Texas Instruments

User's Guide

eZ430-RF2480 Demonstration Kit

(formerly known as eZ430-RFZACC06)

SWRU151A
2008 *Low-Power RF*

If You Need Assistance

Support for the CC2480 and MSP430 device is provided by the Texas Instruments Product Information Center (PIC) Contact information for the PIC can be found on the TI web site at www.ti.com. Additional device-specific information can be found on the MSP430 and LPW web sites at www.ti.com/msp430 and www.ti.com/lpw

We Would Like to Hear from You

If you have any comments, feedbacks, or suggestions, please let us know by contacting us at support@ti.com.

Contents

1 Acronyms and Abbreviations

API	Application Programming Interface
FFD	Fully Functional Device – A ZigBee Device where the transceiver is always on
HAL	Hardware Abstraction Layer
LED	Light Emitting Diode
MT	Monitor and Test
OTA	Over-The-Air – An RF transmission of data by the CC2480
RF	Radio Frequency
RFD	Reduced Function Device – A ZigBee Device that allows the transceiver to be turned off to save power
RPC	Remote Procedure Call – an interface designed so that a program running on one processor can invoke functionality of software running on another processor "as if" it were also running right on that other processor
SAPI	Simple API - a very simplified interface to a subset of the ZigBee stack functionality
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
ZASA	ZigBee Accelerator Sample Application

2 Z-Accel Overview. Wireless Made Easy.

Z-Accel is a new concept by TI to make the design of a new wireless product or the addition of wireless into existing products easy. Z-Accel devices function as ZigBee network processors, encapsulating all of the complexities of wireless into a black box. A host processor running the application simply communicates to a Z-Accel device via a simple serial interface and well defined API. Z-Accel cleanly separates application and networking components for simple system design and integration.

The CC2480, our first product in the Z-Accel family, provides ZigBee made simple. CC2480 provides a fully compliant ZigBee solution based on the ratified ZigBee-2006 specification, and includes a unique preprogrammed IEEE address for each device. This includes full mesh capabilities, Coordinator / Router / End Device functionality, security, power savings, reliable communication, interoperability, and much more.

The eZ430-RF2480 Demo Kit is a complete USB-based MSP430 wireless demonstration tool providing all the hardware and software to evaluate the CC2480 2.4GHz ZigBee network processor and the MSP430F2274 microcontroller. Our demonstration kit has everything needed to understand and fully evaluate the capabilities of CC2480 in a short amount of time with minimal effort.

The eZ430-RF2480 Demo Kit utilizes a free Kickstart version of the IAR Embedded Workbench Integrated Development Environment (IDE) to write, download, and debug your application. The debugger is unobtrusive allowing the user to run an application at full speed with both hardware breakpoints and single stepping available while consuming no extra hardware resources.

eZ430-RF2480 Demonstration Kit features:

- ZigBee Accelerator Sample Application (ZASA) for temperature and battery voltage sensor reporting using the eZ430-RF2480 Sensor Monitor PC application tool
- Sample Host Processor Code to demonstrate configuration, network instantiation or selection, binding, and communication
- 5 available GPIO development pins
- Highly integrated, ultra-low-power MSP430 MCU
- Two general-purpose digital I/O pins connected to green and red LEDs for visual feedback
- Interruptible push button for application control

Please visit www.ti.com/CC2480 for more information!

Note! The eZ430-RF2480 Target Board has been optimized for board size, not RF range, and the achieved RF performance **DOES NOT** reflect the true capability of the CC2480.

3 eZ430-RF2480 Demo Kit Contents and Requirements

3.1 Hardware Contents

- The demo kit includes:
  - 3 CC2480 Target Boards
  - 1 eZ430 USB Emulator Board
  - 2 AAA eZ430 Battery Boards (batteries included)

3.2 Software, Documentation, and Sample Applications

Please visit www.ti.com/eZ430-RF2480 for the latest sample application code, tools, and documentation including:

- ZASA installer with Project files and Quick Start Guide
- eZ430-RF2480 Sensor Monitor Installer
- ZigBee Documentation

Please visit www.ti.com/CC2480 for the latest information on the Z-Accel ZigBee Network Processor

3.3 System Requirements

- Microsoft Windows XP or Vista

4 eZ430-RF2480 Demo Kit Quick Start Instructions

To get up and running with your eZ430-RF2480 Demo Kit please perform the following:

- Install the eZ430-RF2480 Sensor Monitor application from the eZ430-RF2480 website
    - www.ti.com/eZ430-RF2480
- Connect one of the CC2480 Target boards to the eZ430 USB Emulator board creating a eZ430-RF2480 Dongle which will act as the ZigBee network Coordinator and a gateway for communication with your PC. Plug the Dongle into the USB port.
    - When the Dongle is connected to the PC initially you will see both RED and GREEN LEDs blink showing it is awaiting configuration.

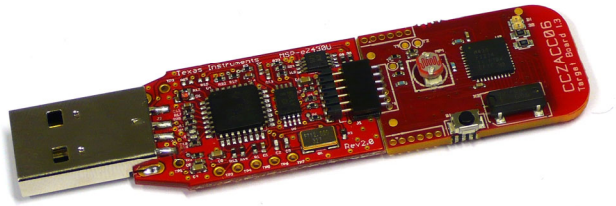

Figure 1: eZ430-RF2480 Dongle

    - If this is your first time connecting the Dongle you will be notified that your PC has detected new hardware. The driver is a standard windows driver (usbser.sys) and will be detected automatically. When the first dialog box appears select “No, not at this time”. Then choose to install this driver automatically. When you are notified that the software has not passed Windows Logo testing, select “Continue Anyway”. Please see the eZ430-RF2480 Sensor Monitor User Guide for further information if you have problems installing the hw driver.
- Connect the remaining two target boards to the provided eZ430 Battery boards with batteries installed to create Battery Nodes.
    - When the target boards and battery boards are connected you will see both RED and GREEN LEDs blink showing the Nodes are awaiting configuration

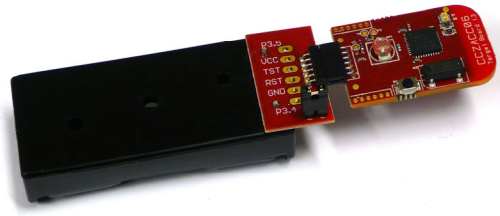

Figure 2: eZ430-RF2480 Battery Node

Note: The jumper on the battery board must be in place to connect power, and can be used as an on/off switch as desired.

- Run the eZ430-RF2480 Sensor Monitor application
    - The application can be run from the start menu at
        - All programs - Texas Instruments - eZ430-RF2480 Sensor Monitor
- Establish the Dongle as network Coordinator-Sink by pressing the button on the target board once. The device could take as long as 10 seconds to setup a network so please be patient.
    - When setup as Coordinator-Sink the Dongle will display the RED LED as always on.
    - Note: If both LEDs remain OFF the Coordinator has failed to create a network. Please disconnect and reapply power and attempt configuration again until a valid network and Coordinator is established.
    - Note: The Dongle is now both the network Coordinator and the application data Sink for other devices in the network.
- Setup a Battery Node as a Router-Source, meaning it will act as both a ZigBee Router and an application Source of sensor data, by pressing the button once on the target board of the Battery Node. The Router can take up to 10 seconds to scan, join, and bind into the network, so please be patient during this step.
    - When setup as Router-Source the GREEN LED on the Battery Node will remain ON.
    - Note: If after trying to configure the Router-Source, both LEDs remain in the OFF position, the attempt to discover or join the network has failed. Verify that the network Coordinator has its RED LED ON. If the RED LED is blinking on the Coordinator then permit joining has been disabled. Press the button a single time on the Coordinator to toggle the permit joining flag to allow joining (the RED LED should go into an always ON state). If the Router-Source will not join when reset (remove and reapply power) then try bringing the device closer to the Coordinator to make sure it is in RF range).
- Setup a Battery node as an End Device-Source, meaning it will act as both a ZigBee End Device (low power) and an application Source of sensor data, by pressing the button twice on the target board of the Battery Node. The End Device can take up to 10 seconds to scan, join, and bind into the network, so please be patient during this step.
    - When setup as End Device-Source the GREEN LED on the Battery Node will blink at 1 Hz.
    - Note: If after trying to configure the End Device-Source, both LEDs remain in the OFF position, the attempt to discover or join the network has failed. As before, verify that the Coordinator or Router has the RED or GREEN LED in the

always on position allowing joining. If either or both does not, press the button a single time to toggle joining state and remove and reapply power to the End Device to attempt joining again.

- Once all devices are connected on the network the Source devices will automatically perform binding to the Coordinator-Sink
- Once binding is complete the Source devices will begin to periodically (every 10 seconds) report temperature and battery voltage data to the Sink
    - Note: The RED LED will be used by the Router-Source and End Device-Source to depict the successful transmission and reception of an acknowledgement for a sensor report. If, upon startup, you do not see the RED LED blink periodically please first check to ensure that the device is in range of the Coordinator or a working Router, and if problems persist, disconnect power from the device and perform configuration again to try and resolve the problem.
- Watch the eZ430-RF2480 Sensor Monitor to see both Sources connect, observe the network topology, and periodically report their temperature and battery voltage readings
- Note: Once joined to the network, the button on the Coordinator and Router devices can be used to turn Permit Join on and off. This controls whether additional devices are allowed to join the network through that specific Router and Coordinator. If you would like to see an End Device joined a hop from the Coordinator, simply connect the Coordinator, then Router, and turn permit joining OFF on the Coordinator (press the button once) prior to setting up the End Device.

5 eZ430-RF2480 Demo Kit Hardware

The eZ430-RF2480 Demo Kit comes with the hardware necessary to demonstrate ZigBee functionality using the CC2480 device. This includes 3 CC2480 Target Boards, 1 eZ430 USB Emulator Board, and 2 eZ430 Battery Boards. All schematics for these boards are available on the www.ti.com/eZ430-RF2480 web page.

5.1 CC2480 Target Board Design

The CC2480 target board demonstrates Z-Accel by connecting a MSP430F2274 to our CC2480 Network Processor over SPI. The CC2480 target board is a reference design for CC2480 and demonstrates the use of Z-Accel designed to minimize board size utilizing a chip antenna. The target board includes two LEDs, one Red and one Green, a push button, and 5 GPIO lines exposed for expanding the I/O interface. The MSP430F2274 GPIO pins available on the target board include P2.2, P2.3, P2.4, P4.4, and P4.5. None of the GPIO pins for the CC2480 are available on the target board.

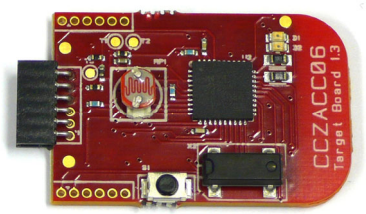

Figure 3: CC2480 Target Board

5.2 eZ430 USB Emulator Board Design

The eZ430 USB Emulator Board provides a USB to Serial connection and power supply for the CC2480 Target Board.

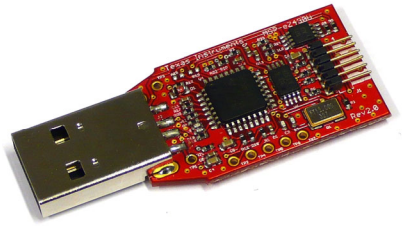

Figure 4: eZ430 USB Emulator Board

5.3 eZ430 Battery Board Design

The eZ430 Battery Board interfaces to the CC2480 target board to provide power for use in mobile situations.

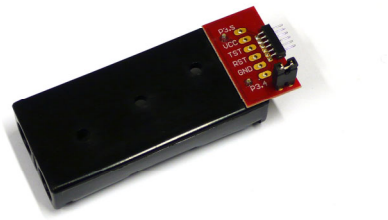

Figure 5: eZ430 Battery Board

6 ZASA – ZigBee Accelerator Sample Application

The eZ430-RF2480 Demo Kit comes with the ZigBee Accelerator Sample Application (ZASA), built to demonstrate the functionality and ease of use of the CC2480 device. ZASA runs on the MSP430F2274 and interfaces to the CC2480 device via SPI. The CC2480 demo hardware and MSP430F2274 ZASA software, coupled with the eZ430-RF2480 Sensor Monitor application provides everything needed to demonstrate Z-Accel.

The ZASA project is provided as a reference project for customers interested in evaluation or reference software on the host side of Z-Accel.

The ZASA project is built using the IAR Embedded Workbench Kickstart for MSP430 V4, available for download on the IAR website (www.iar.com).

When running the installer, the ZASA Project is installed in the C:\Texas Instruments\eZ430-RF2480\Projects\zaccel\ZASA\IAR\ folder and is titled SampleApp.eww

6.1 ZASA Application Description

The ZASA application is comprised of two types of devices, Sources and Sinks, operating in potentially all three ZigBee roles; Coordinator, Router, and End Device. Sources provide temperature and bus voltage readings collected by the MSP430 using its on chip temperature sensor and A2D converter. The ZASA application provides code for operating as either type of device, and allows for easy configuration via the button on the CC2480 target board.

Figure 6: ZASA State Machine

Note: The ZASA description below includes concepts of network startup, association or joining, binding, and data reporting that are specific to the way ZigBee networks operate. If you would like to learn more about the ZigBee protocol please see the suggested reading section at the end of this document for more information.

6.1.1 Startup and Configuration

- Startup
    - Upon normal startup a device will blink both the RED and GREEN LEDs awaiting configuration
- Configuring the Device Type
    - A device can be configured in 2 ways using 1 button press or 2 button presses based on the desired functionality
    - Pressing the button 1 time in 2 seconds will first attempt to join the device as a Router-Source in an existing network, and if no network exists, will then create a new network 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 2 times in 2 seconds will configure the device as an End Device-Source
    - If, after using the button to attempt to configure a device, the device's LEDs are both OFF and do not display a clear state (see below for LED configurations as per device state), simply reset the device and try again as there may have been a problem during scanning or joining. Please make sure a Coordinator has successfully created a network before proceeding to configure additional devices.

6.1.2 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 permit joining on and off, with the RED LED in an always ON mode 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 permit joining on and off, with the GREEN LED in an always ON mode 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

6.1.3 ZASA Application Role

- 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-RF2480 Sensor Monitor, the Sink will also send the received data report up to the PC for display on the GUI.
- Source
    - Once joined, the Source will initiate binding to discover 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 – although this rate is configurable (see configuration section below).
    - 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.

6.1.4 Summary of LED states

Device Type	**RED LED**	**GREEN LED**
Not Configured	Blinking	Blinking
Coordinator – Sink	ON –Joining Permitted Blinking – Joining Not Permitted	Blink – Source Report Received
Router – Source	Blink – Data Send Acked	ON –Joining Permitted Blinking – Joining Not Permitted
End Device – Source	Blink – Data Send Acked	Blinking – Joined to network

Figure 7: Table of LED States Based on Device Configuration

6.2 ZASA Detailed Code Description

This ZASA is intended to demonstrate as cleanly and quickly as possible how to drive the CC2480 device, implementing the RPC protocol across the SPI bus, and to provide a simple demonstration of ZigBee through the sending and routing of sensor data.

Although Z-Accel supports an extensive range of functionality through a complete set of API calls documented in the CC2480 Interface Specification, this Sample Application only utilizes the subset of ZigBee functionality provided by the SAPI, and thus only implements the RPC for SAPI (and one interface on RPC for SYS.).

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

APP - Application: This is the customer-specific code that implements the application functionality previously described.

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

MT - Monitor-Test: this code is a software abstraction to communicate with a PC Tool.

SAPI - Simple API: this body of code provides the identical or very similar API to the APP code that it would have if it were being written to be compiled directly with the ZigBee Stack code.

ZACCEL - This body of code implements the RPC and SPI protocol required by the ZigBee Accelerator (CC2480) acting as a slave on the SPI bus.

In the Sample Application, a device that is configured to be a sensor is referred to as a "Source" and a device that is configured to be the single, common collector of sensor data is referred to as the "Sink". Thus Sink and Source are roles defined by the APP code in solving the case problem and FFD and RFD are roles defined by the ZigBee Accelerator in order to fulfill the ZigBee specified behaviors for routing and low-power respectively.

This Sample Application itself comprises the functionality referred to as "host" - this application must drive the SPI bus as the master and implement the client responsibilities described in the "CC2480 Interface Specification" document while the CC2480 fulfills the server responsibilities.

The method of user input to configure the device is the button press. The manner for communicating state back to the user is by the LEDs.

The code transitions through a series of states depending on user input and CC2480 feedback. The state is tracked by this 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.
```

The first code to run after reset is compiler-generated and includes such house-keeping as initializing global variables. The first Sample Application code to run is found in sample_main.c in this C-code function:

```
void main(void)
```

After essential initialization, the program loops infinitely, servicing the HAL events that are set obviously by hardware events, and the ZigBee Accelerator events that are set by feedback from the CC2480. The primary responsibility for the host is to act on the ready indication from the CC2480 known as slave ready (SRDY) - this signal is polled in the function *appExecHost()*. The aforementioned function is where the application fulfills its responsibilities as the host.

After power up, this Sample Application (the host), is in the state *appIniting* which is to say, it is waiting to initialize the CC2480. CC2480 will always restore its previous network state from NV memory, unless it was pre-configured not to do so by the host before the last reset. This Sample Application does not take advantage of the NV restore behavior so as to have a perfectly predictable out-of-the-box behavior. Whenever the CC2480 resets, it always sends a SYS API message to indicate that it has reset. Thus, the host is waiting for this indication when it is in the *appIniting* state. Upon receiving the reset indication, the host pre-configures CC2480 to not perform NV restore on the next reset and transitions to the *appWaiting* state.

When the host is in this *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. While in the *appJoining* or *appJoinWaiting* state, the LEDs will be off.

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.

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 SAPI to negotiate OTA and 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 10-seconds, the 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 OTA, so it retries sending the same data OTA up to 3 times every time that the SAPI feedback from the CC2480 indicates that the message has not been acknowledged. 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.

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 negotiate OTA and 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 10-seconds, the 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 OTA, so it retries sending the same data OTA up to 3 times every time that the SAPI feedback from the CC2480 indicates that the message has not been acknowledged. While awaiting the acknowledge, the device enters low power and awakes every 2 seconds by a timer in order to poll for the SAPI feedback from the CC2480. 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.

6.3 ZASA Configuration and Compile Options

The ZASA includes configuration options that can be changed based on performance requirements. Many configuration options are provided but not all are meant to be changed for the eZ430-RF2480 Demo Kit. The configuration options of interest for ZASA are as follows:

- HOST_PROFILE_ID

The Profile ID defines the ZigBee Profile being used by the application. In the case of ZASA we use a TI specific Profile ID.

Value must be set to 16 bt Profile ID obtained from the ZigBee Alliance

- HOST_NV_CHANLIST

The Channel List provides a bitmap of all valid channels that a device will Scan during startup. By default the ZASA channel list is configured to only utilize a single channel. For guidelines on modifying or increasing the list of available channels please read the comment provided within the configuration file for more information on how this affects ZASA performance and consult the ZigBee Developer's Guide referenced in the Suggested Reading section of this document for further technical detail.

- HOST_NV_PANID

The PAN ID, if set to a number other than 0xFFFF, specifies the PAN ID to be used when creating a network, or the PAN ID of the network to which a device is allowed to join. If 0xFFFF is set as the PAN ID then a Coordinator will randomly generate a PAN ID to use when starting a network, and a Router or End Device will join any network that is discovered.

7 eZ430-RF2480 Demo Kit Tools

The eZ430-RF2480 Demo Kit comes with the eZ430-RF2480 Sensor Monitor. The installer and documentation for this tool can be found on the TI website.

www.ti.com/eZ430-RF2480

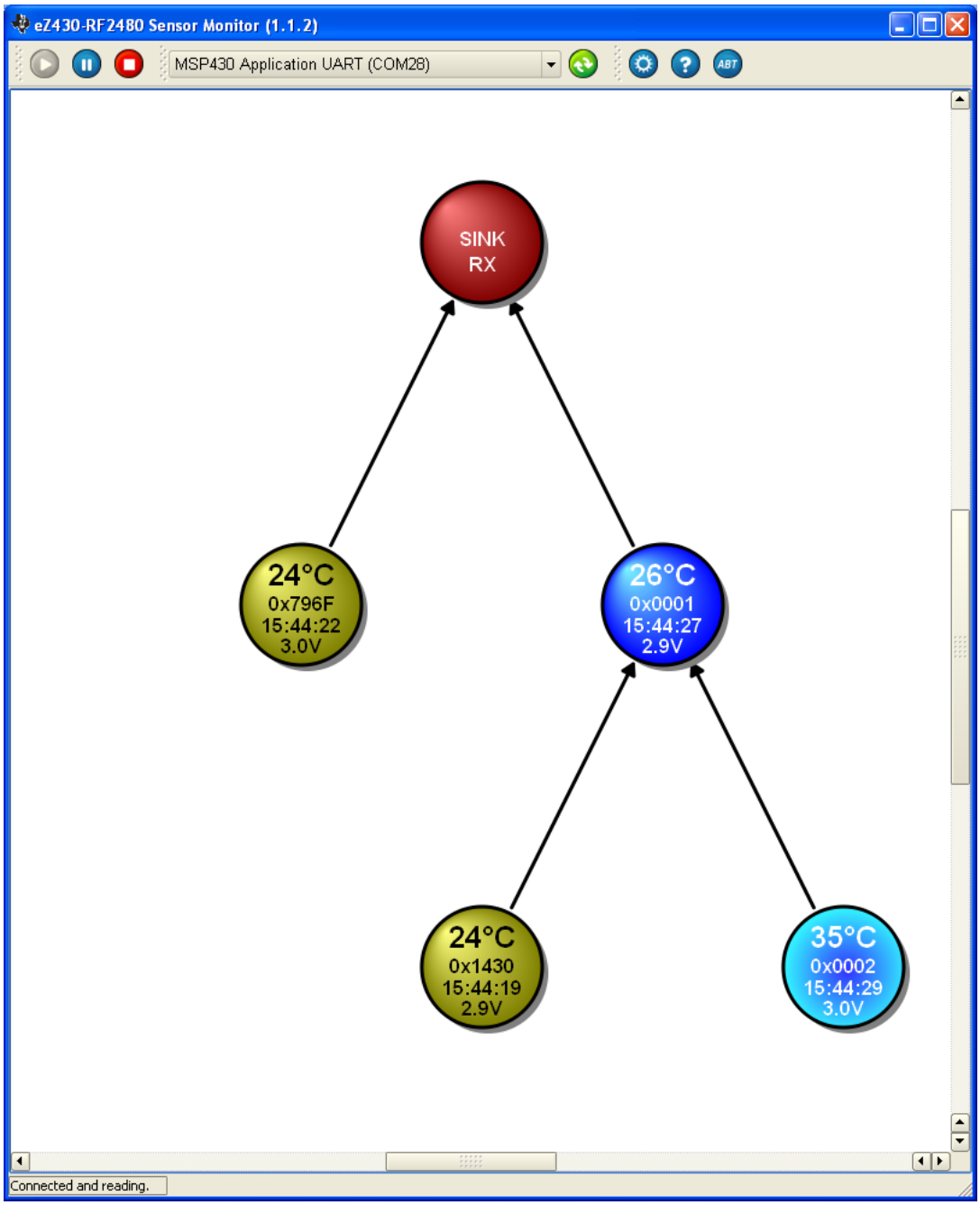

Figure 8: Screen shot of eZ430-RF2480 Sensor Monitor

8 Suggested Reading

For more information on CC2480 please visit www.ti.com/CC2480 and download the datasheet and API Guide today.

If you would like to go beyond simply demonstrating ZigBee functionality using the eZ430-RF2480 Demo Kit, there are many additional documents that will be of interest. Below please find a list of documentation with a simple description to help point you in the right direction.

- **CC2480 Datasheet**

The CC2480 Datasheet provides information on the CC2480 API, SPI and UART interface, RPC calls, performance characteristics, etc…

- **CC2480 Interface Specification**

The CC2480 Interface Specification provides detailed information about the CC2480 device, as well as the complete API available to the Host processor. This includes the SPI interface and Remote Procedure Calls (RPC) for communication between the host processor and the CC2480.

- **Z-Stack Developer's Guide**

The Z-Stack Developer's Guide contains a general description of ZigBee, and how it works as a networking protocol. This includes information on devices, forming and joining a network, binding, sending and receiving data, and much more.

In addition there is an extensive list of application and design notes, reference design, and additional documentation provided on the CC2480 and eZ430-RF2480 product pages. Please visit www.ti.com/CC2480 and www.ti.com/eZ430-RF2480 for more information.

9 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 this calibration and store the calibration offset somewhere in the information area of the MSP430. ZASA does read from the information memory (at address 0x10F4) but will only read FF 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, ZASA was designed so that a single button press was 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, ZASA 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 which establishes a connection between the Source and Sink. This procedure takes a few seconds to complete and can cause a noticeable delay to the user.

4. **What kind of range should I expect to get with the eZ430-RF2480 hardware?**

Based on practical range testing, using ZASA, with one node connected to a PC and the other node connected to the battery board, we have measured indoor line-of-sight range of more than 35 meters. This range can be significantly affected by the orientation of the boards and the environment. Note that the eZ430-RF2480 target board was designed to optimize form factor and does not focus on maximizing RF range. Please visit the CC2480 product folder on the TI website for additional reference designs and antenna options.

10 Revision History

Revision	Date	Comment
SWRU151A	2008-04-03	Use new part number for CC2480 (previously CCZACC06). Minor modifications, general brush-up.
SWRU151	2008-03-04	Initial revision.