by the application, by the logical device type on the network and according to the IEEE 802.15.4 definition.

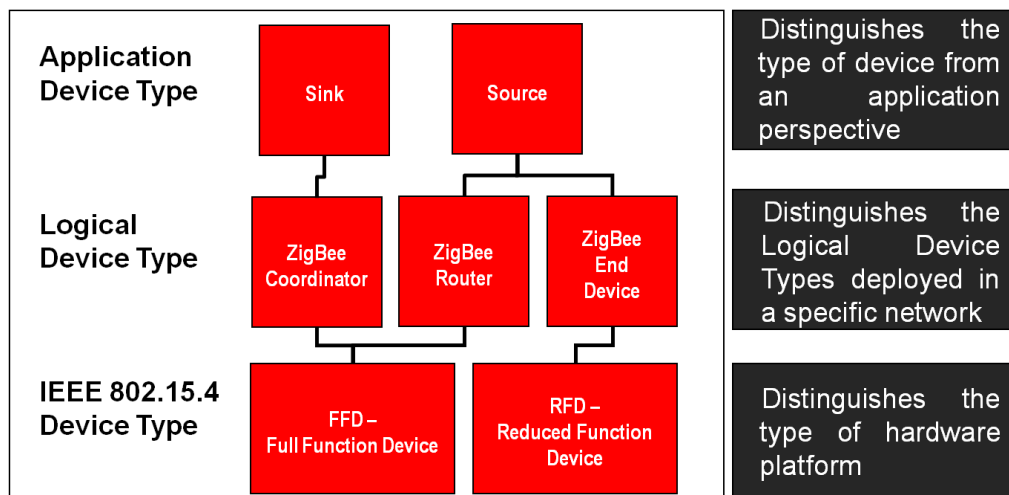


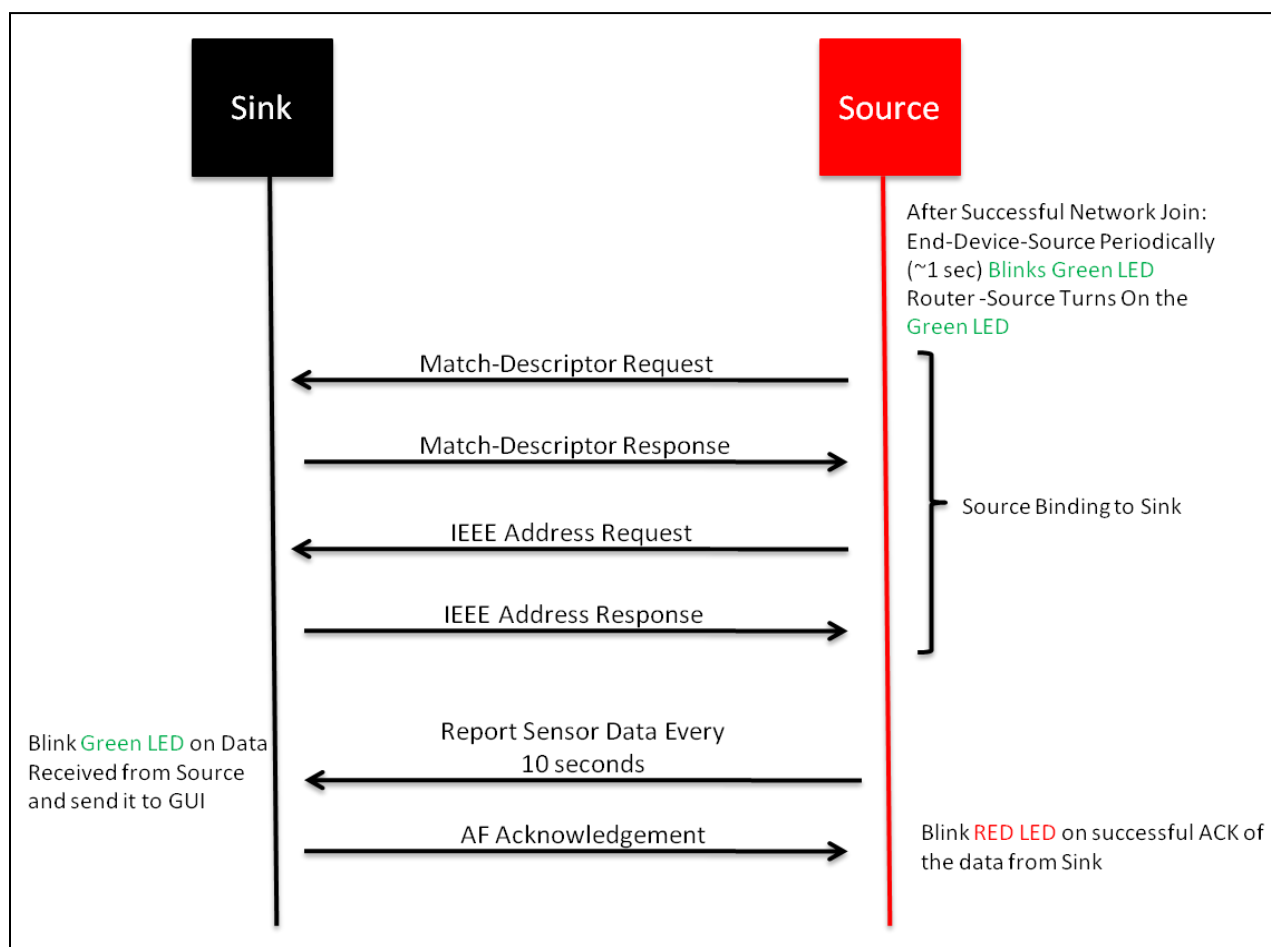
Figure 2: Summarizes the ZigBee Device roles in the application and the network.

### 3.1.3 Summary of LED States

| Device Type         | RED LED                                                    | GREEN LED                                                  |
|---------------------|------------------------------------------------------------|------------------------------------------------------------|
| Not Configured      | Blinking                                                   | Blinking                                                   |
| Coordinator – Sink  | ON - Joining Permitted<br>Blinking - Joining Not Permitted | Blink (once) - Source Report Received                      |
| Router – Source     | Blink (once) - Data Send Ack'ed                            | ON - Joining Permitted<br>Blinking - Joining Not Permitted |
| End - Device Source | Blink (once) - Data Send Ack'ed                            | Blinking - Joined to Network                               |

**Table 1: LED States based on device configuration for sensor monitor network application**

### 3.1.4 Sensor Monitor Network Application Role



**Figure 3: Showing Network operation and over the air messages (after network join operation) illustrating binding and then Source devices periodically reporting the data and receiving acknowledgement from the Sink device.**

## Sink

The Sink will allow binding by a Source at any given time. When data is received from a Source, the Sink will blink the GREEN LED once to show data has been received. If connected to a PC running the eZ430-RF2530 Sensor Monitor Network, the Sink will also send the received data report to the PC for display on the GUI [4].

## Source

Once joined, the Source will initiate binding to discover (Figure 2) and establish a connection with the network Sink. Once bound, the Source will send periodic sensor reports. These reports contain temperature and battery voltage information for the Source. Reports will be sent once every 10 seconds. For reliable data communication, data reports by the Source will utilize application level acknowledgements. The Source will blink the RED LED every time data is successfully acknowledged by the Sink.

### 3.2 Sensor Monitor Network Detailed Code Description

The sensor monitor network application demonstrates a simple ZigBee network implementation using the CC2530ZNP Mini-Kit. Although the ZNP supports an extensive range of functionality through a complete set of API calls documented in the CC2530ZNP Interface Specification document, this sample application only utilizes the subset of ZigBee functionality provided by the Simple API, also described in [3]

The Sample Application code is organized into the following 5 groups of modules:

APP - Application: This is the application-specific code that implements the sensor monitor network application functionality.

HAL - Hardware Abstraction Layer: This is the host processor specific code that implements the device drivers required for the SPI; the hardware timer that drives the virtual software timers; and the method of receiving manual user input and implementing low power operation.

MT - Monitor-Test: this code provides methods to communicate with a PC Tool.

SAPI - Simple API: this body of code provides methods that implement the CC2530ZNP Simple Application Programming Interface (Simple API).

ZNP – Zigbee Network Processor: This body of code contains library methods that implement the CC2530ZNP interface specification as well as error checking and return values.

The code transitions through a series of states depending on user input and CC2530ZNP feedback. The state is tracked by following C-code variable:

```
static AppState appState;
```

The possible states are defined by this C-code enumeration –

```
typedef enum {  
    appIniting,  
    appWaiting,  
    appJoining,  
    appJoinWaiting,  
    appBinding,  
    appBindWaiting,  
    appRunning  
} AppState; // Valid appState values.
```

Following section describes various states the application works in and the device's function in these states.

#### applniting State – Source/Sink

With NV\_Restore disabled after power up, this Sample Application (the host) is in the state applniting, which is to say, it is waiting to initialize the CC2530ZNP. After the CC2530ZNP resets, it sends a SYS API message to indicate successful reset. The host is waiting for this indication when it is in the applniting state. Upon receiving the reset indication the host sets the NV option to clear the previous configuration (if any) and resets the CC2530ZNP. A reset is required for the NV write to take effect. The device then transitions to the appWaiting state.

#### appWaiting / appoining / appJoinWaiting State – Source/Sink

When the host is in appWaiting state, both LEDs will blink at 1-Hz. The user can configure the device as follows:

One momentary button push in a two-second window configures the device to be an FFD.

Two momentary button pushes in a two-second window configures the device to be an RFD.

When the two-second window expires, the device is “configured” and transitions to the appJoining state. An FFD will try to join an existing network as a ZigBee Router for 10 seconds. If it fails to join, it will start a new network as a ZigBee Coordinator. An RFD will try to join an existing network as a ZigBee End Device for 10 seconds. If it fails to join, it will stop, the state will transition to appJoinWaiting, and it will enter low power for 30 seconds, and try again – repeating the sequence until it succeeds in joining. In the appJoining or appJoinWaiting state, the LEDs will be off.

#### appRunning State – Sink

When an FFD starts a new network as a ZigBee Coordinator, it will set the red LED solid on and allow devices to bind – the state will transition to appRunning and its sub-state will be “allowing joins” as set by the flag appPermittingF in the variable appFlags. The device also maintains a tertiary state, that it is sinking data, as set by the flag appSinkingF in appFlags. Whenever a message is acknowledged, the device blinks the green LED once. The sub-state of allowing joining can be toggled on the Coordinator by a momentary button press. The sub-state transitions to not allowing joining by clearing the appPermittingF, and the red LED starts blinking at 1-Hz.

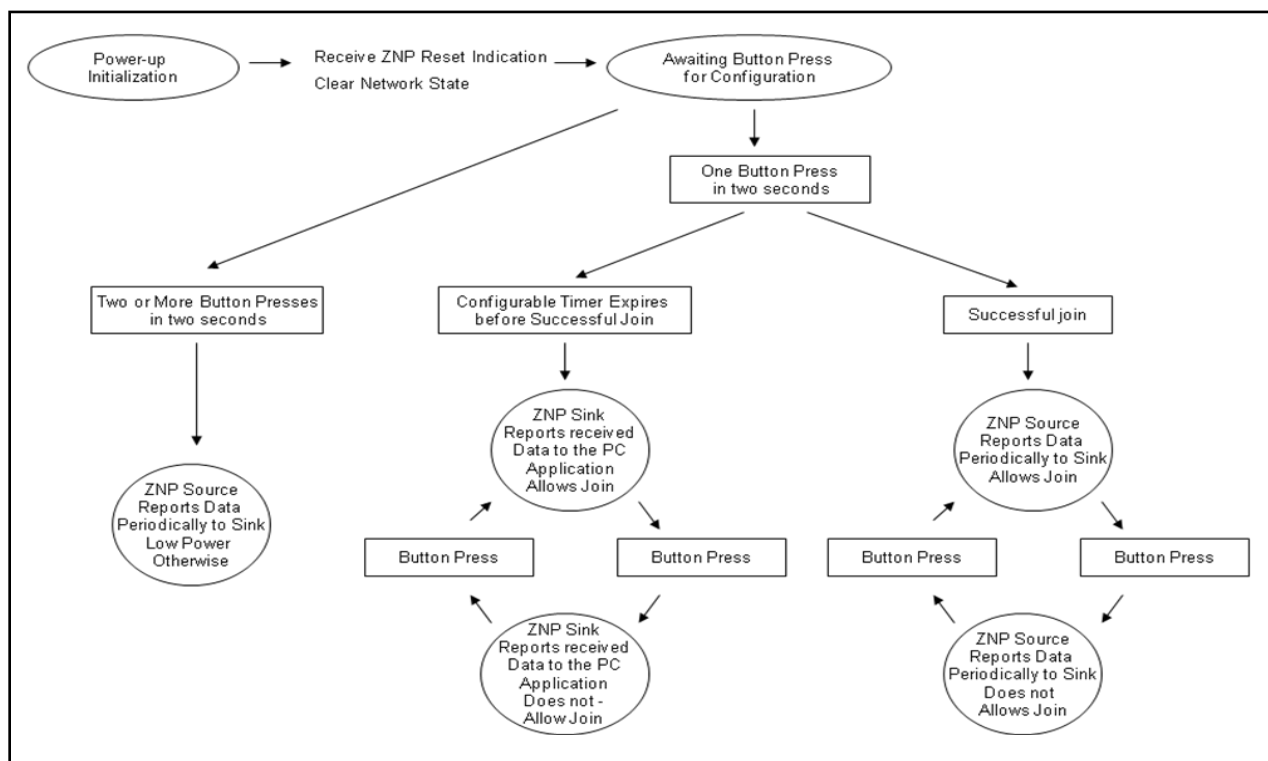
#### appBinding / appBindingWait / appRunning State - Source

When an FFD successfully joins an existing network as a ZigBee Router, its state transitions to appBinding, and the device sets the green LED solid on. The device uses the Simple API to find the device that is sinking data. The FFD will continue to request a bind from the SAPI every 30 seconds until it is successful. Upon successful binding the state will transition to appRunning and its sub-state will be “allowing joins” as set by the flag appPermittingF in the variable appFlags. The sub-state of allowing joining can be toggled on the Router by a momentary button press. The sub-state transitions to not allowing joining by clearing the appPermittingF and the green LED starts blinking at 1 - Hz.

Every ten seconds, the FFD Source device sets a tertiary state, that it is sending data, as set by the flag appSendingF in appFlags. The device requests that the sinking device should acknowledge receipt of the data message. Whenever a message is acknowledged, the device blinks the red LED once. If the application does not receive acknowledgement it retries sending the same data up to 3 times. When the message is acknowledged or the retries have expired, the sub-state of sending data is cleared by clearing the appSendingF flag. If the retries have expired the device then tries to bind again.

When an RFD successfully joins an existing network as a ZigBee End Device, its state transitions to `appBinding`, and the device starts blinking the green LED at 1-Hz. The device uses the SAPI to find the device that is sinking data. The RFD will wait up to 10 seconds for a bind request to succeed and then transition the state to `appBindingWait` and enter low power for 30 seconds. The device will continue to alternate between the two binding states and low power every 30 seconds until it is successful. Upon successful binding the state will transition to `appRunning` and its sub-state will be “allowing low power” as set by the flag `appLowPwrF` in the variable `appFlags`.

Every ten seconds, the RFD Source device sets a tertiary state that it is sending data, as set by the flag `appSendingF` in `appFlags`. The device requests that the sinking device should acknowledge receipt of the data message, so it retries sending the same data OTA up to three times every time that the SAPI feedback from the CC2530ZNP indicates that the message has not been acknowledged. While waiting for the acknowledgement, the device enters low power and awakens every two seconds by a timer in order to poll for the SAPI feedback from the CC2530ZNP. When the message is acknowledged or the retries have expired, the sub-state of sending data is cleared by clearing the `appSendingF` flag. Whenever a message is acknowledged, the device blinks the red LED once.



**Figure 4: State machine diagram for sensor monitor network application without NV\_Restore compile flag enabled**

The first code segment to run after power-up is compiler-generated and includes such house-keeping as initializing global variables. The first sample application code segment to run is found in `sample_main.c` in this C-code function:

```
void main(void)
```

Program execution then begins with the `hallInit()` function which performs the following

- Sets direction for I/O pins, interrupts, pull-up/pull-down resistors, etc.
- Oscillator: turns off WDT, configures clocks
- Initializes timer A

Currently, `mtInit()` initializes the UART for the MT interface to talk to the sensor monitor network visualizer GUI [4].

ZNP is then initialized using the function `znpInit()`. This function initializes the SPI port for communication with CC2530ZNP and then resets the CC2530ZNP.

When `NV_RESTORE` is disabled, the program then calls `appInit()` function which performs the following

- Enables the timer for periodic LED blinks
- Enables the interrupts

```
void main(void)
{
    /* Initialize the hardware */
    halInit();

    #if HOST_MT
    /*Initialize the UART module*/
    mtInit();
    #endif

    /* Initialize the hardware interface to the ZNP (SPI) and reset the ZNP */
    znpInit();

    #ifndef NV_RESTORE
    /*If NV_RESTORE is not defined then initialize the ZNP application*/
    /*Initialize the application - reset ZNP and enable the interrupts */
    appInit();
    #else
    /*Initialize the application parameters from last configured state*/
    appSync();
    #endif

    for (;;)
    {
        /* Prioritize HAL events to maintain quasi-realtime performance on H/W related events.
         * After all HAL events are processed, service the ZNP events.
         */
        if (!appExecHal())
        {
            /* If all HAL and Host events have been processed, it is ok to go to low power.*/
            if (!appExecHost() && (appFlags & appLowPwrF))
            {
                /*Enter low power mode*/
                HAL_SLEEP();
            }
        }
    }
}
```

**Figure 5: Showing Code Snippet**

The code then loops infinitely servicing either the HAL events that are set by hardware events or the ZNP events that are set by feedback from the CC2530ZNP.

### 3.3 Compile Options

#### 3.3.1 NV\_Restore

If the `NV_RESTORE` Compile option is enabled, then on startup the sensor monitor network code in MSP430F2274 will call the function

`appSync();`

instead of `appInit()`. This function checks the PAN ID stored in NV. If it is not equal to `0xFFFF`, then the device has stored configuration parameter from its previous network join. It will then start the device with stored configuration parameters in the NV by reading the device type.

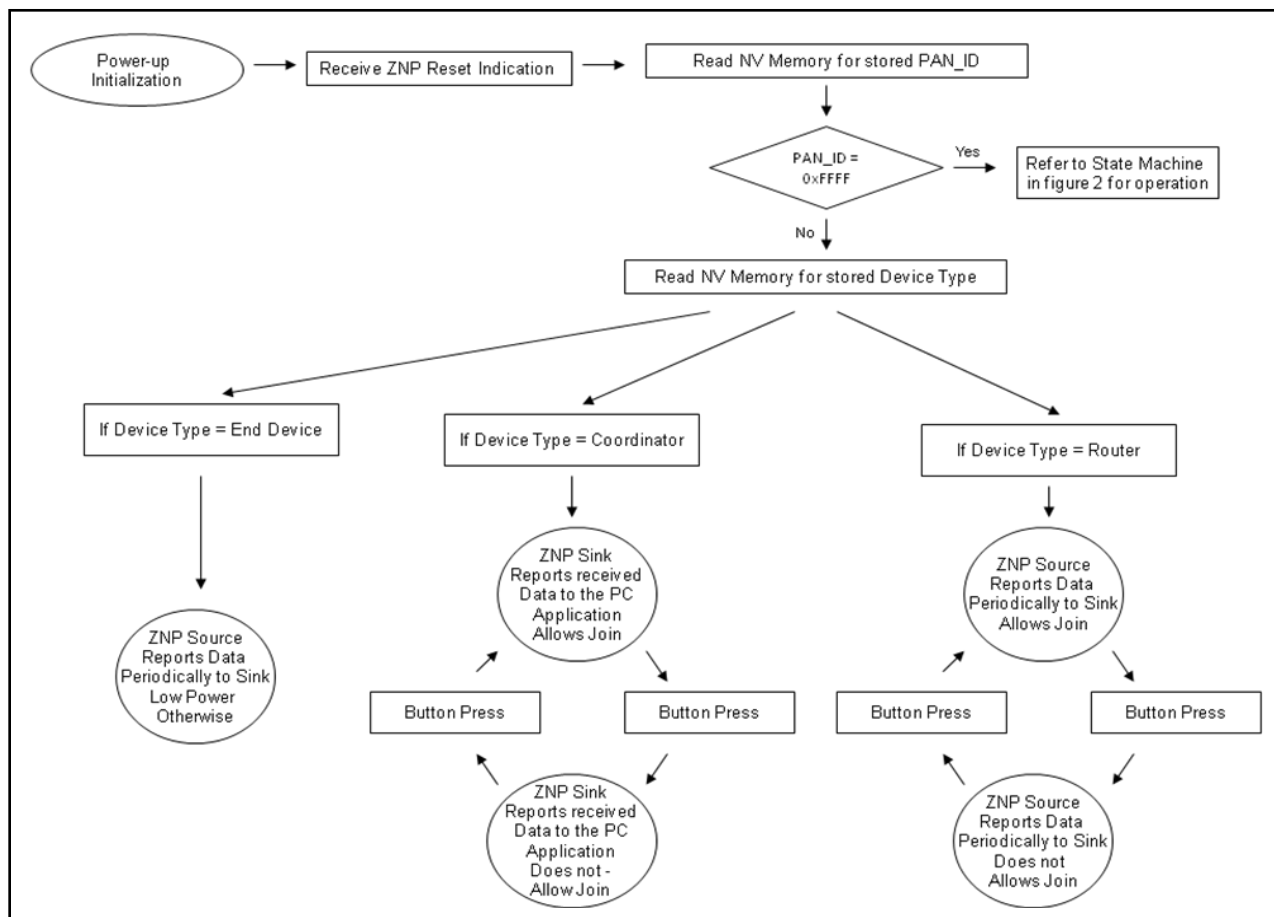
If the stored device type is coordinator it configures the device for Sink functions. The device state is initialized as `appRunning` and will work as described in this state.

If the device was either Router-Source or End-Device-Source it configures the device for Source functions where they report the sensor data every ten seconds. Also, the binding information is stored in the NV so the devices do not need to bind again. The appState after the NV\_Restore is appRunning. The devices will then work as described above in the appRunning State.

NOTE: If the previous network still exists the device will come up in the network silently as if it were always there. But if the previous network is not found and the device is an end device then this condition has not been handled and has been left on the developers; they can use the AF-ZDO interface and use the ZDOStateChange callback checking for the orphan Status message from ZNP to take appropriate action.

Additionally, if the device needs to join a new network or needs to join the network again as a different device or under different parent, etc., the CC2530ZNP needs to be reset and its configuration parameters cleared. To clear the NV configuration parameters of the CC2530ZNP and start it as a new device press the button and keep holding it down until you see both the Green and Red LED's periodically flashing at an interval of ~1sec. The device will then await the configuration inputs (button press(s)) from the user and will work as explained in previous sections).

Figure 6 below shows the state machine for the application with NV\_Restore compile flag enabled.



**Figure 6: State machine diagram for sensor monitor network application with NV\_Restore compile flag enabled**

### 3.3.2 HOST\_MT

Enable this compile option on the coordinator. This will enable the functions to allow the Sink to talk to the sensor monitor network GUI over the UART interface. Section 4.2 illustrates the message format for communication with the sensor monitor network GUI.



### 3.3.3 Z-TOOL

The Z-Tool application can be used to communicate with a Z-Stack™ target device and provides debug information via the serial port interface on the target. From the Z-Tool application different kinds of commands can be issued to the target, and responses to the commands from the target can be viewed. The different types of commands are System, MAC interface, NWK interface, ZDO interface and User Test commands. The target can be reset from Z-Tool, and memory commands allow the tester to read and write memory locations on the target RAM and Non-Volatile memory. Z-Tool can also subscribe to callbacks at the target and write out the callback information when the callback event happens.

To use the Z-Tool application using the Sensor Monitor Network Application, enable the compile flags HOST\_MT and ZTOOL. When these compile flag are enabled the ping response from the application is not modified, as required for the Sensor Network Monitor GUI application and the ping response will be sent to the PC as received from the ZNP. This allows Z-Tool to recognize the ZNP capabilities and enables the different exposed interfaces (System, AF, ZDO, Simple API, Util and App). Please refer to [3] to understand the various commands and responses in these interfaces. The Z-Tool application enables faster network development and debugging.

Follow the steps below to interface the Z-Tool with CC2530ZNP Mini Kit.

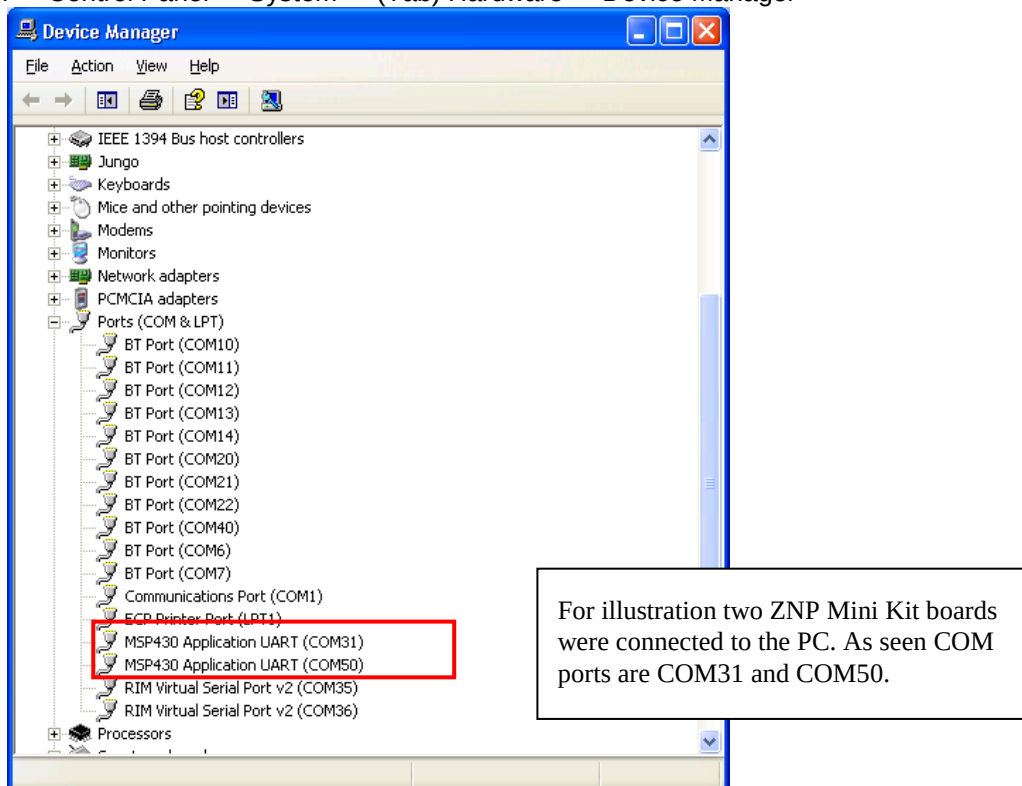
1. Connect the Target board connected to the eZ430 USB stick to a USB port on your PC



Figure 7: eZ430-CC2530ZNP Mini Kit board connected to the PC

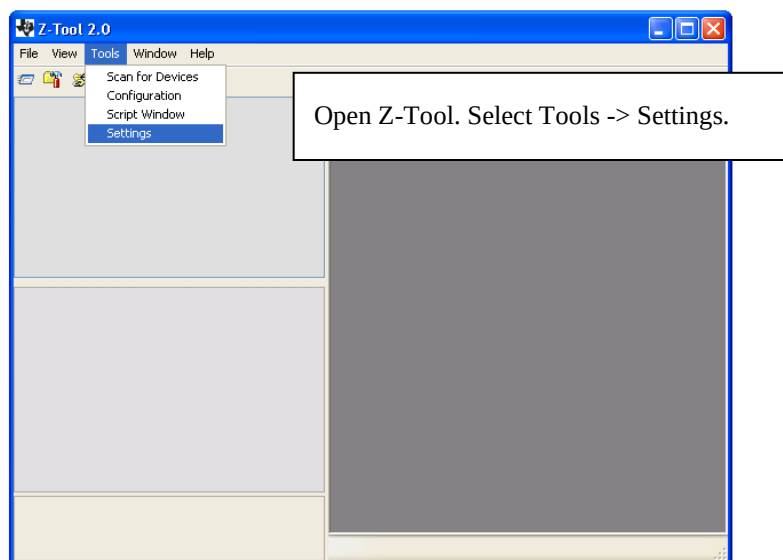


2. Identify the COM port number on the PC of the connected eZ430-CC2530ZNP Mini Kit.
  - Open -> Control Panel -> System -> (Tab) Hardware -> Device Manager

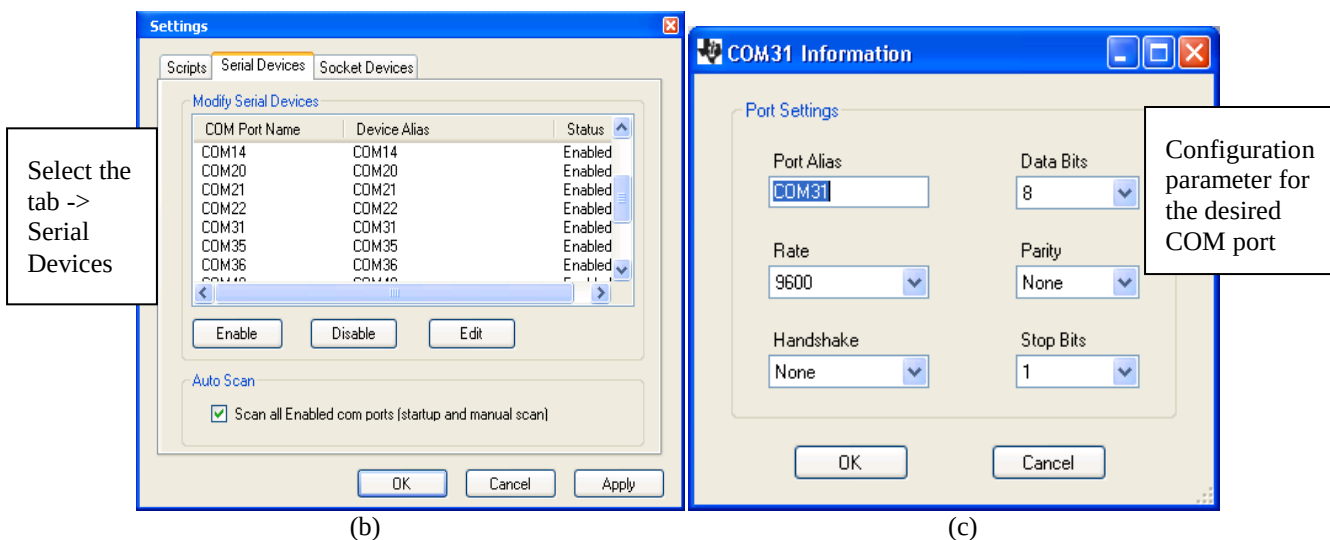


**Figure 8: Device Manager View of the PC to identify COM port number of connected devices**

3. Open Z-Tool. For the identified devices (ex- devices on COM31 and COM50 in this example) configure the communication interface parameters as shown in the figure 9 below

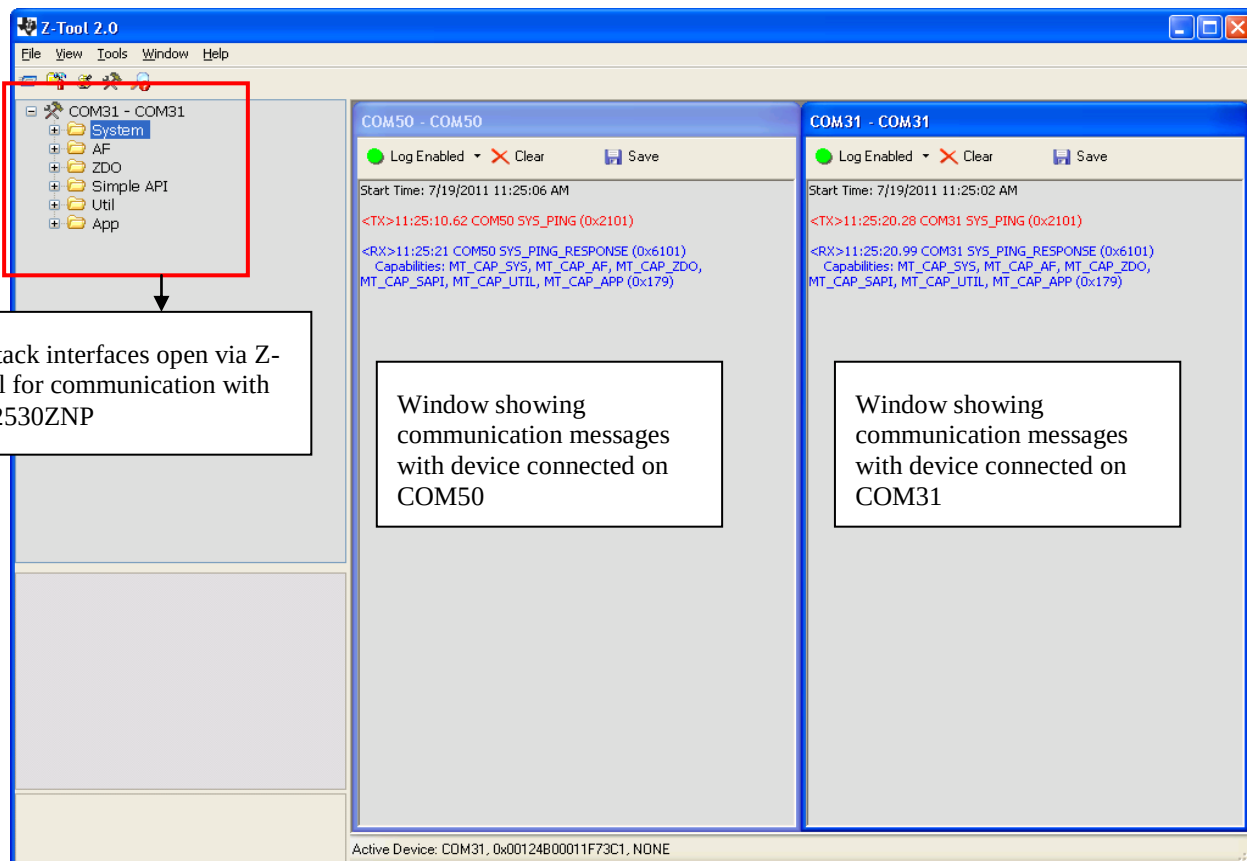


(a)



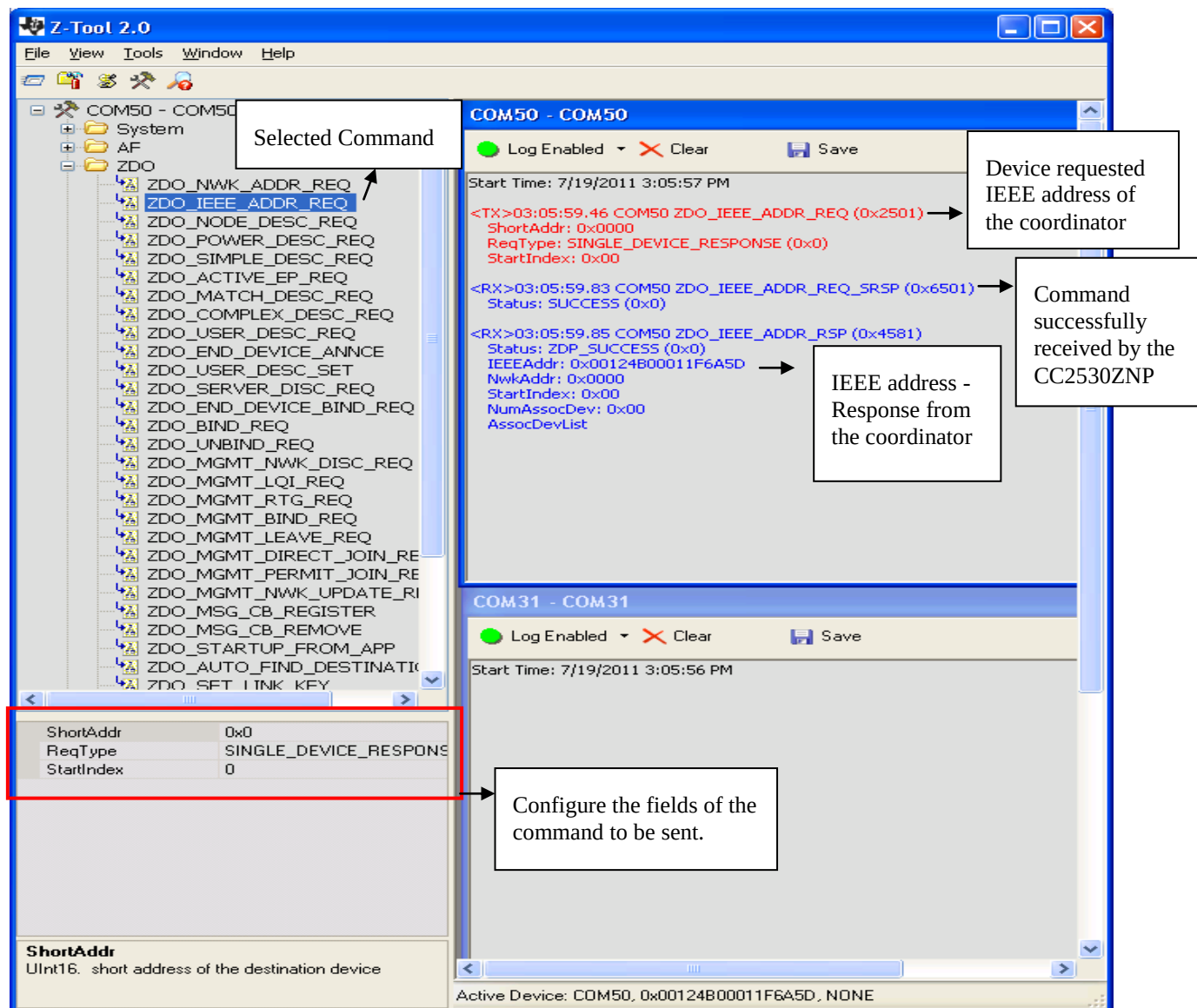
**Figure 9: Steps illustrating the how to configure the configuration parameters for the COM port for communication with the Z-Tool application.**

4. Close Z-Tool and start again
5. Z-Tool should now be able to recognize the CC2530ZNP and the window as shown in Figure 10 should appear.



**Figure 10: Z-Tool Application showing two devices connected and exposed interfaces – System, AF, ZDO, Simple API, Util, and App on the CC2530ZNP on the CC2530ZNP Mini Kit.**

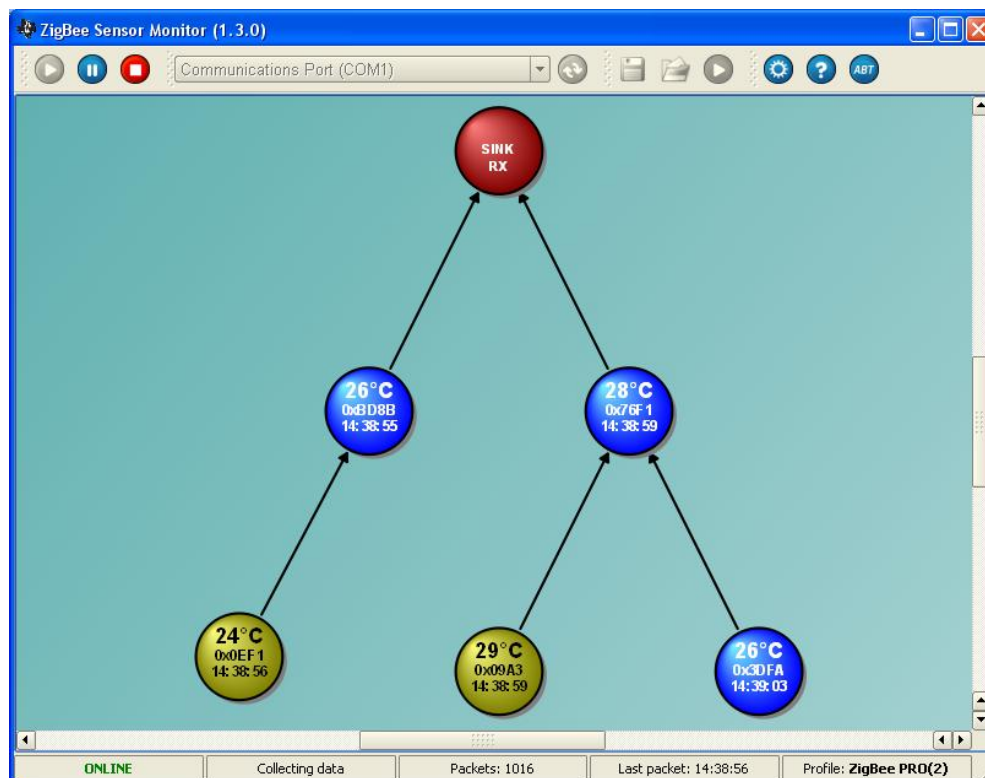
Figure 11 shows an example where Z-Tool was used to query the coordinator for its IEEE address using the ZDO\_IEEE\_ADDR\_REQ command. For details on the frame format for this command please see [3]. Device address 0x0000 for coordinator is sent to the ZNP which returns the SRSP indicating success. This implies the command was successfully received by the ZNP. The ZNP then sends the command OTA and when it receives the response message back from the queried device, it sets the SRDY low to interrupt the MSP430. The MSP430 then receives the ZDO\_IEEE\_ADDR\_RSP message and sends it to the Z-Tool application as seen in Figure 11.



**Figure 11: Z-Tool Application window with two connected devices on COM50 and COM31. Device on COM50 requested IEEE address from the device at address 0x0000 (coordinator) and gets the response back of type IEEE\_ADDR\_REQ\_SRSP.**

## 4. SENSOR MONITOR NETWORK GUI

Figure 12 shows the sensor monitor network visualizer GUI. The installer and documentation for this tool can be found on the TI website [4].



**Figure 12: Sensor Monitor Network GUI representing Coordinator-Sink device as red circle, Router-Source as blue circle and End-Device-Source as yellow circle with temperature, network address and time stamp of the last packet received.**

Enabling the HOST\_MT compile flag on the sink allows the device to communicate with the network visualizer GUI. It reports the periodic message received over the air to be displayed on the GUI. To start the communication sensor monitor network the GUI first sends the Sys\_ping command described below. And, after a successful response, the coordinator-Sink periodically sends the OTA message received to the GUI with the format described in this section.

### 4.1 Sys\_Ping

The sensor monitor network visualizer application sends a sys\_ping to the coordinator sink. The response message must be as shown below. The profile Id of the response message should be 0x42.

| Field | SOF | LENGTH | Cmd 0 | Cmd 1 | Profile Id | FCS |
|-------|-----|--------|-------|-------|------------|-----|
| Bytes | 1   | 1      | 1     | 1     | 2          | 1   |

**Figure 13: Frame Format for Sys\_Ping\_Response message to the GUI**

## 4.2 Sensor Monitor Network Messages

The frame format to send the sensor network message to the GUI is shown below

| Field | SOF | LENGTH | Cmd 0 | Cmd 1 | Data | FCS |
|-------|-----|--------|-------|-------|------|-----|
| Bytes | 1   | 1      | 1     | 1     | 4    | 1   |

**Figure 14: Frame Format for the OTA sensor monitor network messages**

The Data field consists of four bytes in the following order:

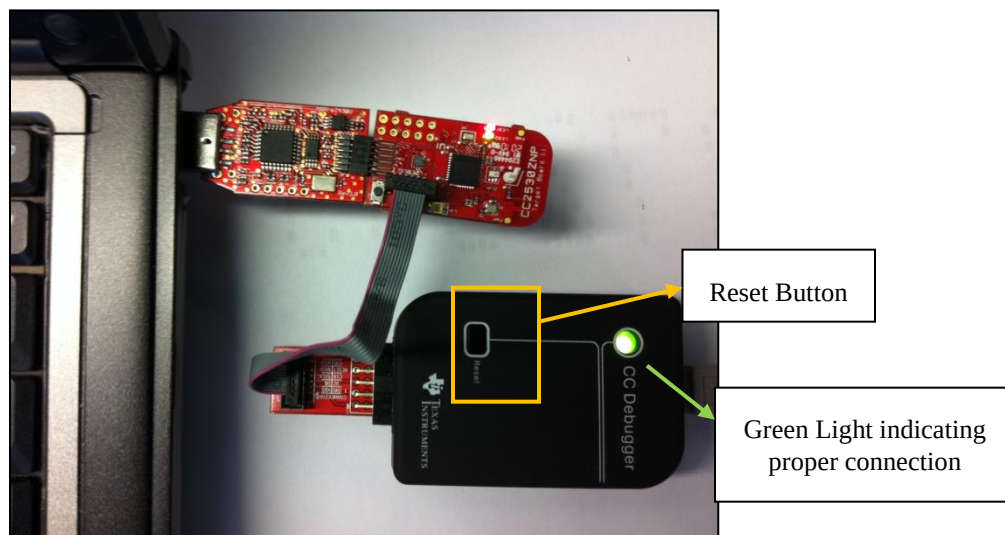
1. Temperature – (one byte)
2. Voltage – (one byte)
3. Parent's Short Address – (two bytes)

## 5. UPGRADING CC2530 ZNP IMAGE

The sensor monitor network sample application has been written using the CC2530ZNP image build from the Z-Stack-2.5.0 release, it is recommended to use the image CC2530ZNP-MK-Pro.hex for running this sample application. The image is included with the install of this sample application at <install-directory>\CC2530ZNP-MK Sensor Network Monitor Application\Bin. This section illustrates how to use the SmartRF® Flash programmer [9] and CC-Debugger to program the CC2530 on the ZNP Mini Kit with a new ZNP Image. The CC-debugger [10] has to be purchased separately from the ZNP Mini Kit development board.

Following are the steps to program a new image into CC2530 on the ZNP Mini Kit.

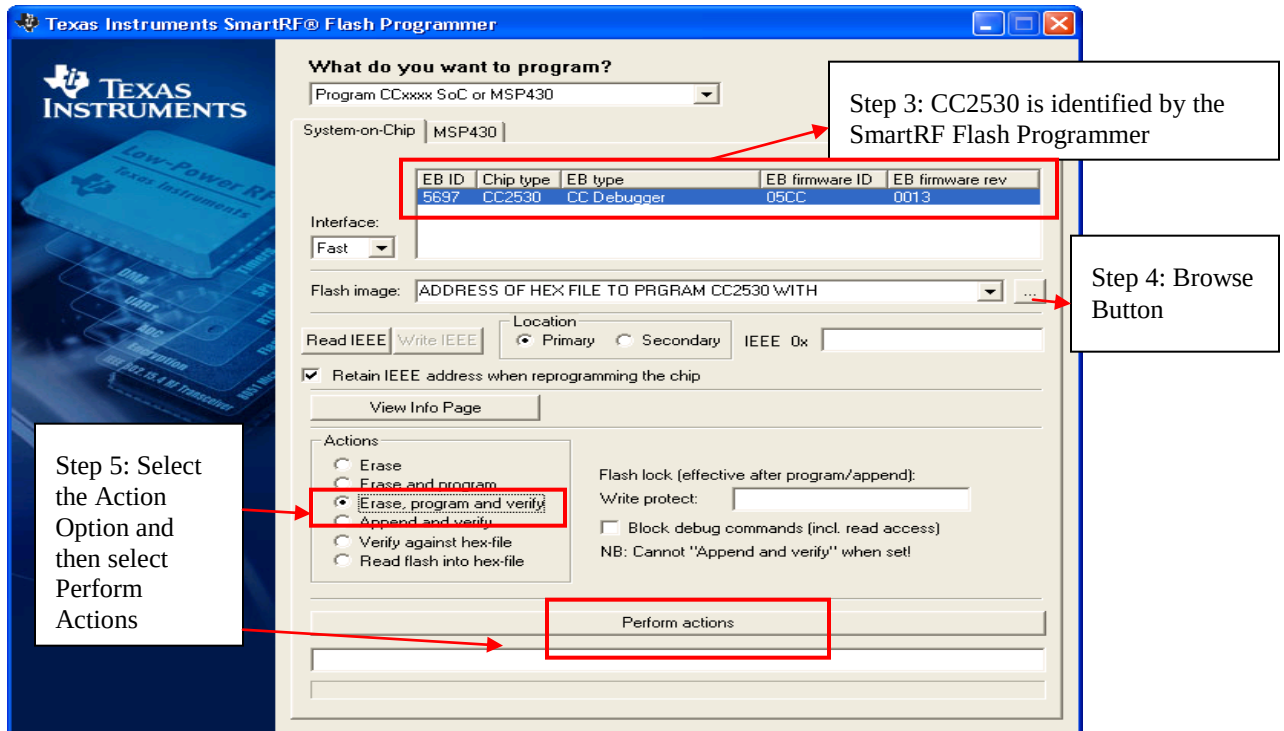
1. Connect the CC-debugger 10 pin header to the 10-pin debug header as shown in Figure 15



**Figure 15: CC-Debugger Connected to CC2530ZNP Mini Kit board. Green Light indicates proper connection**

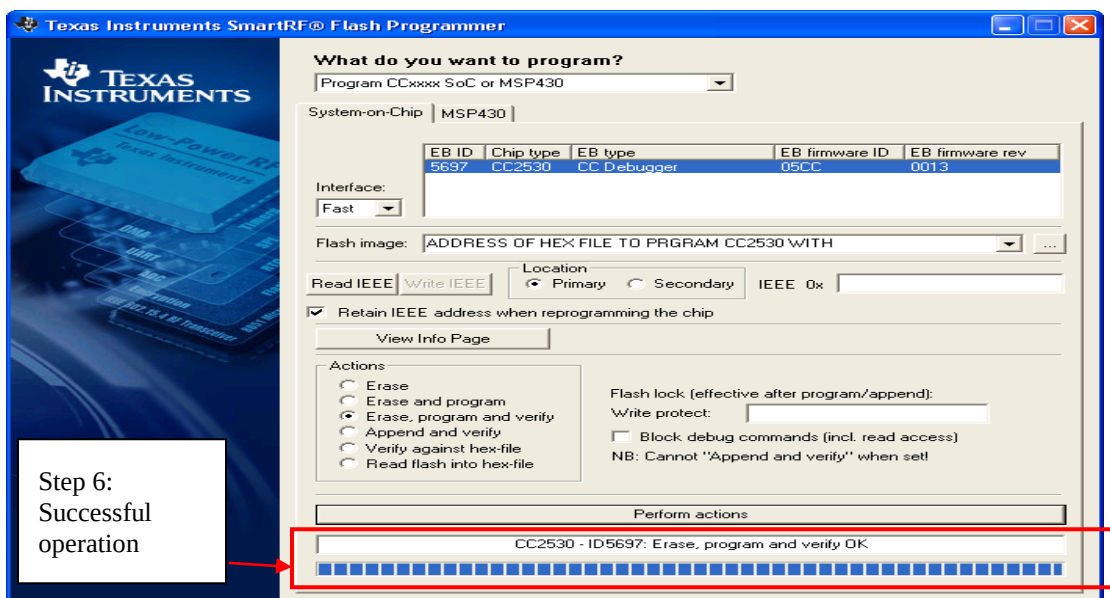
2. Press the reset button on the CC-Debugger. The light on the debugger must turn green indicating device has been correctly connected to the debugger. If the Red light remains ON, it indicates that the device is not connected properly. Ensure that the CC-Debugger's 10-pin connector is not reversed and pin 1 on the connector is connected to the pin-1 on the header and so on.

3. Start the SmartRF Flash Programmer. The device will be recognized in the SmartRF flash programmer window as shown in the Figure 16 below.



**Figure 16: SmartRF Flash Programmer window**

4. Select the desired ZNP image by pointing the SmartRF Flash Programmer to a destination on your PC, by selecting the browse option: see Figure 16 above.
5. Download the image by selecting the option – Erase, program and Verify as shown in the Figure 16 above.
6. Erase, program and verify success message will then appear as shown in Figure 17 below.



**Figure 17: SmartRF Flash Programmer window with status message showing successful operation status message**

## 6. FREQUENTLY ASKED QUESTIONS

### 1. I am getting inconsistent temperature readings reported by nodes. Is there something I need to do to make temperature readings more accurate?

The temperature sensor on the MSP430 needs to be calibrated in order to provide a correct temperature. The sensor is not calibrated in the factory, and it is left up to the user to perform the calibration and store the calibration offset somewhere in the information area of the MSP430. The Sensor Monitor Network application does read from the information memory (at address 0x10F4), but will only read 0xFF if a value has not been set by the user.

### 2. Why does the Coordinator take a few seconds to setup a new network?

To keep things simple, the sensor monitor network application was designed such that a single button press is used to first attempt to join an existing network and, if one cannot be found, to then initiate the formation of a new network as the network Coordinator. Because a device first attempts to join an existing network, there is a noticeable lag in time during which the device is performing an active scan to discover any network in range that meet the joining criteria. After determining that no network exists, to start a network as a Coordinator the device must then also perform an active and passive scan. This entire process takes several seconds and creates a noticeable delay to the user.

### 3. Why does it take a few seconds for the node to join or bind to the Coordinator?

To keep things simple, the sensor monitor network application was designed to incorporate joining and binding into the network startup procedure. When triggered by one or two button presses (depending on the device type), a node first must scan and perform a join to an existing network (assuming one can be found), and then must go through a several step binding process (Figure 1) which establishes a connection between the Source and Sink. This procedure can take a few seconds to complete and can cause a noticeable delay to the user.

### 4. Why do some router nodes not appear as Blue Circles after rejoining the network via the NV\_Restore feature?

The GUI recognizes a device as a router when, after the network join, it sends the first message to the GUI with a dummy temperature reading of 0xFF and a voltage reading of 0xFF. This dummy message is sent after receiving the bind confirm from ZNP. While network rejoining, if the parent of the router joins the network after the router, then the GUI rearranges the node and will not receive the dummy temperature and voltage readings again. This will cause the router node to be recognized as an end device and displayed as a yellow circle. To solve this, bring in the network nodes one by one starting from a network depth of zero.

Please visit the CC2530 product folder [7] on the TI website for additional reference designs and antenna options.



## 7. REFERENCES

- [1]. CC2530ZNP Mini Kit [Wiki Resource Guide](#)
- [2]. CC2530 ZNP Mini Kit Example Software ([swrc211](#))
- [3]. CC2530ZNP Interface Specification (swra312)
- [4]. Sensor Monitor Network Visualizer GUI ([swrc096](#))
- [5]. [Order](#) CC2530ZNP Mini Kit
- [6]. CC2530 Second generation System-on-Chip Solution for 2.4GHz IEEE 802.15.4/RF4CE/Zigbee ([swrs081](#))
- [7]. Sensor Monitor GUI User Guide (swru157)
- [8]. Zigbee Specification document [\[8\]](#)
- [9]. SmartRF Flash Programmer [\[9\]](#).
- [10]. Debugger and Programmer for RF System-on-Chips [\[10\]](#)
- [11]. Getting Started with ZigBee using the CC2530ZNP Mini Kit [\[11\]](#)