



CY3271

PSoC FirstTouch Starter Kit with CyFi Low-Power RF

Spec. # 001-48286 Rev. **

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone (USA): 800.858.1810
Phone (Intl): 408.943.2600
<http://www.cypress.com>

Copyrights

© Cypress Semiconductor Corporation, 2008. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

FirstTouch™, PSoC Designer™, Programmable System-on-Chip™, and PSoC Express™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Flash Code Protection

Cypress products meet the specifications contained in their particular Cypress PSoC Data Sheets. Cypress believes that its family of PSoC products is one of the most secure families of its kind on the market today, regardless of how they are used. There may be methods, unknown to Cypress, that can breach the code protection features. Any of these methods, to our knowledge, would be dishonest and possibly illegal. Neither Cypress nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Cypress is willing to work with the customer who is concerned about the integrity of their code. Code protection is constantly evolving. We at Cypress are committed to continuously improving the code protection features of our products.

Contents



1. Introduction	5
1.1 Welcome	5
1.2 CY3271 System Overview	6
1.2.1 CY3271 Hardware Overview	6
1.2.2 CY3271 Software Overview	10
1.3 Document Revision History	13
1.4 Documentation Conventions	13
2. Installation Guide	15
2.1 CY3271 Installation Instructions	15
3. Design Examples	21
3.1 Design Example Summary Table	21
3.2 Out of Box Design Examples	22
3.2.1 CY3271 PSoC FirstTouch MultiFunction Expansion Card CapSense Slider	22
3.2.2 Ultra Low Power Wireless Temperature Sensor	22
3.2.3 Wireless MultiFunction Demonstration.....	26
3.2.4 Standalone MultiFunction Demonstrations.....	29
4. Firmware	33
4.1 PC Bridge Wireless Hub	33
4.1.1 Design Features.....	33
4.1.2 Firmware Architecture	33
4.1.3 Firmware Model	37
4.2 RF Expansion Card Ultra Low Power Temperature Sensor	42
4.2.1 Design Features.....	42
4.2.2 Firmware Architecture	42
4.2.3 User Modules used in the Temperature Sensor PSoC Project	46
4.2.4 Firmware Model	52
4.2.5 Architecture	52
4.3 Wireless I2C Bridge for RF Expansion Card	58
4.3.1 Design Features.....	58
4.3.2 Firmware Architecture	58
4.3.3 User Modules used in the I2C Bridge PSoC Project.....	61
4.3.4 Firmware Model	67
4.3.5 Architecture	67
4.4 MultiFunction Expansion Card Light Sensor.....	72
4.4.1 MultiFunction Expansion Card CapSense Slider	72
4.4.2 MultiFunction Expansion Card Proximity Sensor	75
4.4.3 MultiFunction Expansion Card Temperature Sensor.....	78
4.4.4 CY3271 PSoC FirstTouch MultiFunction Expansion Card Light Sensor	83

5. Hardware	85
5.1 PC Bridge	85
5.1.1 Programming the PC Bridge Application Processor	86
5.2 RF Expansion Card Overview	87
5.2.1 RF Expansion Card.....	87
5.2.2 Programming the RF Expansion Card	88
5.2.3 Hardware Design	88
5.2.4 LED Connections	89
5.3 MultiFunction Card (FTMF Expansion Card)	90
5.4 AAA Power Pack	92
5.5 CR2032 Power Pack	93
6. Specifications	95
6.1 General RF	95
6.2 RF Expansion Card	95
6.3 PC Bridge	95
6.4 MultiFunction Expansion Card.....	95
6.5 Certifications.....	96
7. Frequently Asked Questions	97
9. Appendix	103

1. Introduction



1.1 Welcome

Thank you for purchasing the CY3271 PSoC® FirstTouch™ Starter Kit with CyFi™ Low-Power RF.

The CY3271 is designed to quickly evaluate the flexibility, integration, and mixed signal capabilities of Cypress Programmable System-on-Chip (PSoC). This evaluation is aided by a wide variety of sample projects.

You can use the sample projects to explore:

- PSoC's programmable analog and digital blocks to interface to common sensors (such as thermistors) and actuators (such as LEDs). In addition, you can also create common serial interfaces (for example, SPI and I2C).
- PSoC Designer Integrated Development Environment (IDE) to create embedded designs using two methods: traditional chip-level designs that involve writing code, and code free system level designs.
- Cypress 2.4 GHz CyFi Low-Power RF technology to easily add reliable, simple, and power-efficient wireless connectivity to your embedded designs.

You are invited to evaluate the included sample projects, and then experiment with the included hardware and software to create your own designs.

If you have questions about or need help with the CY3271 kit, visit our online support center at <http://www.cypress.com/support> for support options, or contact your local Cypress sales representative or authorized distributor.

1.2 CY3271 System Overview

1.2.1 CY3271 Hardware Overview



1.2.1.1 Hardware Components

The CY3271 kit hardware consists of five boards:

PC Bridge (FTPC)



The PC Bridge (FTPC) can be used to:

- Program all PSoC devices in the CY3271 kit
- Act as a bridge between all boards in the CY3271 system and the PC, using a USB-to-I2C interface
- Feature a CyFi low-power RF transceiver (with RF output power up to +20 dBm). When this is combined with an onboard PSoC, it acts as the Hub in CyFi wireless networks.

RF Expansion Card (FTRF)



The RF Expansion Card (FTRF) features a PSoC device and a CyFi transceiver (with RF output power up to +20 dBm). FTRF serves the following functions:

- Combined with one of the power packs, it can act as a standalone CyFi wireless node with an onboard thermistor for temperature measurements.
- With its female expansion header, it can be used as a CyFi low-power RF module to add wireless connectivity to boards that are connected to it. For example, connecting the MultiFunction Expansion Card (FTMF) into the FTRF Card enables you to wirelessly transmit the values of the sensors on FTMF to the PC.
- Its male interface header features an I2C interface and unused GPIOs. This enables you to use FTRF as a CyFi low-power RF module for prototyping in your own system.

MultiFunction Expansion Card (FTMF)



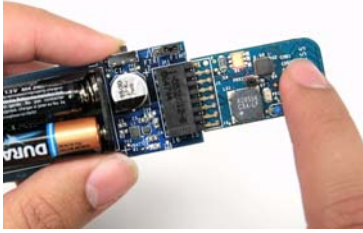

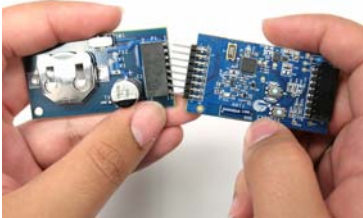

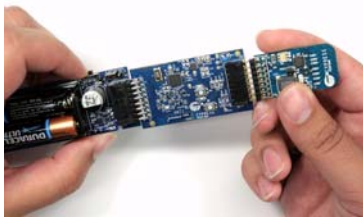


The MultiFunction Expansion Card (FTMF) features a PSoC device, and several sensors and actuators that enable easy experimentation:

- 7-element CapSense slider
- CapSense proximity sensor
- Thermistor
- Ambient light level sensor
- Red or green or blue triple LED cluster
- Speaker

The FTMF interface header also features an I2C interface and four unused GPIOs for prototyping in your own system.

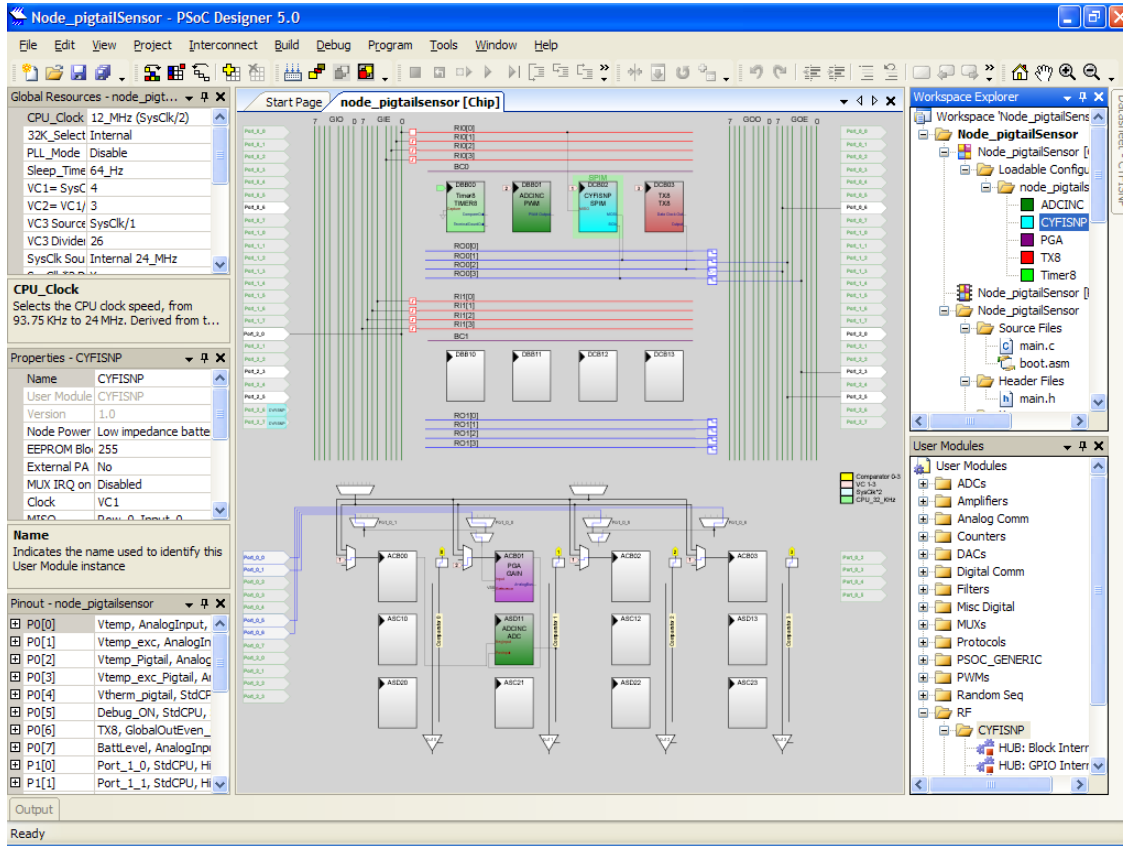
1.2.1.2 Hardware Connectivity

Scenario	PC Side	Wireless Link	Remote Side
Programming the FTMF	 <p>FTPC ↔ FTMF</p>	Not Applicable	Not Applicable
Programming the FTRF	 <p>FTPC ↔ FTRF</p>	Not Applicable	Not Applicable
Standalone FTMF Demos	Not Applicable	None	 <p>AAA ↔ FTMF</p>
Wireless Ultra Low-Power Temperature Sensor Demo	 <p>FTPC</p>	<p>↔</p> <p>Wireless</p> <p>↔</p>	 <p>CR2032 ↔ FTRF</p>
FTMF Demos over Wireless	 <p>FTPC</p>	<p>↔</p> <p>Long Range Wireless</p> <p>↔</p>	 <p>AAA ↔ FTRF ↔ FTMF</p>

Introduction

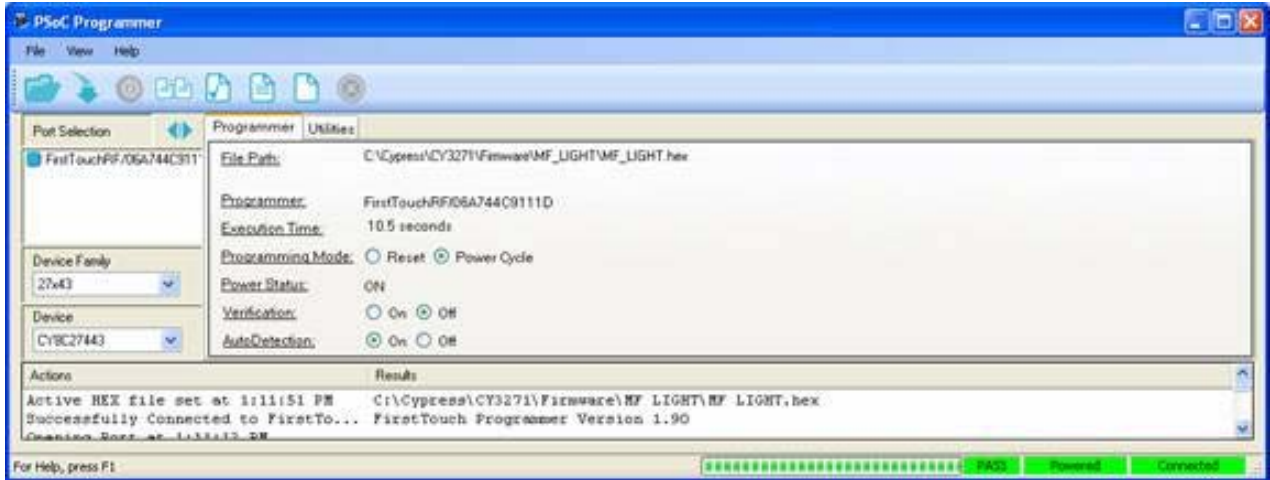
1.2.2 CY3271 Software Overview

1.2.2.1 PSoC Designer



PSoC Designer is the integrated development environment (IDE) where all PSoC projects are created, edited, built, and debugged. You can open every firmware example that is included with the CY3271 kit in PSoC Designer.

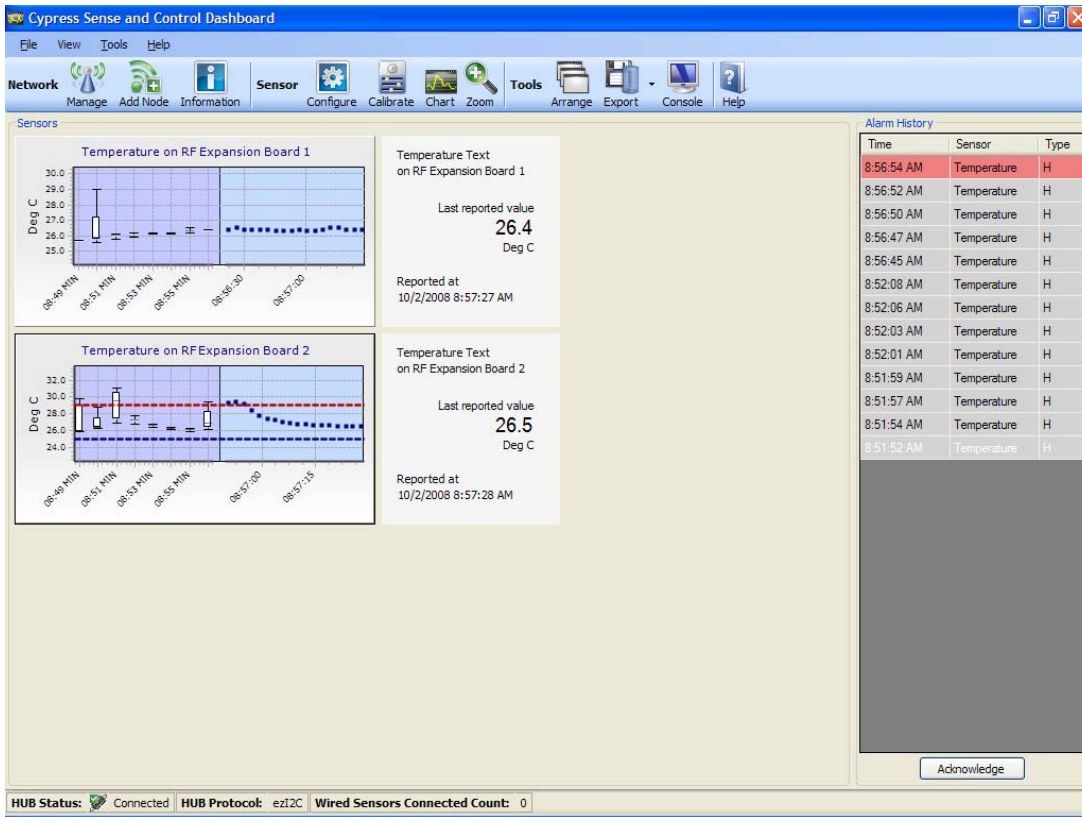
1.2.2.2 PSoC Programmer



PSoC programmer recognizes the PC Bridge as a programmer and is used to program all elements of the kit. To program the RF or Multifunction Expansion card simply connect the card to the PC Bridge interface connector. The PC Bridge master device automatically detects the presence of an external card. To program the slave processor in the PC bridge, you must ensure that there is no card connected to the PC bridge interface connector.

1. Programming Mode: Reset
2. AutoDetection: On
3. Connect the PC Bridge into an available USB port and make sure it is recognized and selected in the port selection window.
4. Connect the expansion card to the PC Bridge interface connector unless programming the slave processor, and then select the appropriate .HEX file from the Firmware directory.
5. Press F5 or Program to start programming.

1.2.2.3 Cypress Sense and Control Dashboard (SCD)



SCD enables data logging and monitoring of wired and wireless sensors created using PSoC. The features include data logging, calibration, alarms, and data aggregation from hundreds of sensors.

In the CY3271, SCD is used to wirelessly log data from sensors connected to the PC, using the FTRF.

1.3 Document Revision History

Table 1-1. Revision History

Revision	PDF Creation Date	Origin of Change	Description of Change

1.4 Documentation Conventions

Table 1-2. Document Conventions for Guides

Convention	Usage
Courier New	Displays file locations, user entered text, and source code: C:\ ...cd\icc\
<i>Italics</i>	Displays file names and reference documentation: Read about the <i>sourcefile.hex</i> file in the <i>PSoC Designer User Guide</i> .
[Bracketed, Bold]	Displays keyboard commands in procedures: [Enter] or [Ctrl] [C]
File > Open	Represents menu paths: File > Open > New Project
Bold	Displays commands, menu paths, and icon names in procedures: Click the File icon and then click Open .
Times New Roman	Displays an equation: $2 + 2 = 4$
Text in gray boxes	Describes Cautions or unique functionality of the product.

2. Installation Guide



2.1 CY3271 Installation Instructions

Insert the CY3271 kit CD into your CD drive. This automatically launches the installer. If the autorun fails, then manually choose "autorun.exe" on the root of the CD, as shown in [Figure 2-1](#).

Figure 2-1. Selecting autorun.exe

Name	Size	Type
Files Currently on the CD		
Cypress		File Folder
installers		File Folder
autorun.exe	236 KB	Application
autorun.inf	1 KB	Setup Information
CY3271_Setup.exe	6,685 KB	Application
PSoClogo.bmp	1,686 KB	Bitmap Image

The installer presents three options. The first option launches the kit installer, which installs the following:

- PSoC Designer 5.0
- PSoC Programmer 3.00
- Cypress Sense and Control Dashboard
- Kit Contents

Figure 2-2. Kit Installer

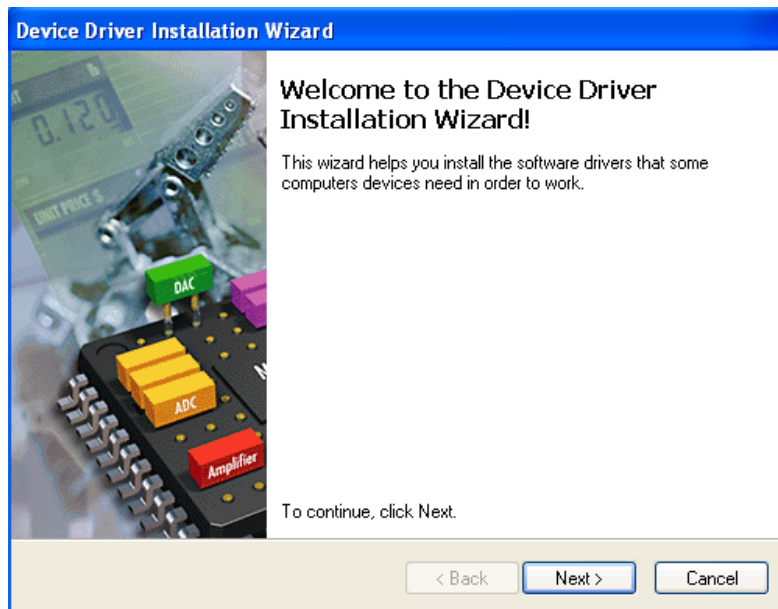


Click **Install CY3271 Kit and Tools** to start the kit installations. Click **Next** to start the installer and then choose **Install** to launch the PSoC Designer installer.

Click **Next** through the next several screens.

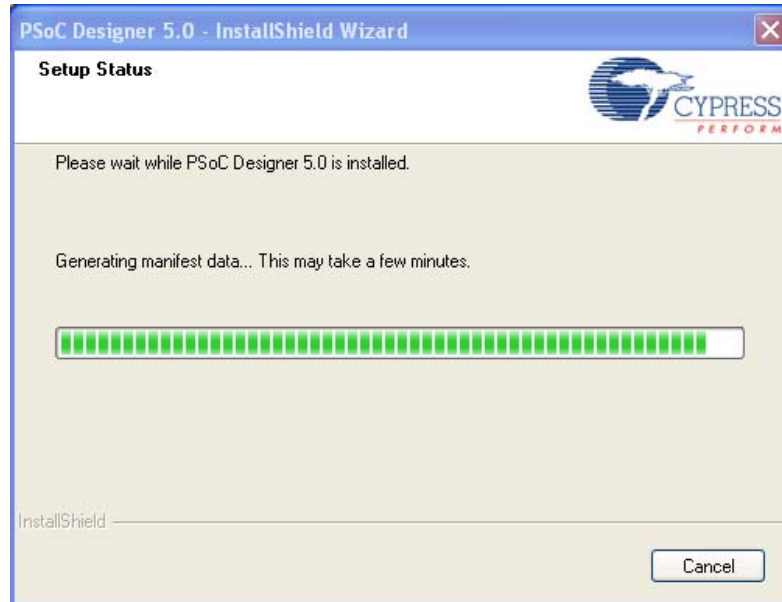
When the Device Driver screen appears, click **Next** and then **Finish**.

Figure 2-3. Device Driver Screen



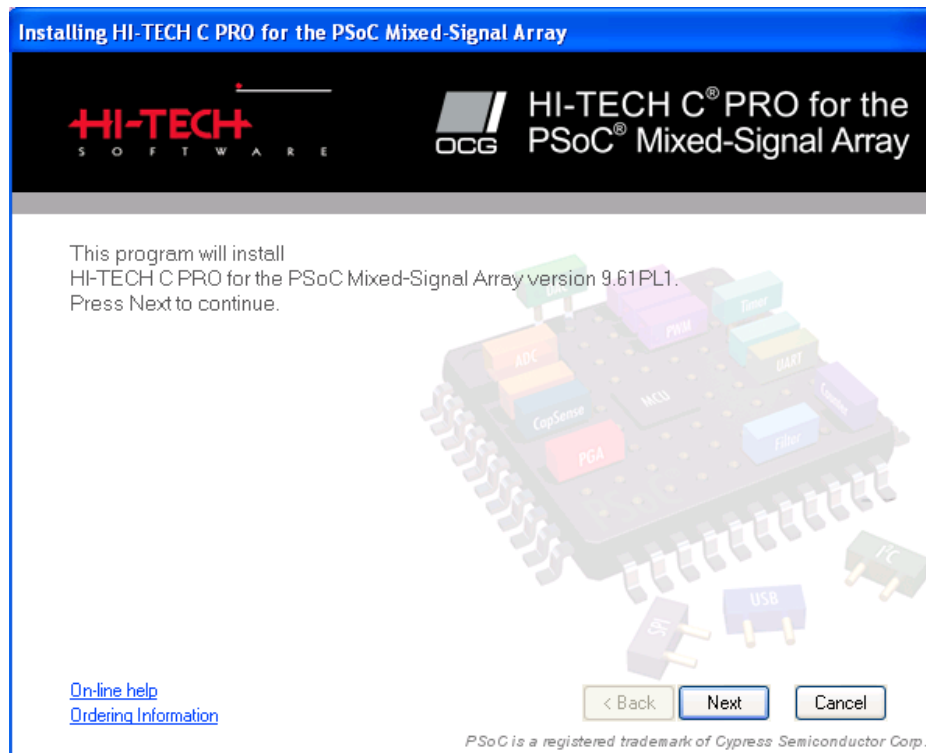
Wait for the Setup Status screen to complete. Then select **Finish** to complete the installation of PSoC Designer 5.0.

Figure 2-4. Setup Status Screen



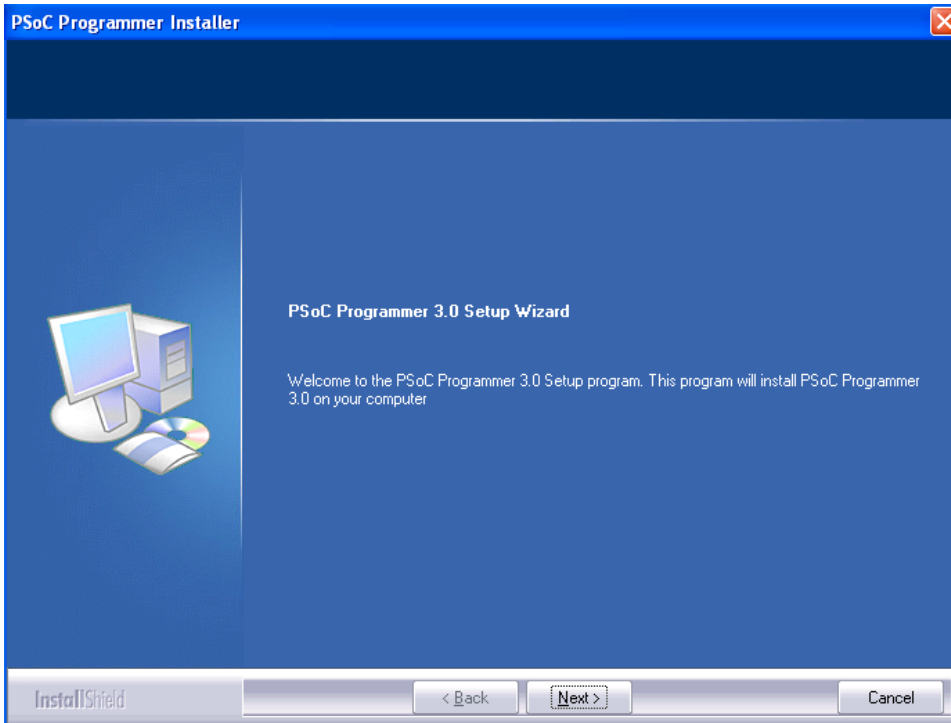
The Hi-TECH compiler for PSoC Designer begins installation.

Figure 2-5. Hi-TECH Compiler



PSoC Programmer 3.0 begins to install. Click **Next** through the next several screens.

Figure 2-6. PSoC Programmer Installer



Another Device Driver Installation Wizard appears. Click **Next** and **Finish** to complete the installation of PSoC Programmer.

The Sense and Control Dashboard Software setup wizard appears. Click **Next** through the next several screens to install the default configuration. This installer also installs Microsoft SQL server.

Figure 2-7. Installing SCD Software

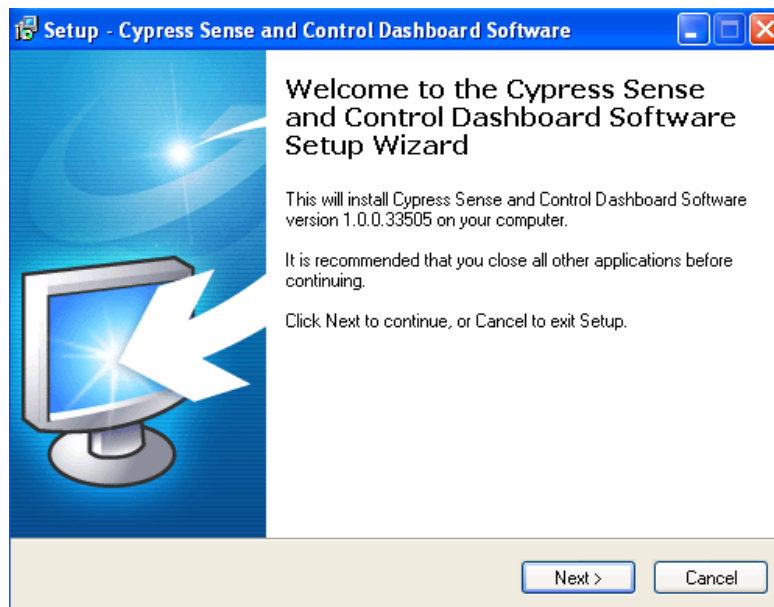
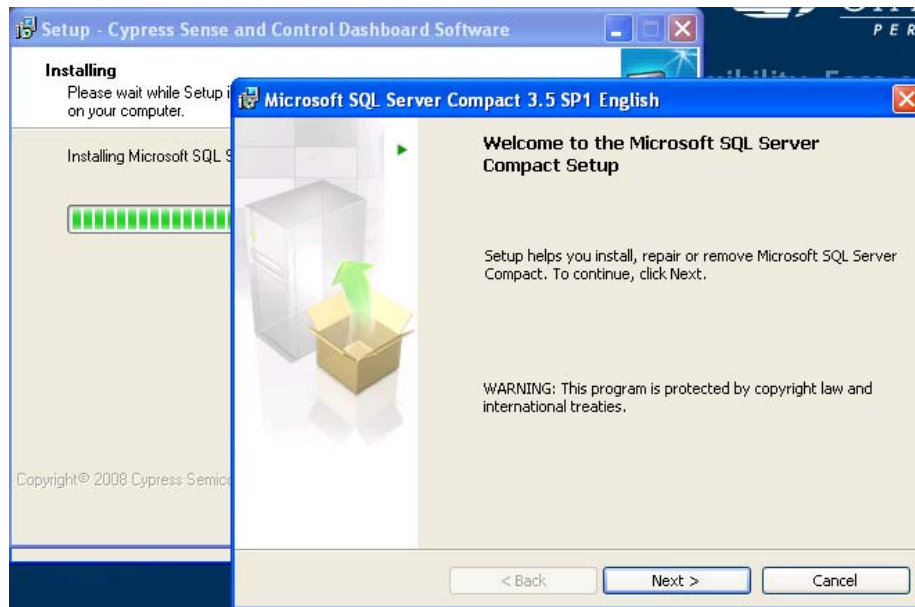


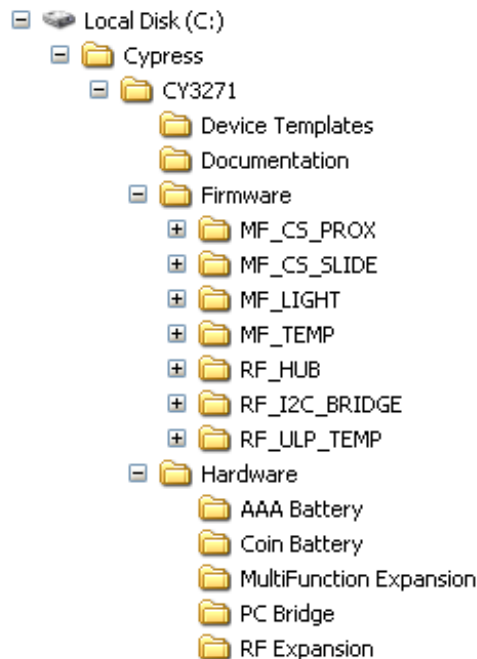
Figure 2-8. Installing Microsoft SQL Server



Click **Finish** to complete installing the CY3271 kit.

A directory structure similar to that shown in Figure 2-9 is created during the installation process.

Figure 2-9. Device Templates Directory



The device templates directory contains templates for all of the FTRF Design examples as outlined in the [Design Example Summary Table on page 21](#).

The firmware section contains the firmware projects for all of the projects used in this kit. Each project contains the source code as well as the compiled .HEX image enabling you to quickly pro-

gram each application into the hardware. It is advised to generate and build each project before making changes to the project source code.

The Hardware section contains the design files for the schematics and PCB layout. There are also in PDF format for ease of viewing.

3. Design Examples



3.1 Design Example Summary Table

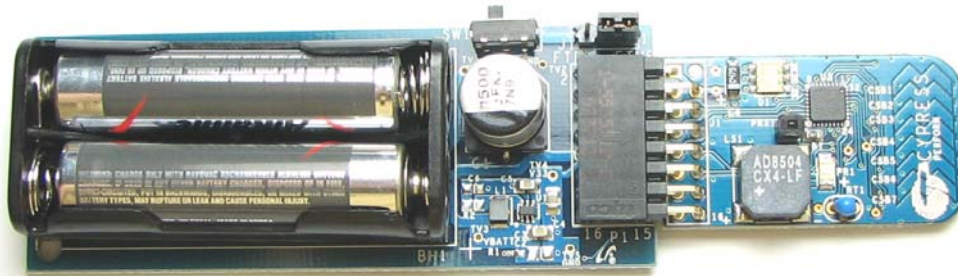
Design Example	Overview
CY3271 PSoC FirstTouch MultiFunction Expansion Card CapSense Slider (MF_CS_SLIDE)	The preprogrammed CapSense Touch Sensing demonstration shows how to use the CapSense Touch Sensing slider to control LED color. Run your finger across the CapSense Touch Sensing slider and notice how the color of the LED changes. The CY8C21434 PSoC that resides on the FTMF Expansion Card detects your finger's position on the CapSense Touch Sensing slider and controls the LEDs output.
Ultra Low Power Wireless Temperature Sensor	In this example, the hub board talks to a RF Expansion Board driven by AAA/CR-2032 coin cell. The RF Expansion Board transmits temperature data acquired from an onboard thermistor. The hub receives this data and sends it to the host PC, which displays the temperature data in text or graph form in the SCD dashboard.
FTMF Standalone Design Examples	
Light Sensor(MF_LIGHT)	The FTMF is programmed to demonstrate light sensing. The on board light sensor is used to alter the brightness of the LED based on the amount of light being detected.
CapSense Proximity Sensor(MF_CS_PROX)	The FTMF is programmed to demonstrate proximity detection.
Temperature Sensor(MF_TEMP)	The FTMF is programmed to demonstrate a temperature sensor. The measured temperature alters the LED display.
FTRF Design Examples	
CapSense Slider (MF_CS_SLIDE)	This example demonstrates the capacitive sensing capability of PSoC. You can change the color of the LED array by moving your finger across the CapSense slider.
CapSense Proximity Sensor(MF_CS_PROX)	This example demonstrates the capacitive sensing and proximity detection capability of the Cypress PSoC device. As you move your finger near and far from the proximity detection antenna, the red and green LED turn ON and OFF.
Temperature Sensor(MF_TEMP)	This example demonstrates the temperature sensing, thermistor reading, and calibrating capabilities of the PSoC device. When the temperature goes above or below a certain threshold, different colored LED (red, green, and blue) blink, and a buzzer is sounded out as an alert mechanism.
Light Sensor(MF_LIGHT)	This example demonstrates how to control the brightness of an LED array.

3.2 Out of Box Design Examples

3.2.1 CY3271 PSoC FirstTouch MultiFunction Expansion Card CapSense Slider

1. Connect the MultiFunction board to the AAA battery board as shown in [Figure 3-1](#).

Figure 3-1. Connecting the MultiFunction Board to the AAA Battery Board



2. Ensure that jumper J1 on the battery board is connected across pins 1 and 2, as shown in [Figure 3-1](#).
3. Move the switch to the ON position or away from the batteries.
4. Move your finger along the slider at the end of the MultiFunction board and observe how the LED changes color.

3.2.2 Ultra Low Power Wireless Temperature Sensor

This demonstration showcases a low-power RF solution that runs on a coin cell. It operates at 0 dBm RF power output.

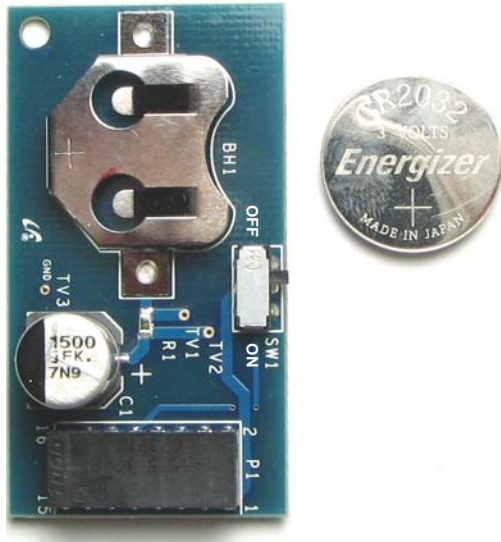
1. Install the software provided on the CD. Follow the instructions in the installation section before proceeding to these steps.
2. Connect the RF PC Bridge into any free USB port on your computer. The blue LED should start flashing indicating that the PC Bridge is enumerated on the USB bus
3. Program the PC Bridge slave processor with the latest installed FW. Follow the steps outlined in [1.2.2 CY3271 Software Overview](#). Select the .Hex file RF_HUB located in the Firmware\RF_HUB\RF_HUB\output directory.
4. Program the RF Expansion board with the latest installed FW. Again, follow the steps outlined in [1.2.2 CY3271 Software Overview](#). Select the .Hex file RF_ULP_TEMP located in the Firmware\RF_ULP_TEMP\RF_ULP_TEMP\output directory. Before programming be sure to connect the RF Expansion card into the PC Bridge interface connector.
5. Open the SCD Dashboard software GUI that is installed with the installation pack.

To access the SCD Dashboard software, go to: **Start > Programs > Cypress > Cypress Sense and Control Dashboard > Cypress Sense and Control Dashboard**.

Click Continue to Load the last configuration. The red LED should start blinking on the PC Bridge indicating that there is I2C activity between the SCD software and the PC Bridge slave RF Hub application.

6. Insert the coin cell into the battery board + side up.

Figure 3-2. Coin Cell into the Battery Board



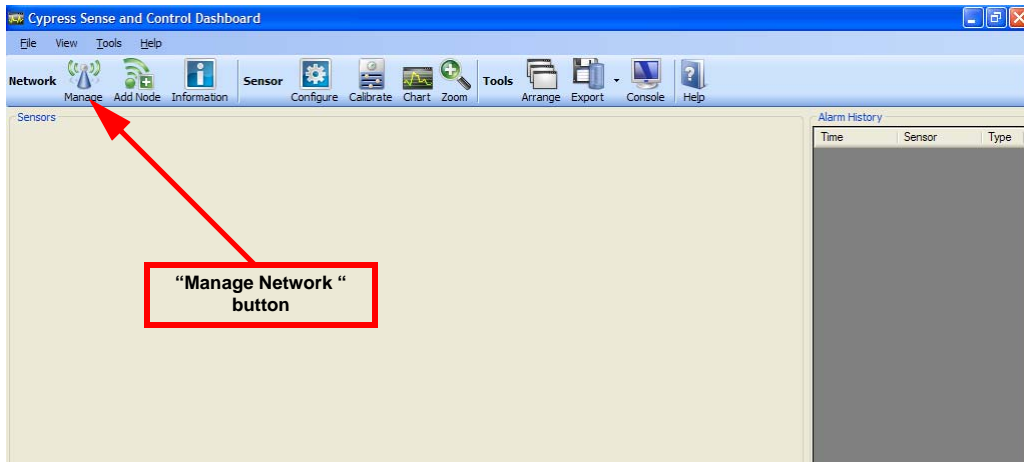
7. Connect the RF Expansion Board into the CR-2032 coin cell battery pack, as shown in [Figure 3-3](#).

Figure 3-3. RF Expansion Board into the CR-2032 Coin Cell Battery



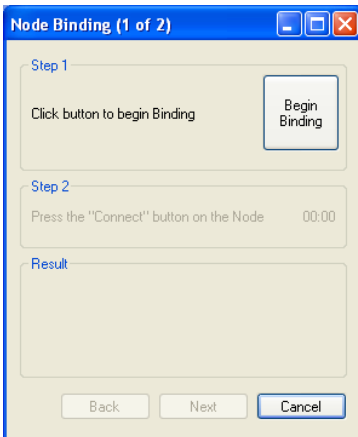
8. Switch on power to the RF Expansion Board by placing the position of the switch to ON position.
9. Place the PC Bridge in Bind mode using the SCD Dashboard. This is described in the following method:
 - Click the Manage Network button to add a new node.

Figure 3-4. Manage Network Button in the SCD Dashboard



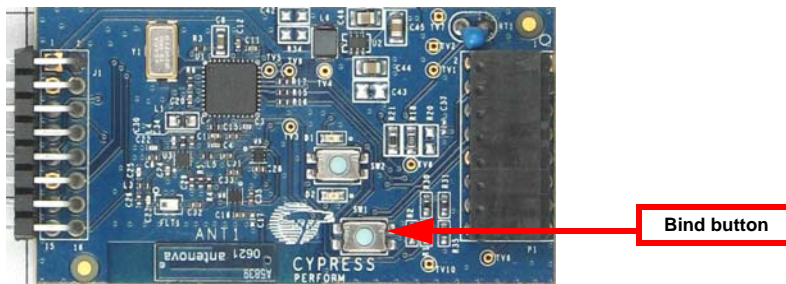
- In the Manage Network screen, click **Add..** to add a new node.
- On the Node Binding screen, click **Begin Binding**. When the green LED on the bridge lights up, the bridge is in bind mode (there is no need to press the button).

Figure 3-5. Node Binding Screen



8. Place the RF Expansion Board in **Bind** mode, by pressing the **Bind** button on the board when instructed by the GUI.

Figure 3-6. RF Expansion Mode (Bind Button)

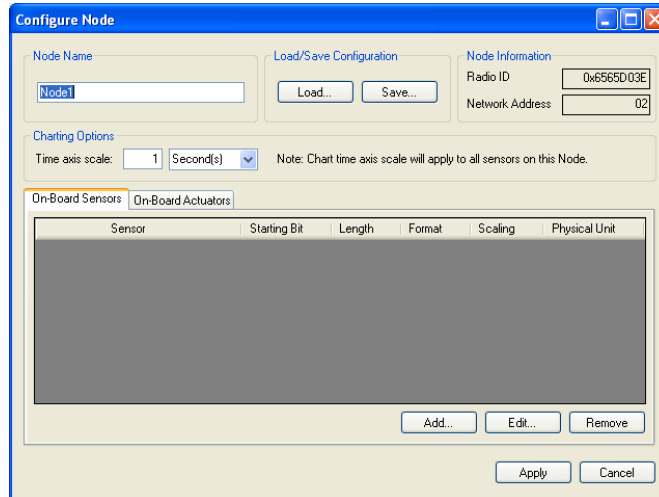


9. Verify the success of the bind and click **Next**.

10. After it is bound, name the node and the sensor. Then configure the data format as discussed in this step.

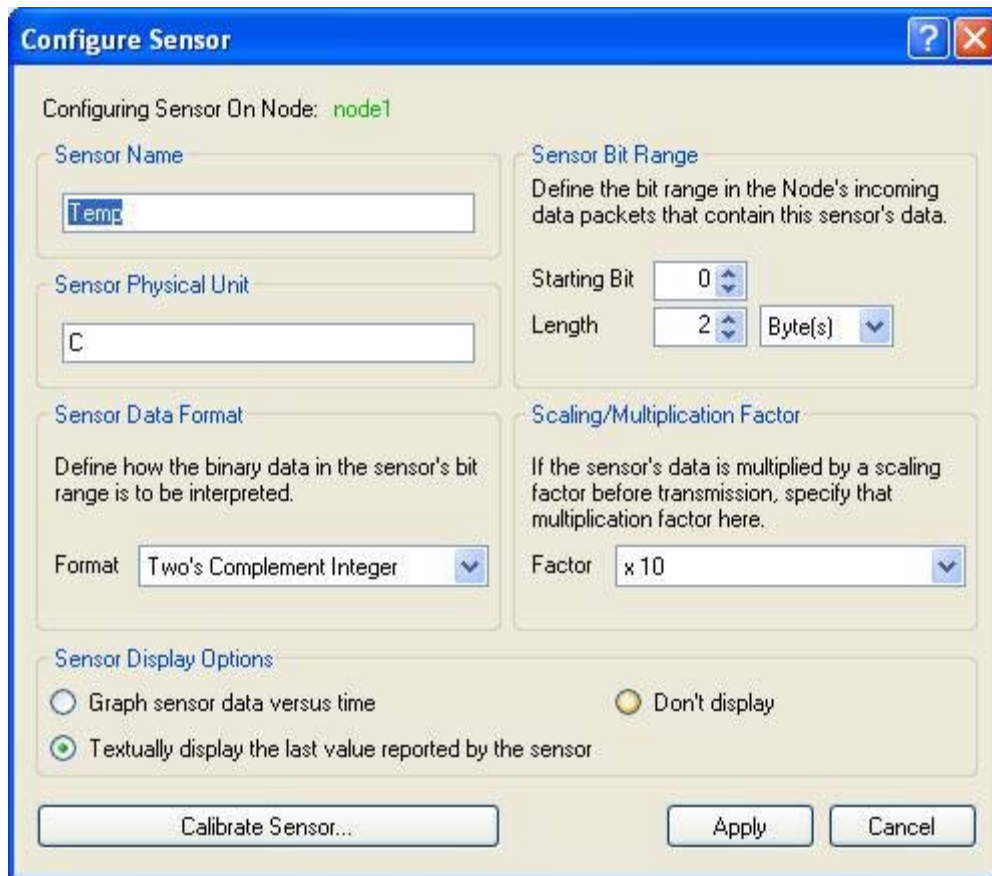
- In the Configure Node screen, click **Add..** to define the sensors for this node.

Figure 3-7. Configure Node Screen



Configure the sensor as shown in [Figure 3-8](#).

Figure 3-8. Define Sensors in the Configure Node Screen



11. Select graphical or textual mode of data display. The data is displayed in graphical or text format on the SCD screen.

12. Click **Apply** on all successive dialog boxes until the main SCD window reappears.

The red LED blinks at a five second interval when bound. Modify the report interval by pressing SW2 for more than two seconds. The red LED illuminates solid indicating that the report interval was advanced to the next interval. When SW2 is released the red LED flashes according to the selected interval:

- 1 = 1second
- 2 = 5 seconds
- 3 = 30 seconds
- 4 = 1 minute
- 5 = 5 minutes

The power on default is five seconds.

3.2.3 Wireless MultiFunction Demonstration

The MultiFunction demonstrations can be operated by programming the corresponding .hex file (CapSense, Light Sensor, Proximity Sensor, and Temperature Sensor) onto the Multifnction board. The example described in this section is specific for the Light Sensor. However, a similar approach

can be used for the other MultiFunction demonstrations. Follow the instructions outlined in [1.2.2 CY3271 Software Overview](#) for details on programming.

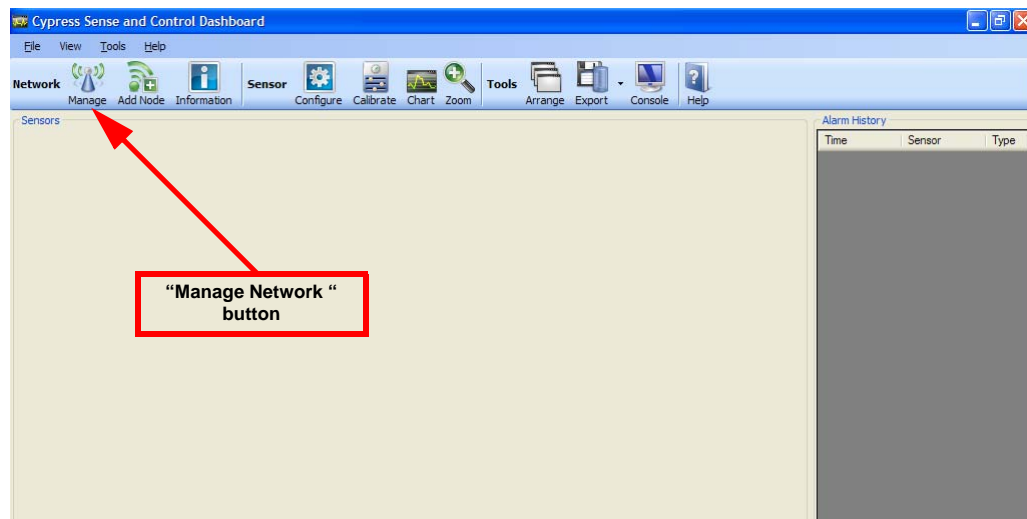
1. Install the software provided on the CD. Follow the instruction in the installation section before proceeding to the following steps.
2. Connect the RF PC Bridge into any free USB port of your PC/Laptop.
3. Program the RF Expansion board with the latest installed FW. Again, follow the steps outlined in [1.2.2 CY3271 Software Overview](#). Select the .Hex file RF_I2C_BRIDGE located in the Firmware\RF_I2C_BRIDGE\RF_I2C_BRIDGE\output directory. Before programming be sure to connect the RF Expansion card into the PC Bridge interface connector.
4. Open the SCD Dashboard software GUI that is installed with the installation pack.
5. Connect the RF Expansion Board into the 2x AAA alkaline cell battery pack and the MultiFunction board as shown in [Figure 3-9](#).

Figure 3-9. RF Expansion Board into the 2x AAA Alkaline Cell Battery and MultiFunction Board



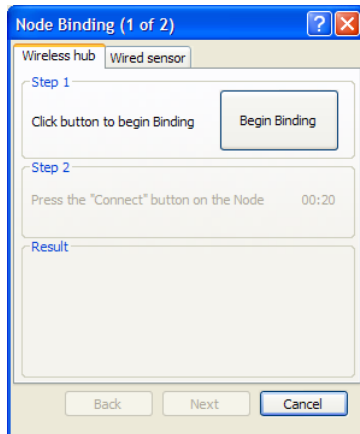
6. Switch on power to the RF Expansion Board by placing the position of the switch to ON position.
7. Place the PC Bridge in Bind mode using the SCD Dashboard. This is described in the following method:
 - Click the Manage Network button to add a new node.

Figure 3-10. Manage Network button in the SCD Dashboard



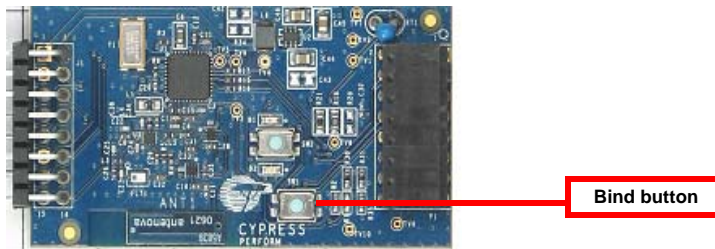
- In the Manage Network screen, click **Add..** to add a new node.
- On the Node Binding screen, click the **Begin Binding**. When the red LED on the bridge lights up, the bridge is in bind mode (there is no need to click the button).

Figure 3-11. Node Binding Screen



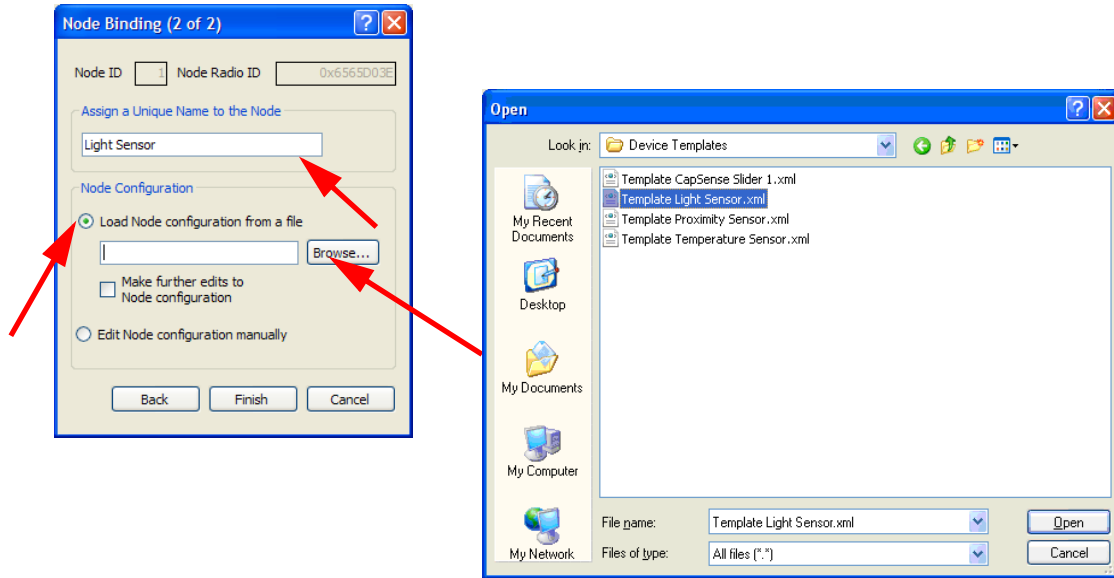
8. Place the RF Expansion Board in Bind mode, by clicking **Bind** on the board when instructed by the GUI.

Figure 3-12. RF Expansion Mode (Bind Button)



9. Verify the success of the bind and click **Next**.
10. After it is bound, name the node and the sensor. Then configure the data format as discussed in the following steps.
 - In the Configure Node screen, click **Load Node Configuration** radio button to load the stored configuration. Click **Browse** and load the 'Temperature Light Sensor.xml' file. This file is located in Cypress/CY3271/Device Templates.
 - Type 'Light Sensor' as the Node Name and click **Finish**.

Figure 3-13. Configure Node Screen



11. Select graphical or textual mode of data display. The data is displayed in graphical or text format on the SCD screen.

12. Click Apply on all successive dialog boxes until the main SCD window appears again.

A similar approach can be employed for the other MultiFunction demos. Program the MultiFunction board with the appropriate .hex file. Follow the binding method described for the Light Sensor. In Step 10, load the corresponding template from the 'Device Templates folder' and click **Finish**. Also the report interval can be modified as noted at the end of section [3.2.1 CY3271 PSoC FirstTouch MultiFunction Expansion Card CapSense Slider](#).

3.2.4 Standalone MultiFunction Demonstrations

The First Touch Multifunction (FTMF) Card provided with the CY3271 First Touch RF kit is able to support four stand alone demonstrations (CapSense Slider, Light Sensor, Proximity Sensor, and Temperature Sensor). Each demonstration has an associated PSoC Express project and is described in detail in sections 4.4, 4.5, 4.6 and 4.7.

You need to program the FTMF card with the appropriate firmware to operate the demonstrations. Programming is affected by connecting the FTMF card to the FTFC bridge, selecting the appropriate .hex file on PSoC Programmer software, and programming the CY8C21434 chip on board. [Figure 2-9 on page 19](#) is an illustration of where to locate the files.

These demonstrations are made possible by simply powering the FTMF card using the 2x AAA battery board.

1. Connect the FTMF board to the 2x AAA battery board as shown in [Figure 3-1](#).
2. Ensure that jumper J1 on the battery board is connected across pins 1 and 2.
3. Move the switch SW1 to the ON position or towards the FTMF board.

The following sections explain the purpose and output for each of the FTMF demonstrations.

3.2.4.1 MultiFunction Expansion Card CapSense Slider

Purpose:

The purpose of this project is to demonstrate the capacitive sensing capability of PSoC. You can change the color of the LED array by moving your finger across the CapSense slider.

What to observe:

- When the finger position on the slider is at the origin, the LED is OFF.
- When the finger position is in between the origin and the 1/3 mark of the width of the Capsense slider, the LED emits the color blue.
- When the finger position on the slider is between the 1/3 and 2/3 marks of the width of the Capsense slider, the LED emits the color green.
- When the finger position is between the far end and the 2/3 mark of the width of the capsense slider, the LED emits the color red.
- For all other slider positions, the LED is OFF. This includes the absence of a finger on the slider.

3.2.4.2 *MultiFunction Expansion Card Proximity Sensor*

Purpose:

The purpose of this project is to demonstrate the capacitive sensing and proximity detection capability of Cypress's PSoC technology. As you move your finger near and far from the proximity detection antenna, the red and green LED turn ON and OFF. This project consists of the following driver elements:

What to observe:

- When the finger is not close enough to the proximity antenna, the red LED is ON and the green LED is OFF.
- As the finger is brought closer and crosses the proximity threshold of the proximity antenna, the red LED turns OFF and the green LED turns ON.
- When the finger is gradually taken away again as it moves further than the proximity threshold, the red LED turns back ON and the green LED turns OFF.

3.2.4.3 *MultiFunction Expansion Card Temperature Sensor*

Purpose:

This project demonstrates the temperature sensing, thermistor reading, and calibrating capabilities of the PSoC device. Depending upon the temperature range within which a particular temperature reading is recorded, different colored LED blink (red, green, and blue). When the temperature goes above or below a certain threshold, a Buzzer is sounded out as an alert mechanism.

What to observe:

The Buzzer is sounded only when the temperature is between -10.1°C and 15.6°C, and also when the temperature is between 35.0°C and 55.1°C

- The Red LED is ON only if the temperature is between 26.7°C and 55.1°C.
- The Green LED is ON only when the temperature is between 21.1°C and 29.4°C.
- The Blue LED is ON only when the temperature is between -10.1°C and 23.9°C.
- LEDs blink only if the Blink Timer is triggered.

3.2.4.4 *CY3271 PSoC FirstTouch MultiFunction Expansion Card Light Sensor*

Purpose:

The purpose of this project is to demonstrate a light sensor. In this project, the light sensor is used to control the brightness of the LED array.

What to observe:

- The shadow cast by the palm of your hand diminishes the LED intensity
- Removing your palm leads to returns the LED intensity to its initial state

4. Firmware



4.1 PC Bridge Wireless Hub

The PC Bridge consists of two CY8C28494 processors. One device is used for the Master microprocessor that provides USB to I2C bridge functionality in addition to programming support for all kit elements. The second CY8C28494 processor acts as the wireless hub and communicates with the SCD SW application via an I2C interface to the master processor USB/I2C bridge.

This section discusses the design goals, architecture, firmware source code modules, and configuration options for the slave process or the wireless hub application processor.

4.1.1 Design Features

The Wireless Hub application uses the CyFISNP user module configured as a hub to communicate with the wireless nodes. All configuration and node data is communicated over a I2C interface.

4.1.2 Firmware Architecture

ROM/RAM Usage

Table 4-1. ROM/RAM Usage

	Total ROM (Bytes)	Total RAM (Bytes)
Hub functionality with Debug enabled	13116	703

There may be future extra build options, in which case this document will be updated.

Device Configuration for Slave CY8C24894

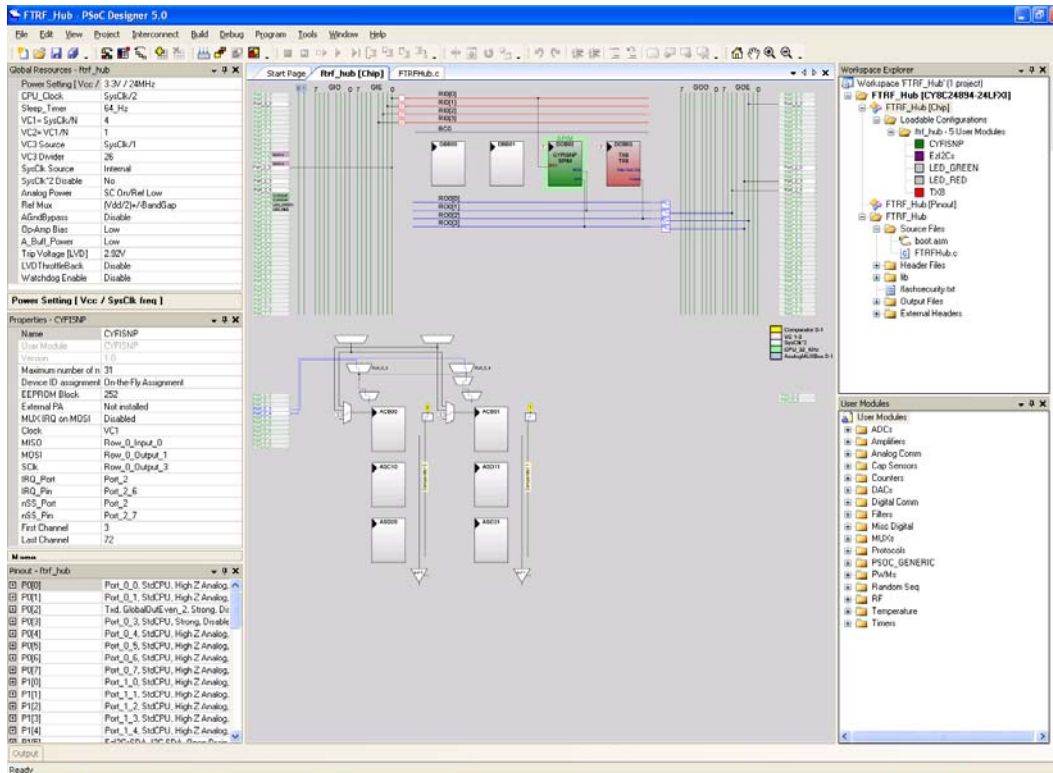
The slave CY8C24894 is configured using the Device Editor in PSoC Designer 5.0. The bridge uses the CYFISNP, EzI2Cs, LED (Red and Green), and TX8 user modules. The usage of each user module in the application is described:

- **CYFISNP.** This user module implements the entire Star network wireless protocol and all protocol modes, in addition to low level radio communication and radio control by the MCU.
- **EzI2Cs.** This user module implements the I2C slave functionality and takes care of data communication through the I2C interface with the master CY8C24894.
- **LED.** There are two instances of this user module: one is configured as RED and the other as GREEN. These implement the API to turn ON/OFF the LEDs according to the needs of the application. The application firmware can just call simple APIs to manipulate the LEDs.

The following sections describe how to configure these individual user modules.

Figure 4-1 shows the Device Editor with user module mapping.

Figure 4-1. Device Editor with User Module Mapping



4.1.2.1 Global Configuration

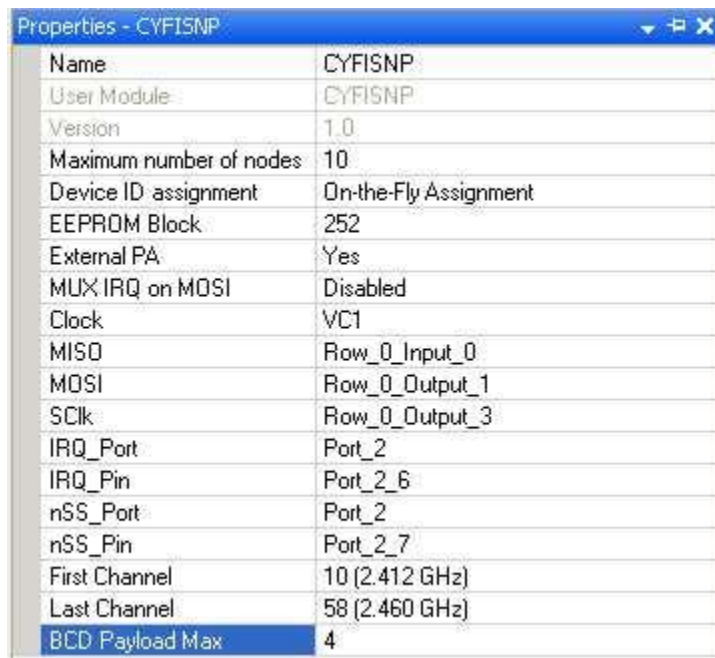
This section describes the global resources that are configured for the slave CY8C24894 running the hub application. These values must be modified with care because they affect the user modules discussed in later sections.

- **Power Setting.** 3.3V/24 MHz operation
- **CPU Clock.** This parameter is set to Internal (24 MHz). To run the CPU at 12 MHz, the CPU Clock/N must be set to '2'. This operating frequency provides faster code execution.
- **VC1.** This clock is chosen as 6 MHz. So VC1 = SysClk/N is chosen as 4.
- **VC2.** VC2 = VC1/N is chosen as 1
- **VC3 Source.** SysClk/1
- **VC3 Divider.** 26
- **SysClk Source.** Internal
- **SysClk*2 Disable.** No
- **Analog Power.** SC On/Ref Low
- **Ref Mux.** (Vdd/2)+/- Bandgap
- **AGndBypass.** Disable
- **OpAmp Bias.** Low
- **A_Buff_Power.** Low
- **Trip Voltage [LVD].** 2.92 V
- **LVD ThrottleBack.** Disable
- **Watchdog Enable.** Disable

4.1.2.2 CYFISNP User Module

The CYFISNP user module is at the core of the hub application firmware. All wireless communication with the remote nodes is taken care of by this user module. You must include this user module firmware in the application and make API calls to this user module from the main application. The CYFISNP is actually comprised of two components. A higher level of medium access protocol firmware controls the network level details, such as node binding, finding a quiet channel, data communication with nodes, and so on. Then there is a lower level radio driver firmware that allows the microcontroller to control the CYRF7936 2.4 GHz CyFi transceiver radio over SPI. The user module also has a E2PROM component of 252 bytes that stores the node records such as MID, Node ID, and others.

Figure 4-2. CYFISNP User Module



Properties - CYFISNP	
Name	CYFISNP
User Module	CYFISNP
Version	1.0
Maximum number of nodes	10
Device ID assignment	On-the-Fly Assignment
EEPROM Block	252
External PA	Yes
MUX_IRQ on MOSI	Disabled
Clock	VC1
MISO	Row_0_Input_0
MOSI	Row_0_Output_1
SCLK	Row_0_Output_3
IRQ_Port	Port_2
IRQ_Pin	Port_2_6
nSS_Port	Port_2
nSS_Pin	Port_2_7
First Channel	10 (2.412 GHz)
Last Channel	58 (2.460 GHz)
BCD Payload Max	4

This user module supports up to 32 nodes. In the default configuration of this module, the node ID is assigned automatically (on the fly assignment) by the Hub. However, there is provision when nodes may ask for specific IDs.

The SPI lines of MOSI, MISO, SCLK, and the IRQ and nSS lines are properly configured.

4.1.2.3 EzI2Cs User Module

The EzI2Cs user module takes care of I2C communication with the master CY8C24894 device on the bridge. The user module has API which specifies the memory location and size which is accessed by the master. All I2C communication is initiated by the master. Whenever the master polls for data, the EzI2Cs, according to the nature of the request, place the data that need to go upstream on the specified memory location. The master comes and reads from that location without application firmware intervention.

The default I2C slave address for this hub application is 8-bit slave address of 4. It also uses the fast I2C clock option of 400 kHz and is configured on P1.5 and P1.7 of the slave CY8C24894.

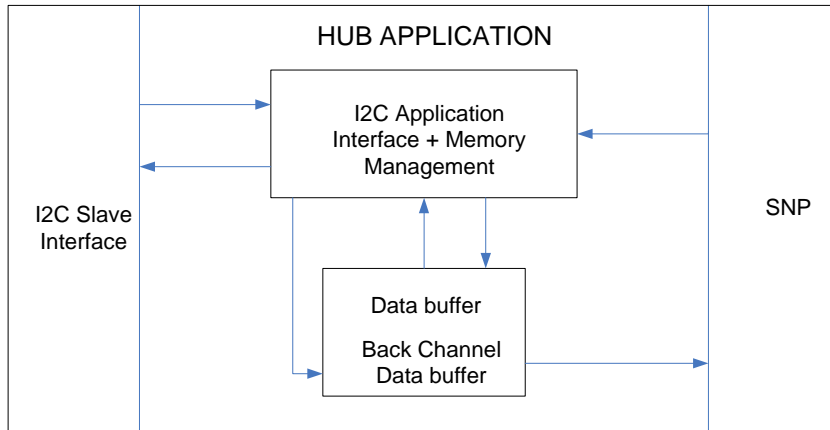
4.1.2.4 *LED_x User Module*

There are two instances of this user module: one is configured as the Red LED and the other as Green LED. These modules implement LED switching states, turning On/Off, Toggle, and so on. The application firmware simply needs to make API calls to these user modules. Refer to LED usage for more information on how application firmware handles the LEDs.

The LEDs are configured to P3.0 and P3.1 of the slave CY8C24894. They have a Drive of Active HIGH.

4.1.3 Firmware Model

Figure 4-3. Functional Block Diagram for the Hub Application

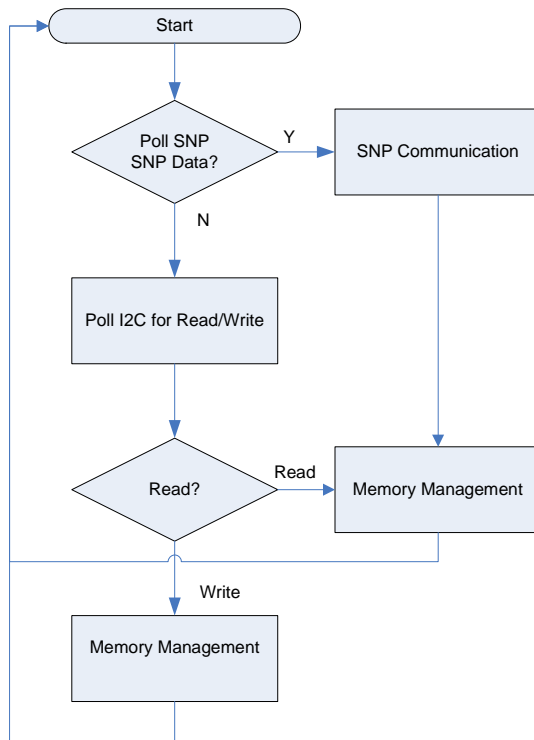


It is evident that the Hub application requires three distinct blocks:

- Communication with SNP
- Memory and Buffer Management
- I2C Application Interface Management

As a result, the main application firmware would have an architecture based on the following state machine shown in [Figure 4-4](#).

Figure 4-4. State Machine Architecture



The separate threads that handle the different functionalities are discussed in the following sections.

4.1.3.1 *Communication with SNP*

The main function calls this function to communicate with the star network protocol.

The framework of this part of the firmware is based on the following pseudocode:

```
void Serve SNP Protocol
if SNP Packet pending
  Put all packets in Data Buffer
  Process packets after SNP Buffer is emptied based on Packet type in
  packet header
  Process packets that don't need to be sent to host and delete to free up
  buffer
  Call memory management
if Back Channel data requested by SNP
  Poll host GUI for back channel data
  Load back channel data buffer
  Call memory management
else Release SNP packet buffer and exit
```

Keep polling the SNP at regular intervals to check if it contains data that needs servicing.

This is implemented in the protocol by an API function call **ServeSNPPackets ()**.

The function first polls the SNP for data through and CYFISNP API calls **CYFISNP_RxDataPend**. If data is pending, then different tasks are performed depending on the received packet type.

If the packet is of data type, then it is written into an Application Buffer, and the Memory management section is called which handles the packets in the buffer. If a data packet is successfully written in the Application Buffer, then the Write pointer *pNextWrite* is incremented to point to the next location in the application buffer. The count value indicating number of packets in the buffer, **numberOfPacketsInBuffer**, is also incremented. These two are implemented in an API called **WriteBufferManager()**.

4.1.3.2 *Memory Management*

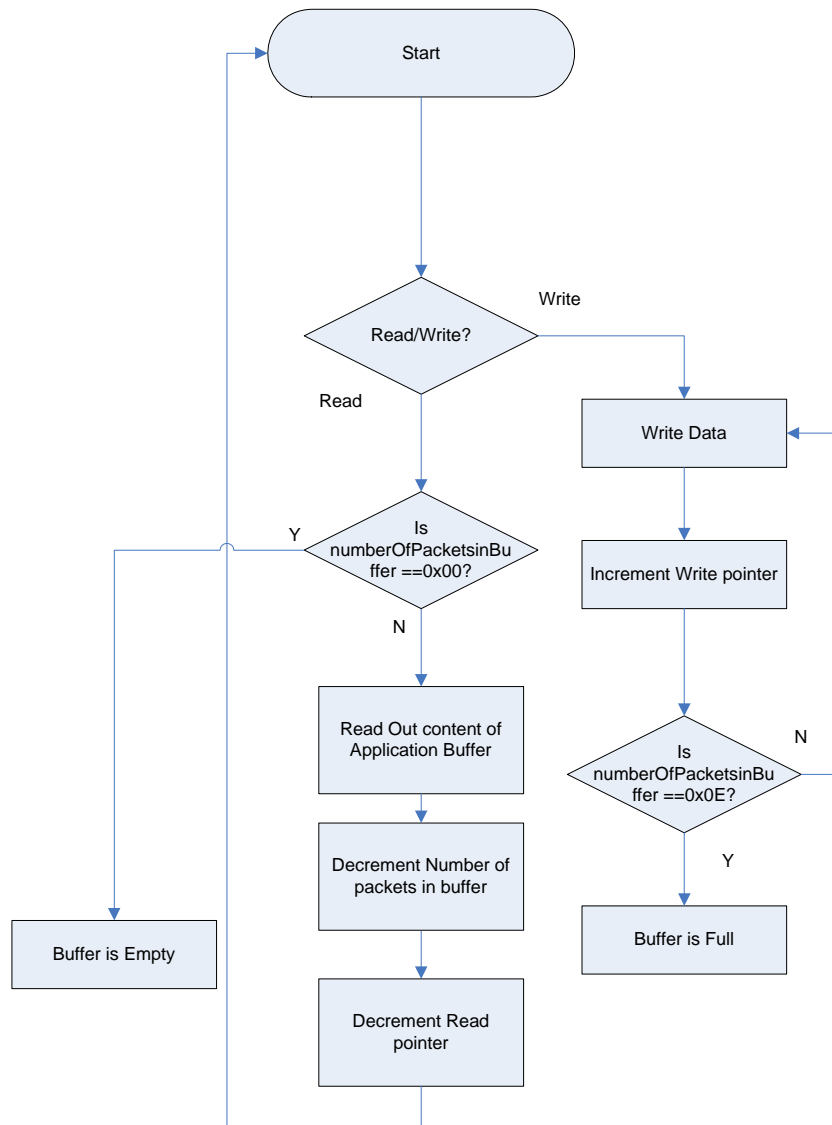
This function manages the data movement between the protocol data buffers, including the back channel data buffer and the Application firmware. This management algorithm prevents buffer overflows and prevents the I2C master bridge from receiving junk data when it polls the slave hub, when there are no packets in the buffer. This guarantees that data is read and written in an efficient manner that saves latency, and also data is not overwritten or lost due to Buffer overflow error. The Data buffer and the back channel Data buffer are implemented as circular buffers.

Data buffer = 252 bytes (18 bytes X 14)

Back channel Data buffer = 128 bytes (16 bytes X 8)

[Figure 4-5 on page 39](#) shows the framework of the two functions that implement memory management.

Figure 4-5. State Machine



Memory management is implemented by the function **BufferManagement()** in the hub firmware. This API is called immediately after the APIs `WriteBufferManager()` and `ReadBufferManager()`. This function checks for four important conditions and takes actions based on these conditions. The conditions are:

- Buffer Full
- Buffer Empty
- Read pointer has reached the last location in the buffer
- Write pointer has reached the last location in the buffer and the buffer is full or not

The first two conditions are checked by checking the count value of the number of packets in the buffer **numberOfPacketsInBuffer**.

- If this count equals zero, then the buffer is empty and in such a condition all pointers are reset to the initial base location of the buffer.
- If this count equals **MAX_BUFF_PKTS**, then the buffer is full. An LED lights up and the buffer is no longer written with a packet from the SNP. Whenever one packet is read from this condition by the I2C master, this condition is immediately cleared and the buffer accepts data from SNP again.

The buffer is designed as a circular buffer. As a result, if the read pointer referencing the application buffer **pNextRead** reaches the last location of the buffer **pEnd** (third condition is met), then immediately after the current read, the pointer is reset to the base address of the buffer **pBeg**. This ensures the pointer never drifts off into random regions of the RAM, and always initializes read from the base address of the Application Buffer.

Similarly, when the write pointer referencing the application buffer **pNextWrite** reaches the last location of the buffer **pEnd**, and there are no reads from the I2C master (that is, if the read pointer has not moved it implies that the buffer is full), no more writes are allowed and **pNextWrite** is stuck at the last location. However, after the data starts getting polled out by I2C master, and the buffer is not full.

4.1.3.3 I2C Interface Management

The I2C is implemented using the EzI2Cs user module available in PSoC Designer 5.0. For details about the user module, refer to the EzI2Cs user module data sheet.

The top level function call that initiates I2C communication between the master CY8C24894 and the slave CY8C24894 is **CheckHostRequest()**.

After this API is called, the master CY8C24894 writes a command into the RAM buffer of the slave. The address and size of the command is specified by the user module API call, **EzI2Cs_SetRamBuffer (sizeof(I2CBuffer), sizeof(I2CBuffer), pI2C)**, at the beginning of the application firmware.

The firmware checks the first byte of the command byte and, depending upon the command byte, executes several functions: for example, reading out data from the application buffer and freeing up buffer space, sending back channel data to a particular node, getting the node configuration of a specific node, and others. These functionalities are implemented using switch statements based on the first command byte. The following code snippet illustrates the API call.

```
switch (I2CBuffer[COMMAND_BYTE])

    //numberOfPacketsInBuffer--;// decrement number of packets
so we can keep track of how many packets are left in the buffer
    length = pNextRead->length;
    devId = pNextRead->devId;

    //stuff I2C buffer with data

    I2CBuffer[1] = numberOfPacketsInBuffer;
    I2CBuffer[2] = devId;
    I2CBuffer[3] = length;
    I2CBuffer[4] = pNextRead->rssI;
    for(index=5; index < (length+5); index++)
    {
        I2CBuffer[index] = pNextRead->payload[payloadIndex];
        payloadIndex++;
    }

    ReadBufferManager();
```



```

    }
    else
    {
        I2CBuffer[1] = 0xFF;    // if no more messages are in the
buffer return 0xFF
    }

    I2CBuffer[RESPONSE_BYTE] = FETCH_NXT_PKT_RSP;

    #if defined(DEBUG)
    LED_RED_Invert();
    #endif

    break;

```

There are seven similar case statements, where each case statement implements a different functionality. The case statements are:

- FETCH_NXT_PKT
- SEND_PKT
- GET_NODE_CFG
- UNBIND_NODE
- ENTER_BIND_MODE
- GET_PROTOCOL_STATE
- GET_LAST_BIND_RESULT

Each case has individual hex values 0x01 to 0x07 assigned, and these are defined in the application header file FTRFHub.h.

Note that FETCH_NXT_PKT (when a data packet is read by the I2C master and sent up to the host GUI) occurs only when the application buffer is not empty. After FETCH_NXT_PKT occurs, **numberOfPacketsInBuffer** is immediately decremented and **BufferManagement()** is called through the API **ReadBufferManager()**.

All the mentioned master I2C requests have a response from the slave. These responses also have hex values 0x81 to 0x87 assigned to them, and are also defined in the application header file FTRFHub.h.

4.1.3.4 Secondary Application Implementations

Apart from the above three fundamental and major tasks, the hub application firmware also performs the following tasks to ensure proper functioning of the hub:

- The bind button is scanned continuously to detect a button bind request from the user.
 - Implemented by the API `checkBindButton()`
- A test back channel data packet is sent if a device requests a back channel data packet, by setting BCDR bit when the host GUI has no data to send.
 - Implemented by the API `SendBackChannelData(BYTE devID)`

4.2 RF Expansion Card Ultra Low Power Temperature Sensor

This section discusses the design goals, architecture, firmware source code modules, and configuration options of the FTRF Temperature Sensor Node.

4.2.1 Design Features

The CY3271 FTRF Kit uses a PSoC CY8C27443 on the RF Expansion Board. This is the application processor, which controls the CyFi Radio and other components on the board. Since the FTRF expansion card has its own PSoC, you can also operate it by removing it from the battery pack and inserting it into your target hardware or other development platforms.

Figure 2-9 on page 19 is an illustration of where to locate project files.

This section describes the firmware implementation of using a temperature sensor on the remote node to transmit temperature data back to the host or hub.

4.2.2 Firmware Architecture

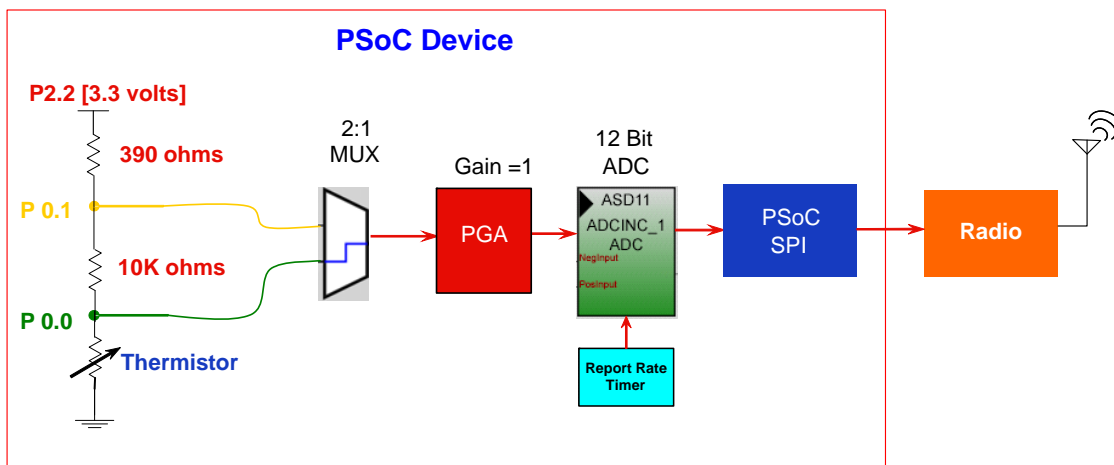
4.2.2.1 ROM/RAM Usage

Table 4-2. ROM/RAM Usage

	Total ROM (Bytes)	Total RAM (Bytes)
Node functionality with TX8 Debug enabled	11051	162

4.2.2.2 Functional Block Diagram of the Temperature Sensor Node

Figure 4-6. Temperature Sensor Node Block Diagram



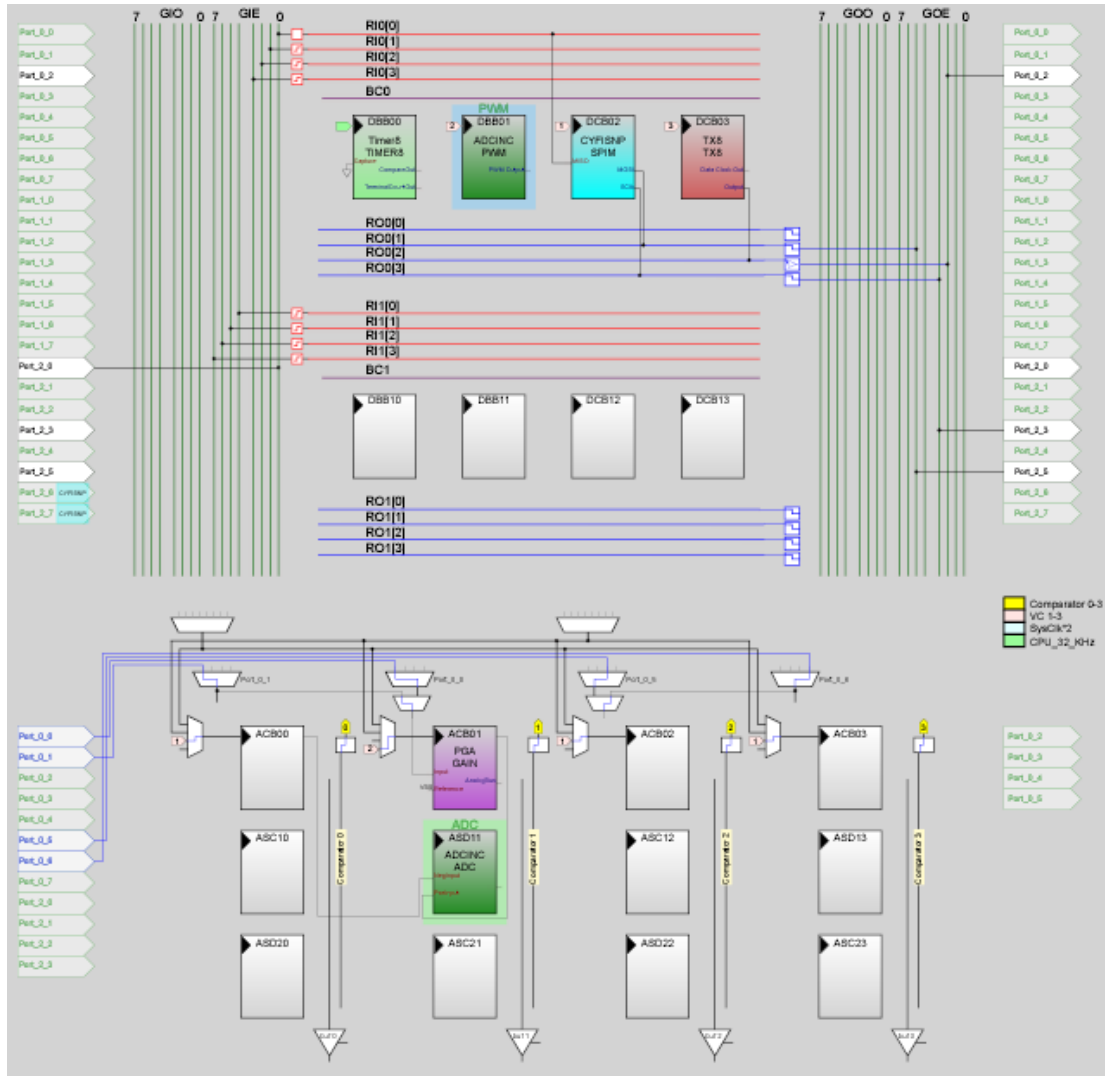
The Temperature Sensor node design consists of the following modules:

1. Analog Input Routing
2. PGA
3. Incremental ADC
4. CYFISNP
5. Timers for Report Rate

4.2.2.3 Device Configuration for CY8C27443 in the RF Expansion Board

The CY8C27443 is configured using the Device Editor in PSoC Designer 5.0. The node uses the CYFISNP, Timer8, ADCINC, PGA, and TX8 User Modules. The following section discusses the configuration of global resources and user module parameters for every user module used.

Figure 4-7. Chip Layout View Showing User Modules Placement and Routing

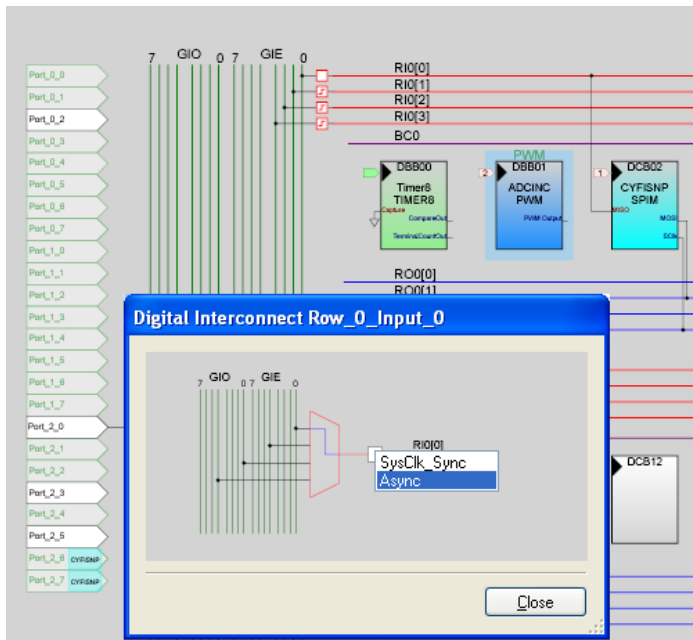


4.2.2.4 Global Configuration

This section describes the global resources available for configuring the CY8C27443, which runs the Node application. Modify these values carefully because they may affect the User Modules discussed in later sections.

- **Power Setting.** 3.3V/24 MHz operation. Supply voltage at 24 MHz.
- **CPU Clock.** This parameter is set to SysClk/2. To run the CPU at 12 MHz, SysClk/N must be set to '2'.
- **VC1.** This clock is set at 6 MHz. As a result, VC1= SysClk/N set as 4. The SPI block in the CYFISNP user module is clocked with VC1. When the SPI clock in PSoC is set at 2 MHz or higher, the MOSI input must be selected as 'Asynchronous' as shown in [Figure 4-8 on page 44](#).

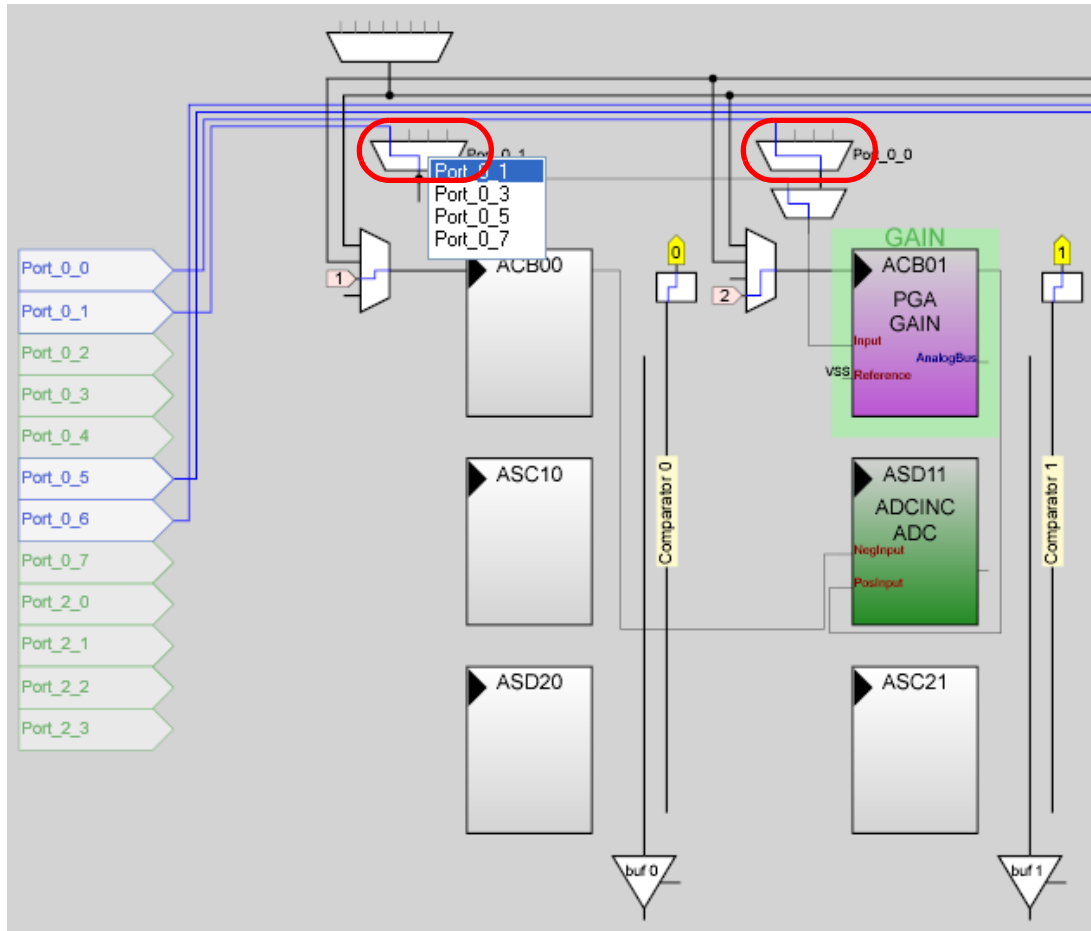
Figure 4-8. Global Configuration



- **VC2.** VC2=VC1/N is selected as 3. As a result, VC2 outputs 2 MHz and is provided as the clock to the incremental ADC.
- **VC3 Source.** SysClk
- **VC3 Divider.** 26 - Clocks the TX8 block.
- **SysClk Source.** Internal (Internal Main Oscillator)
- **SysClk*2 Disable.** Yes - Power savings
- **Analog Power.** All Off - Analog circuitry is powered using firmware during readings.
- **Ref Mux.** (Vdd/2)+/- (Vdd/2)
- **AGndBypass.** Disable
- **OpAmp Bias.** Low
- **A_Buff_Power.** Low
- **Switch Mode Pump.** OFF, not used in design. Saves power when OFF.
- **Trip Voltage [LVD].** 2.92 V
- **LVD Throttleback.** Disable
- **Watchdog Enable.** Enable

4.2.2.5 Chip Layout Configuration

Figure 4-9. Configuration of Input Muxes to Route Pins P0.0 and P0.1



Select pins P0.0 and P0.1 as the default pins to the input muxes as shown in [Figure 4-9](#).

4.2.3 User Modules used in the Temperature Sensor PSoC Project

4.2.3.1 CYFISNP User Module

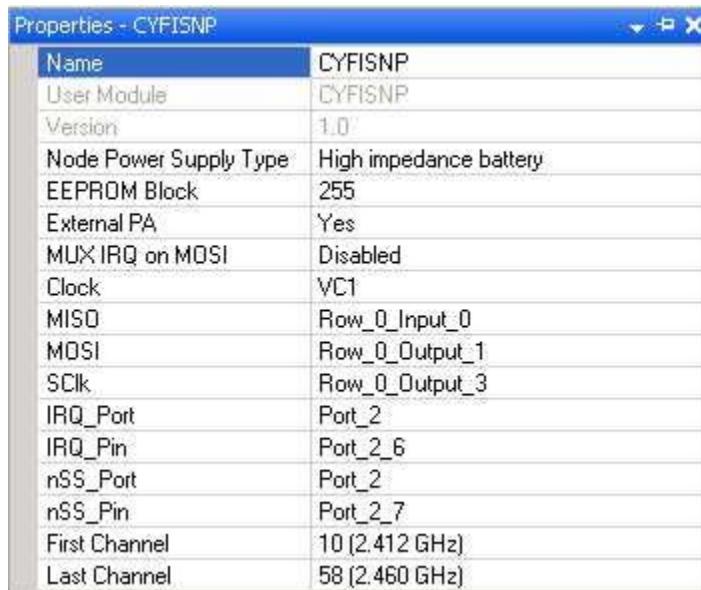
The CYFISNP user module is at the core of the node application firmware. This user module controls all wireless communication with the hub. In addition, this user module also implements the entire Star network wireless protocol, its modes, and the low level communication between the radio and the MCU. API calls to the CYFISNP user module affect control.

The CYFISNP consists of the following components:

- A higher level of medium access protocol firmware that takes care of the network level details, such as node binding, finding a quiet channel, data communication with nodes, and others.
- A lower level radio driver firmware that allows the microcontroller (CY8C27443) to control the CYRF6936 radio over SPI.
- The user module contains an EEPROM component to write the parameters of the Hub to the node's flash.
- The user module also consists of the Sleep timer configured at 64 Hz. This is required for timing the various events and modes in the protocol.

The SPI lines of MOSI, MISO, SCLK, and the IRQ and nSS lines are correctly configured according to the PSoC drive configurations.

Figure 4-10. CYFISNP User Module



Name	CYFISNP
User Module	CYFISNP
Version	1.0
Node Power Supply Type	High impedance battery
EEPROM Block	255
External PA	Yes
MUX IRQ on MOSI	Disabled
Clock	VC1
MISO	Row_0_Input_0
MOSI	Row_0_Output_1
SCLK	Row_0_Output_3
IRQ_Port	Port_2
IRQ_Pin	Port_2_6
nSS_Port	Port_2
nSS_Pin	Port_2_7
First Channel	10 [2.412 GHz]
Last Channel	58 [2.460 GHz]

Node Power Supply

The node is powered by Alkaline cells that have low internal resistance. 'High Impedance Battery' is selected.

EEPROM Block

This parameter defines the EEPROM block number used to store network parameters. The last block in the flash is selected to write and store the parameters. This block must be marked as unprotected in the flashsecurity.txt file.

External PA

This parameter controls the operation of the external Power Amplifier. Coin-cell batteries cannot supply high peak current for operation required by external PAs. As a result, external PAs are not recommended for such applications. However, option 'YES' must be selected in the case of the Ultra low-power Temperature Sensor project in FTRF because the PA must be bypassed on the RF Expansion Board for a coin-cell application. Next, the Config.h file must be modified to limit the PA to less than 0 dBm. This is discussed in the section [Config.h on page 48](#).

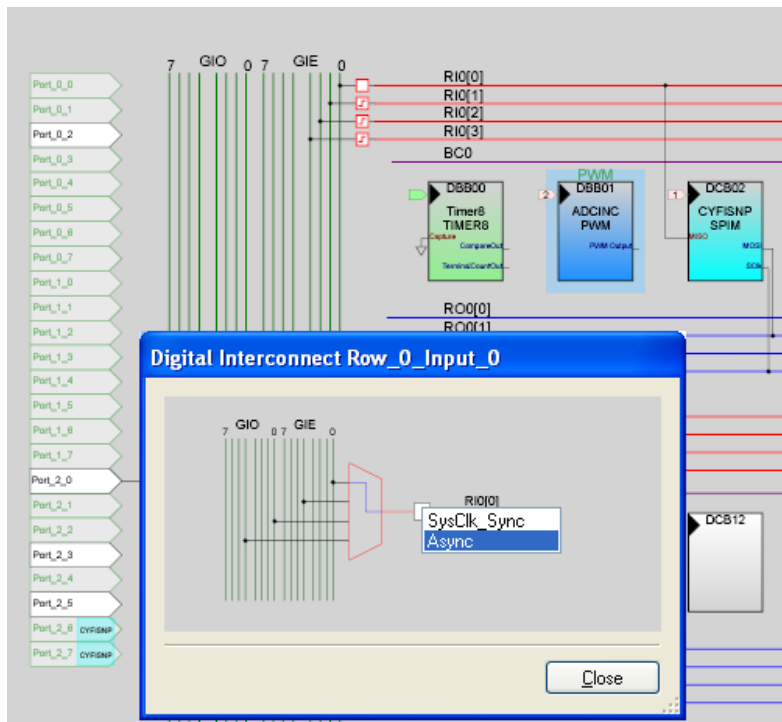
MUX IRQ on MOSI

The CYRF793x device IRQ pin may be multiplexed onto the MOSI pin. The temperature sensor project does not use the multiplexing option.

Clock

Selects the SPIM Block Clocking source. The clock rate is set to 6 MHz and the 'Async' option is selected for the line going to the SPI clock.

Figure 4-11. Clock



MISO

The Master-In-Slave-Out input (MISO) signal is routed to **Row_0_Input_0** net.

MOSI

The Master-Out-Slave-In (MOSI) output signal is routed to **Row_0_Output_1** net.

SClk

This output clock is generated by the SPI Master, and is Routed to **Row_0_Output_3** net.

IRQ_Port, IRQ_Pin

Selects the IRQ pin. Choose Pin P2.6 for the IRQ pin because muxing with MOSI is not opted.

nSS_Port, nSS_Pin

The SlaveSelect pin is selected. Pin P2.7 is selected to route the nSS signal.

First Channel

This parameter controls the low limit of the channel range of CYFISNP protocol. This avoids spilling over to other channels, especially when using external PA. This is an important consideration when seeking FCC or similar RF certification.

When a PA option is selected in the user module configuration, the First Channel defaults to Channel 10 (changeable by the developer). For coin-cell applications, the external PA option is not selected. However, if the hub uses the external PA, and as a result has restricted channel subset, the node must use the same first and last channels.

Last Channel

This parameter controls the high limit of channel range of CYFISNP protocol. This avoids spilling over to other channels, especially when using external PA. This is an important consideration when seeking FCC or similar RF certification.

When a PA option is selected in the user module configuration, the Last Channel defaults to Channel 58 (changeable by the developer). For coin-cell applications, the external PA option is not selected. However, if the hub uses the external PA, and as a result has restricted channel subset, the node must use the same first and last channels.

Config.h

The CYFISNP User Module generates a file named CYFISNP_Config.h that contains tables and variables that you can configure for your project. Most of the code in the generated header and source files in your project is regenerated each time your project is built. As a result, any changes that you make to this code are lost. The exceptions to this rule are sections of code contained between @PSoC_UserCode_name@ and @PSoC_UserCode_END@.

You can change the following items in CYFISNP_Config.h:

■ @PSoC_UserCode_PaPhyTblInternal

This table allows you to customize the eight possible RF power steps for the Dynamic PA feature, when the Internal PA is used. This list must be monotonic for proper operation. This table starts at PA4 since there are no significant current savings below PA4. The table stops with PA6 because the maximum internal PA is PA6.

■ @PSoC_UserCode_PaPhyTblExternal

This table allows you to customize the eight possible RF power steps for the Dynamic PA feature. This list must be monotonic for proper operation. This table is used only if the External PA option is set to 'YES'. This table provides the configuration for the following three options:

FTRF Coin-cell Application

For FTRF coin-cell application, the 'External PA' option must be set to 'YES' in the User Module parameter setting. This option with the External PA table setting is required to effectively disable the PA on the board.

By default, the external PA up to +10 dBm is chosen (second table under External PA). For FTRF coin-cell applications, this table must be commented out and the first table must be uncommented. Ensure that the table that is uncommented goes only up to PA6.

Default Option with External PA up to +10 dBm

This is the default setting or table (Table 2) when external PA is chosen in the User Module parameters. The external PA is configured such that the RF output power can only go up as high

as +10 dBm in this case. This is due to the RF power restrictions imposed by Europe and Japan. The power can be increased to +20 dBm only in the United States and Canada.

Option with External PA up to +20 dBm

RF Applications in most countries other than Europe and Japan (unless there are regional or other restrictions) can go up to +20 dBm. The default table (Table 2) must be commented out and the +20 dBm table (Table 3) must be uncommented for the +20 dBm RF power to be output by the external PA.

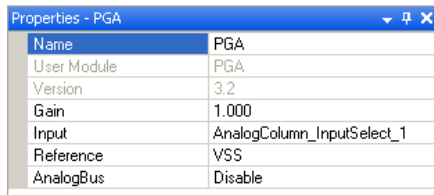
■ **PA_LEVEL_BIND**

This section helps you to set the PA_LEVEL_BIND variable. This variable allows you to change the PA level used during binding. Changing the PA level influences the binding distance. The default value of PA4 is good for short distances (within a few feet). For coin-cell applications this parameter is not recommended to be set higher than PA6. If the devices that are to be bound are across a room from each other, higher PA can be used with agreement to the names and definitions in the PA_PHY_TBL[] table.

4.2.3.2 *PGA User Module*

The PGA user module does not serve the traditional purpose in this design. It exists to facilitate the rout ability of the analog inputs to the ADC analog block. For the same purpose, the PGA gain is selected as '1'. This user module can be exercised to amplify low-power signals or to also attenuate high power signals.

Figure 4-12. PGA User Module



Name	PGA
User Module	PGA
Version	3.2
Gain	1.000
Input	AnalogColumn_InputSelect_1
Reference	VSS
AnalogBus	Disable

The input to the PGA is the output of the Analog Column Select Mux. This mux is used to switch between P0.0 and P0.1 inputs (or) the odd and even Port0 inputs.

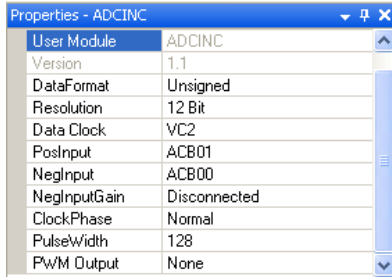
- **Gain.** 1 - No signal attenuation or gain through the PGA block. The PGA block exists only to ease routability of the input signals P0.0 and P0.1 to the ADC block.
- **Input.** AnalogColumn_InputSelect_1 - The P0.0 input is routed to the input of the PGA block by default. When P0.1 is to be read, register ABF_CR0 is configured to route the odd column in Port0 to the input of the PGA block.
- **Reference.** This is the reference to the PGA block and must be chosen as Vss.

4.2.3.3 ADCINC User Module

The ADCINC is a differential or single input ADC that returns a 6 to 14-bit result. The maximum DataClock frequency is 8 MHz, but 2 MHz is the maximum frequency recommended for improved linearity. This ADC may be placed only once, due to its implementation which uses the hardware decimator rather than a digital block.

The incremental ADC is used to calculate counts proportional to the voltages at P0.0 and P0.1. These values are used in calculations to measure the ambient temperature.

Figure 4-13. ADCINC User Module



The incremental ADC is placed in ASD11 block. This is done to ease routability of the two input signals from P0.0 and P0.1 on the PSoC device. The input to the PGA is the output of the Analog Column Select Mux. This mux is used to switch between P0.0 and P0.1 inputs.

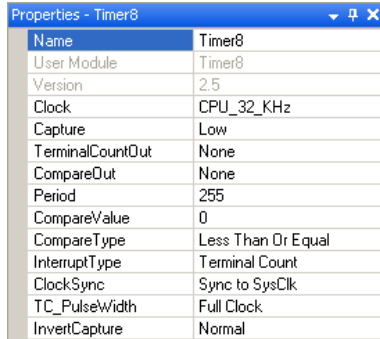
- **Resolution.** 12-bits: Sufficient for temperature sensor application.
- **Data format.** Unsigned: Calculates max counts (4096) when the input equals the reference voltage and min count.
- **Data Clock.** 2 MHz: 2 MHz is used for sample rate and good linearity.
- **PosInput.** ACB01: Input to the ADC block is actually the output from the PGA. The PGA block is placed in analog block ACB01.
- **NegInput.** ACB00: This input can be anything. The differential signal is not measured in this case.
- **NegInputGain.** Disconnected: This disconnects the negative input to the device.

4.2.3.4 Timer8 User Module

This user module implements an 8-bit timer that is clocked by a divider of SysClk. This is used to calibrate the sleep timer that is clocked by the 32 kHz system oscillator.

The Sleep Timer included as a part of the CYFISNP user module is clocked by the 32 kHz oscillator. The 32 kHz oscillator provides very low accuracy in the order of 18-50% worst case. The Timer8 User Module is clocked by the 32 kHz oscillator and is used to calibrate the Sleep Timer (counts) against CPU execution time (1 ms) for better accuracy. The 32 kHz oscillator or the sleep timer interval is not changed here, but the sleep counts or number of sleep timer expirations.

Figure 4-14. Timer8 User Module



Properties - Timer8	
Name	Timer8
User Module	Timer8
Version	2.5
Clock	CPU_32_KHz
Capture	Low
TerminalCountOut	None
CompareOut	None
Period	255
CompareValue	0
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
TC_PulseWidth	Full Clock
InvertCapture	Normal

- #### ■ Capture

This parameter is selected from one of the available sources. A rising edge on this input causes the Count register to be transferred to the Compare register. The software capture mechanism cannot operate correctly if this parameter is set to a value of one or is held high externally. This is set to LOW.

- #### ■ TerminalCountOut

The terminal count output is an auxiliary Counter output. This parameter allows it to be disabled or connected to any of the row output buses. This parameter is also disabled.

- #### ■ CompareOut

The compare output may be disabled (without interfering with interrupt operations) or connected to any of the row output buses. It is always available as an input to the next higher digital PSoC block and to the analog column clock selection multiplexers, regardless of the setting of this parameter. This parameter appears only for members of the CY8C29/27/24/22/21xxx and CY8CLED04/08/16 families of PSoC devices.

- #### ■ Period

This parameter sets the period of the timer. For an 8-bit timer allowed values are between 0 and 255. This value is loaded into the Period register. The period is automatically reloaded when the counter reaches zero or the timer is enabled from the disabled state. This value may be modified using the API, and is set to 255.

- #### ■ CompareValue

This parameter sets the count point in the timer period when a compare event is triggered. This parameter is not used in this configuration.

- #### ■ CompareType

This parameter sets the compare function type "less than" or "less than or equal". This parameter is not used in this configuration.

- **InterruptType**

This parameter specifies whether the terminal count event or the compare event triggers the interrupt. The interrupt is enabled using the API. The interrupt is not used in this configuration.

- **ClockSync**

Sync to SysClk option to synchronize to SysClk. This option is selected because there could be propagation delays and skew when routing the main system clock through the clock dividers.

- **TX8 User Module**

This module is used for serial communication with host. This module is present for debugging purposes.

Port 0.2 is chosen to output the TX8 data and clock selected is VC3 at approximately 920 kHz clock (VC3 clock = SysClk/1 & VC3 divider = 26).

4.2.4 Firmware Model

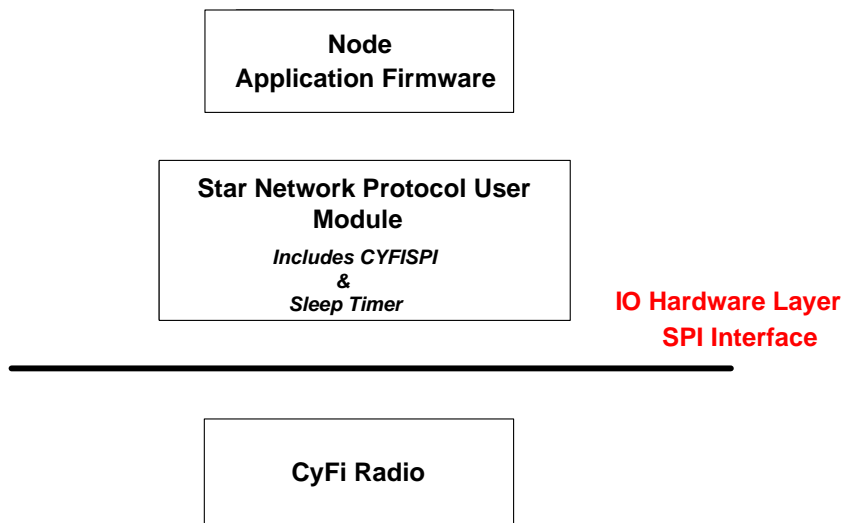
Header file main.h contains the following components:

- Macro defines
- Global variables
- Thermistor Table
- Function Prototypes

4.2.5 Architecture

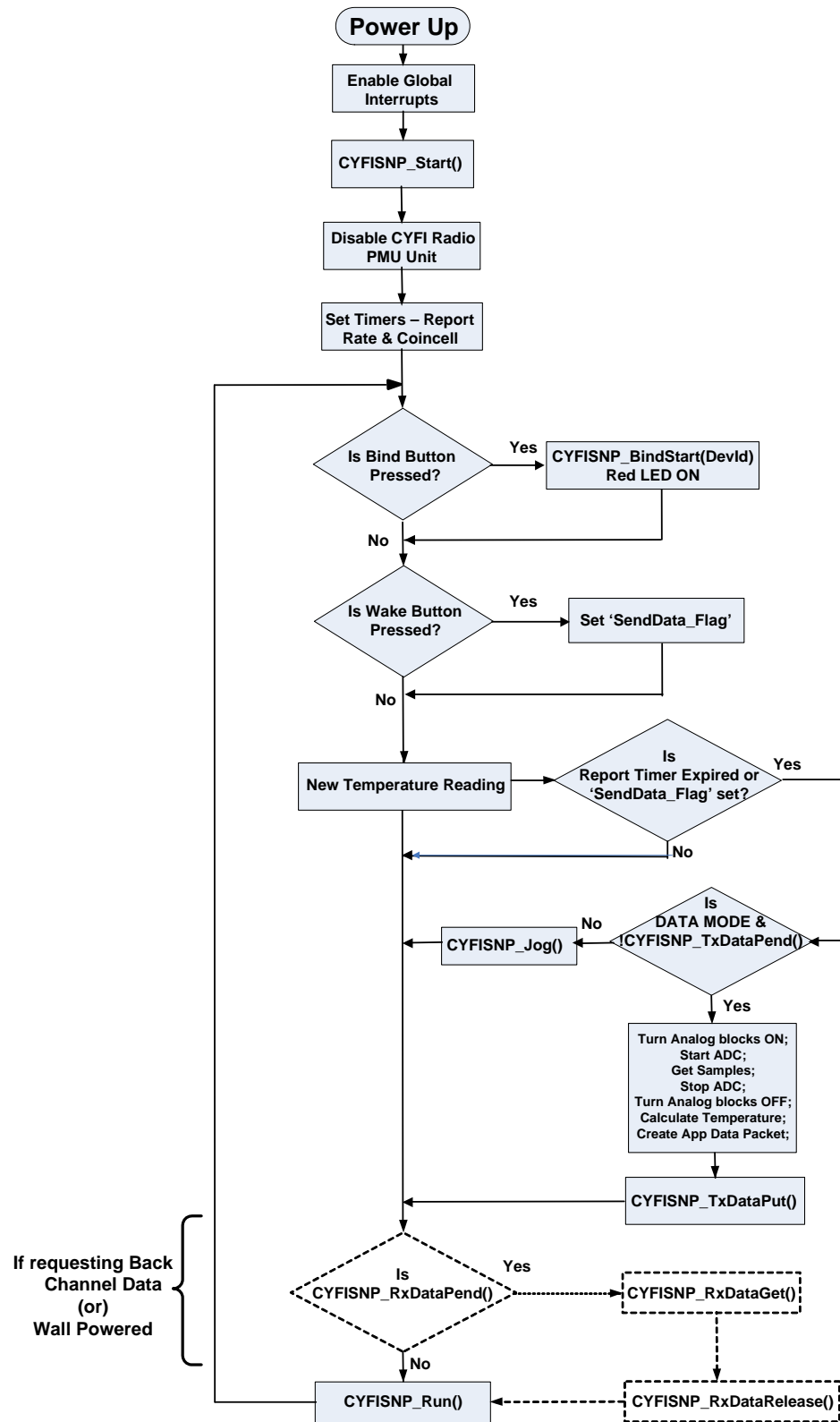
4.2.5.1 Firmware Layering

Figure 4-15. Firmware Layering



The main application firmware architecture is based on the functional block diagram. [Figure 4-16 on page 53](#) shows the flowchart.

Figure 4-16. Functional Block Diagram



4.2.5.2 *Main.c*

Main.c houses all the firmware for the Temperature Sensor application.

The firmware can be broken into the following parts:

1. Power up initialization and setup
2. Temperature Readings
3. Main loop of the application with general tasks
4. Calibration routine for the timer

Low-power considerations are done where applicable in the temperature sensor code. Debug functionality is achieved by using a serial TX8 interface and is included in the firmware code where necessary. The serial port pin is inverted and is fed to Port 0.2.

4.2.5.3 *Power Up Initialization and Setup*

This procedure requires the following steps:

1. Turn red and green LEDs ON for visual debug.
2. Enable global interrupts by calling the 'M8C_EnableGInt' macro. This step is mandatory before the CYFISNP protocol is invoked.
3. Turn TX8 ON for debug purposes.
4. Turn the CYFISNP Protocol ON. **CYFISNP_Start()**. The API starts and initializes the radio. Ensure global interrupts are enabled.
5. Disable the CyFi Radio power management unit. Savings of around 32 uA are observed.
6. The sleep timer is built into the CYFISNP protocol. This protocol provides the CYFISNP_TimeSet API for developers to start and enable timers. These timers are dependent on sleep timer counts.
 - Report Time Interval: The Report time is configured by the 'NewTime' variable in the main.h file. This is set to a default of 0x05, or 5 seconds. This can be configured for the desired report time in seconds.

4.2.5.4 *Temperature Readings*

In this section, the hardware block diagram shown in [Figure 5-5 on page 88](#) is implemented in firmware.

1. The temperature readings are done only if there is a 'Wake button' press or if the report timer expires. Timer Calibration routine is also called here. Timer Calibration is explained in detail in a later section.

```
if ((!CYFISNP_TimeExpired(&reportTimer))&&(!SendData_Flag))
{
    return;
}
```

2. In addition, the application must ensure that it is residing in the 'DATA MODE'. Confirm that the Tx data buffer is empty before starting with the temperature measurement process. If this condition is not satisfied, CYFISNP_Jog API is called to rouse the radio of any connect or ping mode timeouts

```
CYFISNP_eProtState == CYFISNP_DATA_MODE
&& CYFISNP_TxDataPend() == FALSE
```

The temperature reading takes place only if this condition is true.

The following steps account for low-power considerations for temperature measurement.

1. Since this design is based on low-power implementation, the thermistor resistor network is not powered continuously. The port pin P2.2 is driven a 'high' when the temperature needs to be calculated. This powers the thermistor/resistor network. This is done using the `Vtherm_div_ON` define.

```
#define Vtherm_div_OFF (Vtherm_div_Data_ADDR = Port2_DataShade_OFF)
#define Vtherm_div_ON (Vtherm_div_Data_ADDR = Port2_DataShade_ON)
```

2. Turn the analog power ON. This essentially turns on the analog SC and CT blocks. This is effected by configuring register `ARF_CR`.

```
#define Analog_ON      (ARF_CR = 0x16)
#define Analog_OFF    (ARF_CR = 0x10)
```

3. Turn ON the ADC and PGA at medium power.

4. Configure the input mux to connect P0.0 (as shown in [Figure 4-16 on page 53](#)) to the input of the PGA. The output of the PGA is fed to the incremental ADC. Port0, 'even' inputs are connected by clearing bit[0] of the `ABF_CR0` register. Port0, 'odd' inputs are connected by setting bit[1] of the `ABF_CR0` register.

```
#define ReadVTEMPInput      (ABF_CR0 = 0x00)
#define ReadVTEMP_EXCInput  (ABF_CR0 = 0x80)
```

Take an ADC reading with P0.0 as the input and store it in `IVtemp`.

Then change the analog mux setting to connect P0.1 to the PGA input and take another reading and store the value in `iVexc`.

5. Turn ADC and PGA OFF. Turn Analog OFF and disable the strong drive to the thermistor at P2.2.
6. Calculate Temperature based on the two values `IVtemp` and `iVexc`.

```
iTemp = CalculateThermData(iVexc, lVtemp);
```

7. Configure Tx Buffer payload bytes.

```
loadTxData();
```

8. Load the Tx data to be transmitted.

```
CYFISNP_TxDataPut (&txApiPkt);
```

4.2.5.5 *Timer Calibration Routine*

The sleep timer is a part of the `CYFISNP` user module and is used to time various events in the protocol. The sleep timer is clocked by the 32 kHz internal low-power oscillator. The 32 kHz LPO is not very accurate and its accuracy can change as much as 50%. However, you can improve this accuracy by calibrating the sleep timer (and indirectly the 32 kHz oscillator) against the internal main oscillator (IMO) running at 24 MHz. See AN14278 for an alternate implementation of sleep timer calibration.

This calibration routine can be called every Report Timer expiration or as required.

The steps for calibration are:

1. Create a 1 ms delay loop based on CPU cycles.
2. Create an 8-bit timer clocked by the 32 kHz source.
3. Disable global interrupts.
4. Start timer after loading maximum period value (255).
5. Call the 1 ms delay loop.
6. Read the timer value and store it to a variable and stop timer counts.
7. Reenable global interrupts.

The number of 32 kHz counts for 1 ms must be 32. The timer counts value is compared to 32.

If there is no count difference, the same time period is maintained. If there is a difference, the counts are changed for the sleep timer. The sleep timer expires every 16 ms. So, for 5 seconds the sleep timer must expire (5000/16) times. This is the value that is calibrated depending on the difference in counts of the 8-bit timer.

4.2.5.6 Temperature Conversion Algorithm

One of the tasks required of the Temperature sensor node is calculation of absolute temperature from the ADC readings.

A ratio metric value is obtained from the two ADC readings:

$$\text{Ratio} = (V_{\text{temp}} * 10000) / V_{\text{exc}}$$

This metric ensures that there is no voltage dependency on the ADC readings and the temperature calculations.

A temperature table is created that depends on Steinhart's coefficient for the thermistor used on the FTRF Expansion Board. This table has temperature readings corresponding to voltages (ADC readings) at Vtemp(P0.0) and Vexc(P0.1) inputs. The table exists in main.h:

```
const int ThermTable[2][COUNT_VALUES] =
{
    {2301, 2505, 2725, 2960, 3211, 3477, 3757, 4051, 4358, 4675, 5000,
5331, 5664, 5998, 6328, 6652, 6967, 7269, 7557, 7829, 8083, 8317, 8463},
    {5500, 5200, 4900, 4600, 4300, 4000, 3700, 3400, 3100, 2800, 2500,
2200, 1900, 1600, 1300, 1000, 700, 400, 100, -200, -500, -800, -1000}
};
```

This table has values starting from temperature 55.00 to 10.00 in 3C increments.

After the ratio is calculated, check through the tables to find the two temperatures between where the reading is. If it exceeds 55°C or 10°C temperature, fix those as the upper and lower limits and exit the temperature conversion algorithm.

```
if ((int)Vtemp < ThermTable[0][0])
{
    // The voltage ratio is too low, so the temperature is greater than what
    can be measured
    Vtemp = MAX;
}
else if((int)Vtemp > ThermTable[0][COUNT_VALUES-1])
{
    // The voltage ratio is too high, so the temperature is less than what
    can be measured.
    Vtemp = MIN;
}
else
{
    // Scan through the voltage ratio values in the piecewise linear curve
    fit data to find
    // the appropriate line to interpolate
    for(bPointIndex = 0; bPointIndex < (COUNT_VALUES-2); bPointIndex++)
    {
        if (Vtemp < ThermTable[0][bPointIndex+1]) break;
    }
}
```

At the end of this step, there are two temperature values between where the ratio resides. After retrieving these two values, interpolate to find the actual temperature.


```
ivaluel = ThermTable[0][bPointIndex];
ivalue2 = ThermTable[0][bPointIndex + 1];

    // Retrieve the temperatures for interpolation
itemp1 = ThermTable[1][bPointIndex];
itemp2 = ThermTable[1][bPointIndex + 1];

//Interpolate to find the temperature in hundredths of a deg C
Vtemp = (((long) ivalue2 - Vtemp) * (itemp1 - itemp2)) / (ivalue2 -
ivaluel) + itemp2;

    // First, get the temperature value as an integer
ivaluel = Vtemp;
```

The accuracy of the temperature reading obtained is two decimal places. The accuracy requirement is one decimal place. As a result, determine if the 2nd decimal is >5 and if yes, increment the 1st decimal space value by 'one'.

The integer value is then returned back by:

```
CalculateThermData(int Vexc, long Vtemp)
```

4.2.5.7 *Main Application loop tasks*

1. Check for Bind button press. Scan the bind button every cycle through the main application loop to detect a button bind request from the user.

Implemented by the API **checkBindButton()**

2. Check for Test button press. Scan the wake button every cycle through the main application loop to detect a wake event by the user.

Implemented by the API **CheckWakeButton()**

3. Call the ReadNewTemperature() function. This is explained in detail in the section [Temperature Readings on page 54](#).
4. Call the CYFISNP_Run() API. This exercises the protocol and must be called for Bind, connect, ping, and data modes. It is very important for maintaining the RF link between the nodes and the hub.
5. Call M8C_Sleep macro to put the microcontroller in a sleep state. This conserves power. The micro comes out of sleep when there is a sleep timer interrupt.

4.2.5.8 *Memory Management:*

The PSoC 27K parts have 16K of Flash memory. The Flash is organized into 2 banks with 64-byte blocks making up 8K in each bank. The Node firmware is stored and executed out of flash memory. The user need not be concerned with program store page boundaries, because the hardware automatically increments the 16-bit program counter on every instruction making the block boundaries invisible to user code. Upon reset, the program counter is reset to 0x00.

The PSoC 27K parts have 256 bytes of RAM. All memory allocation in the Node firmware uses the same 256 bytes of RAM. If the last byte in the stack is at address 0xFF the Stack Pointer will wrap RAM address 0x00. It is the firmware developer's responsibility to ensure the stack does not overlap with user-defined variables in RAM.

4.3 Wireless I2C Bridge for RF Expansion Card

This section discusses the design goals, architecture, firmware source code modules, and configuration options for the FTRF I2C Bridge Node.

4.3.1 Design Features

The CY3271 FTRF Kit uses a PSoC CY8C27443 on the RF Expansion Board. This is the application processor that controls the CyFi Radio and the other components on the board. Since the FTRF expansion card has its own PSoC, you can also operate it by removing it from the battery-pack and inserting it into your target hardware or other development platform. [Figure 2-9 on page 19](#) is an illustration of where to locate the application files.

This section describes the firmware implementation of using the RF Expansion Board as an I2C Bridge to retrieve data packets from an external board through I2C, and transmit the data back to the central hub. As a result, this implementation can be used to 'cut the cord' (within limits) in a wired monitoring application.

4.3.2 Firmware Architecture

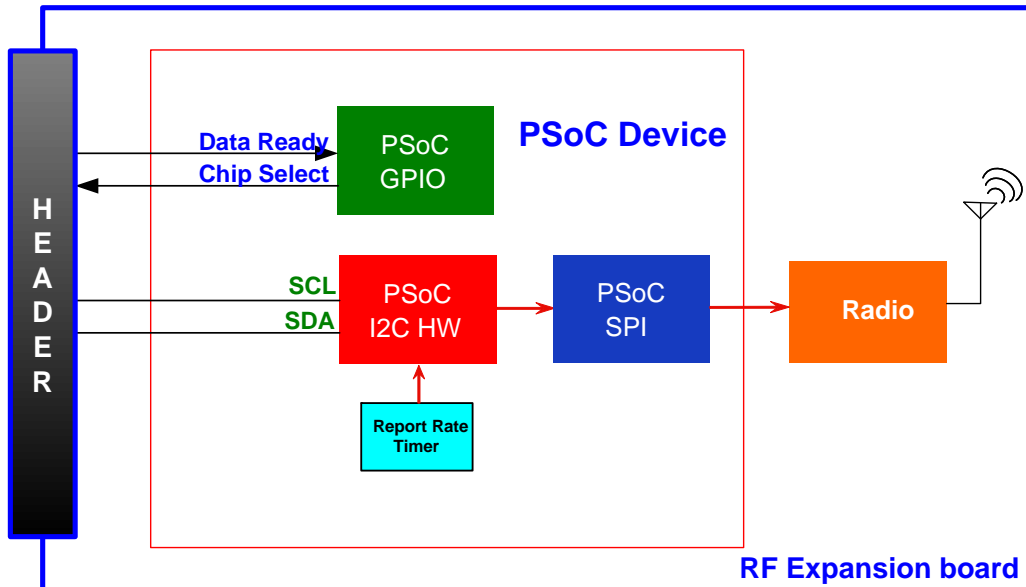
4.3.2.1 ROM/RAM Usage

Table 4-3. ROM/RAM Usage

	Total ROM (Bytes)	Total RAM (Bytes)
Node functionality with TX8 Debug enabled	9499	140

4.3.2.2 Functional Block Diagram of the Temperature Sensor Node

Figure 4-17. Functional block diagram for I2C Bridge Node Application



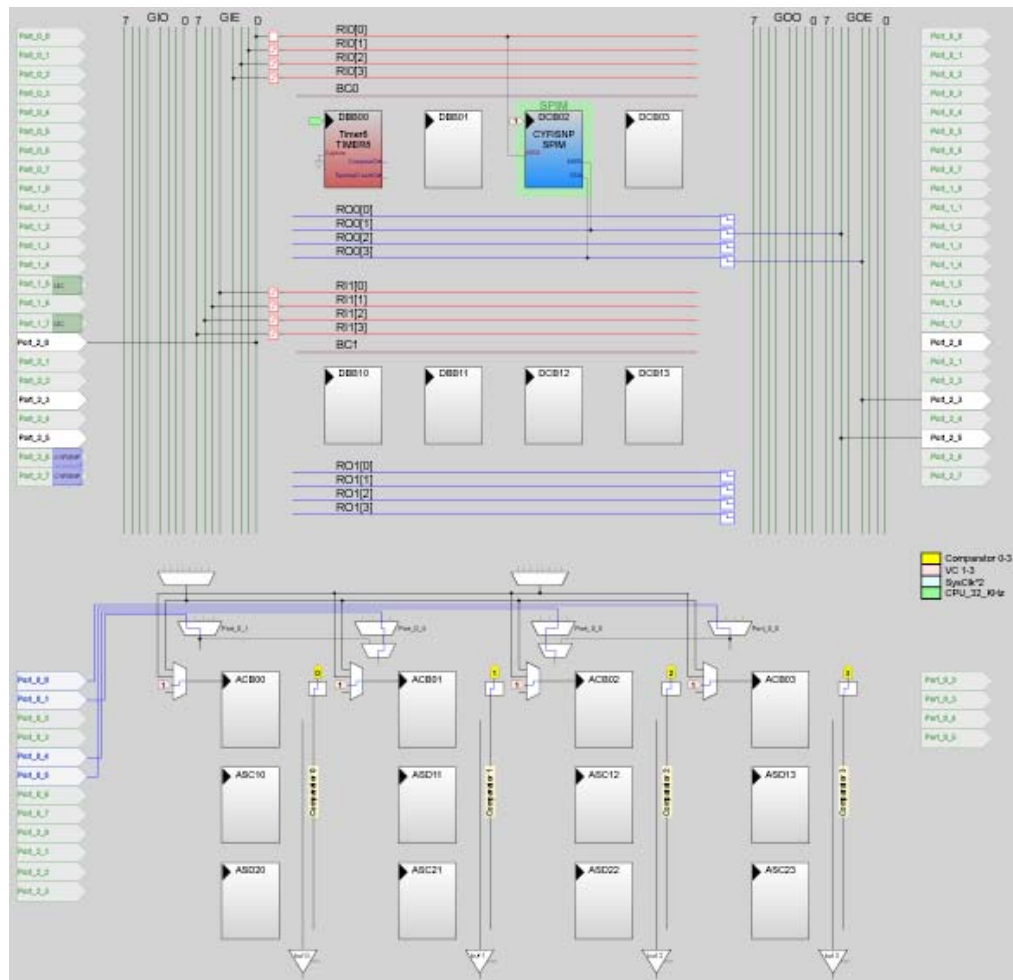
The FTRF I2C Bridge design consists of the following modules:

1. I2C Master
2. CYFISNP
3. Timers for Report Rate

4.3.2.3 Device Configuration for CY8C27443 in the RF Expansion Board

The CY8C27443 is configured using the Device Editor in PSoC Designer 5.0. The node uses the CYFISNP, Timer8, I2C Master - HW and TX8 user modules. The following section discusses the configuration of global resources and user module parameters for every user modules used.

Figure 4-18. Chip Layout View Showing User Modules Placement and Routing

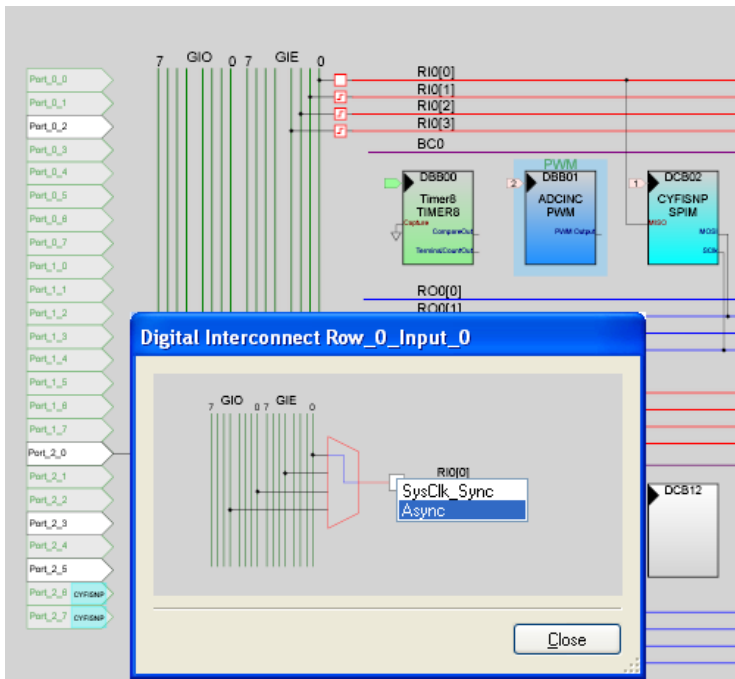


4.3.2.4 Global Configuration

This section describes the global resources available for configuring the CY8C27443, which runs the Node application. Modify these values carefully because they may affect the user modules discussed in later sections.

- **Power Setting.** 3.3V / 24 MHz operation. Supply voltage at 24 MHz.
- **CPU Clock.** This parameter is set to SysClk/2. To run the CPU at 12 MHz, SysClk/N must be set to '2'.
- **VC1.** This clock is set at 6 MHz. As a result, VC1= SysClk/N is set as 4. The SPI block in the CYFISNP User Module is clocked with VC1. When the SPI clock in PSoC is set at 2 MHz or higher, the MOSI input must be selected as 'Asynchronous' as shown in [Figure 4-19 on page 60](#).

Figure 4-19. Global Configuration



- **VC2.** VC2=VC1/N. This is not used in the configuration settings.
- **VC3 Source.** SysClk
- **VC3 Divider.** 26 - Clocks the TX8 block
- **SysClk Source.** Internal (Internal Main Oscillator)
- **SysClk*2 Disable.** Yes - Power savings
- **Analog Power.** All Off - Analog circuitry is powered using firmware during readings
- **Ref Mux.** (Vdd/2)+/- (Vdd/2)
- **AGndBypass.** Disable
- **OpAmp Bias.** Low
- **A_Buff_Power.** Low
- **Switch Mode Pump.** OFF, not used in design. Saves power when OFF.
- **Trip Voltage [LVD].** 2.92 V
- **LVD Throttleback.** Disable
- **Watchdog Enable.** Enable

4.3.3 User Modules used in the I2C Bridge PSoC Project

4.3.3.1 CYFISNP User Module

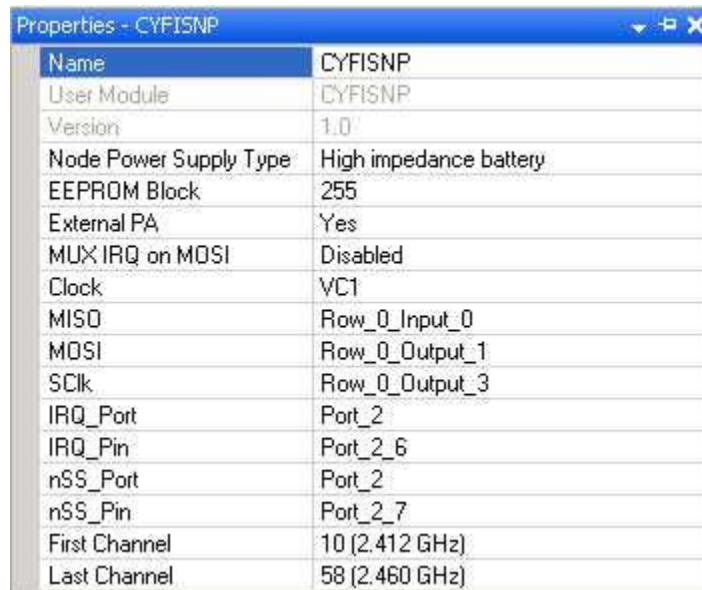
The CYFISNP user module is at the core of the node application firmware. This user module controls all wireless communication with the hub. In addition, this user module also implements the entire Star network wireless protocol, its modes, and also the low level communication between the radio and the MCU. API calls to the CYFISNP user module affect control.

The CYFISNP consists of the following components:

- A higher level of medium access protocol firmware that takes care of the network level details, like node binding, finding a quiet channel, data communication with nodes, and others.
- A lower level radio driver firmware that lets the microcontroller (CY8C27443) control the CYRF6936 radio over SPI.
- The user module contains an E2PROM component to write the parameters of the Hub to the node's flash.
- The user module also consists of the Sleep timer configured at 64Hz. This is required for timing the various events and modes in the protocol.

The SPI lines of MOSI, MISO, SCLK, and the IRQ and nSS lines are correctly configured according to the PSoC drive configurations.

Figure 4-20. CYFISNP User Module



Name	Value
Name	CYFISNP
User Module	CYFISNP
Version	1.0
Node Power Supply Type	High impedance battery
EEPROM Block	255
External PA	Yes
MUX IRQ on MOSI	Disabled
Clock	VC1
MISO	Row_0_Input_0
MOSI	Row_0_Output_1
SCLK	Row_0_Output_3
IRQ_Port	Port_2
IRQ_Pin	Port_2_6
nSS_Port	Port_2
nSS_Pin	Port_2_7
First Channel	10 (2.412 GHz)
Last Channel	58 (2.460 GHz)

Node Power Supply

This node is powered by Alkaline cells that have low internal resistance. 'Low Impedance Battery' is chosen.

EEPROM Block

This parameter defines the EEPROM block number used to store network parameters. The last block in the flash is chosen to write or store the parameters. This block must be marked as unprotected in the flashsecurity.txt file.

External PA

This parameter controls the operation of the external Power Amplifier. Option 'NO' must be selected because there is no an external PA.

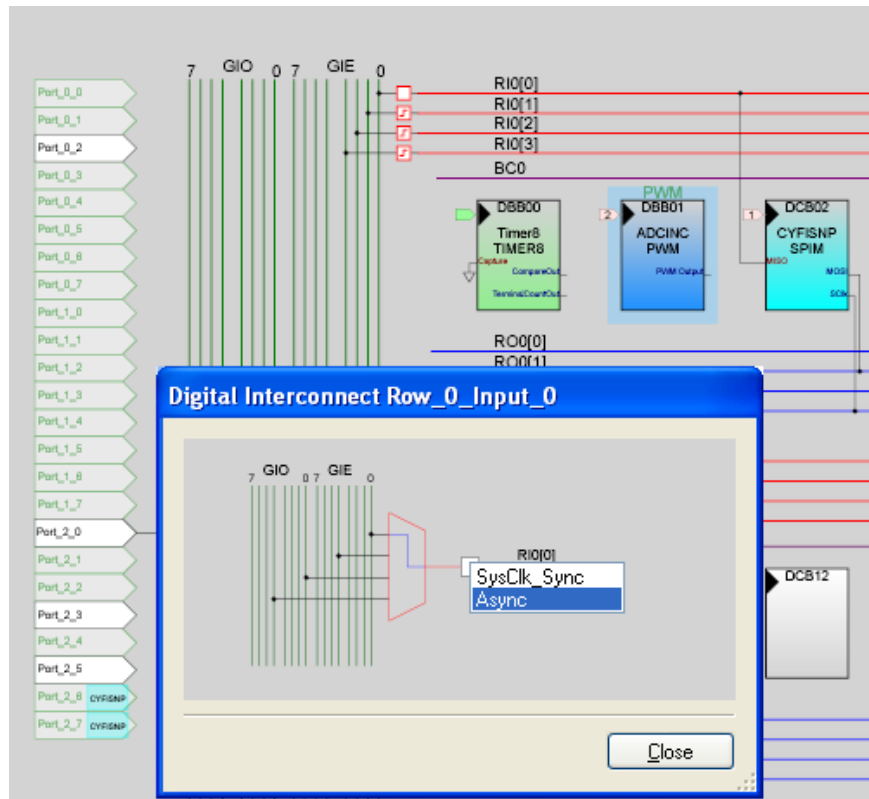
MUX IRQ on MOSI

The CYRF793x device IRQ pin may be multiplexed onto the MOSI pin. The temperature sensor project does not use the multiplexing option.

Clock

Selects the SPIM Block Clocking source. The clock rate is set to 6 MHz and the 'Async' option is chosen for the line going to the SPI clock.

Figure 4-21. Clock



MISO

The Master-In-Slave-Out (MISO) input signal is routed to Row_0_Input_0 net.

MOSI

The Master-Out-Slave-In (MOSI) output signal is routed to Row_0_Output_1 net.

SClk

This output clock is generated by the SPI Master, and is routed to Row_0_Output_3 net.

IRQ_Port, IRQ_Pin

Selects the IRQ pin. Choose Pin P2.6 for the IRQ pin, because muxing with MOSI is not opted.

nSS_Port, nSS_Pin

Selects the SlaveSelect pin. Pin P2.7 is selected to route the nSS signal.

First Channel

This parameter controls the low limit of the channel range of CYFISNP protocol. This avoids spilling over to other channels, especially when using external PA. This is an important consideration when seeking FCC or similar RF certification.

When a PA option is selected in the user module configuration, the First Channel defaults to Channel 10 (changeable by the developer). If the Node application does not use the external PA, but the hub does, then the node must use the same first and last channels to comply with the hub's restricted channel subset.

Last Channel

This parameter controls the high limit of channel range of CYFISNP protocol. This avoids spilling over to other channels, especially when using external PA. This is an important consideration when seeking FCC or similar RF certification.

When a PA option is selected in the user module configuration, the Last Channel defaults to Channel 58 (changeable by the developer). If the Node application does not use the external PA, but the hub does, then the node must use the same first and last channels to comply with the hub's restricted channel subset.

Config.h

The CYFISNP User Module generates a file named CYFISNP_Config.h that contains tables and variables that you can configure for your project. Most of the code in the generated header and source files in your project is regenerated each time your project is built. As a result, any changes that you make to this code are lost. The exceptions to this rule are sections of code contained between @PSoC_UserCode_name@ and @PSoC_UserCode_END@.

You can change the following items in CYFISNP_Config.h:

- **@PSoC_UserCode_PaPhyTblInternal_00**

This table allows you to customize the eight possible RF power steps for the Dynamic PA feature, when the Internal PA is used. This list must be monotonic for proper operation. This table starts at PA4 since there are no significant current savings below PA4. The table stops with PA6 because the maximum internal PA is PA6.

- **@PSoC_UserCode_PaPhyTblExternal_00**

This table allows you to customize the eight possible RF power steps for the Dynamic PA feature. This list must be monotonic for proper operation. This table is used only if the External PA option is set to Yes. This table provides configuration for the following three options:

FTRF Coin-cell Application

Not applicable for this application because alkaline-cell or low impedance battery is chosen.

Default Option with External PA up to +10 dBm

This is the default setting or table (Table 2) when external PA is chosen in the User Module parameters. The external PA is configured such that the RF output power can only go up as high as +10 dBm with this table. This limitation is due to the RF power restrictions imposed by Europe and Japan. The power can be increased to +20 dBm only in the United States and Canada.

Option with External PA up to +20 dBm

RF Applications in most countries other than Europe and Japan (unless there are regional or other restrictions) can go up to +20 dBm. The default table (Table 2) must be commented out and the +20 dBm table (Table 3) must be uncommented for the +20 dBm RF power to be output by the external PA.

■ **PA_LEVEL_BIND**

This section helps you to set the PA_LEVEL_BIND variable. This variable allows you to change the PA level used during binding. Changing the PA level influences the binding distance. The default value of PA4 is good for short distances (within a few feet). If the devices that are to be bound are across a room from each other, higher PA can be used with agreement to the names and definitions in the PA_PHY_TBL[] table.

4.3.3.2 *I2C Master Hardware User Module*

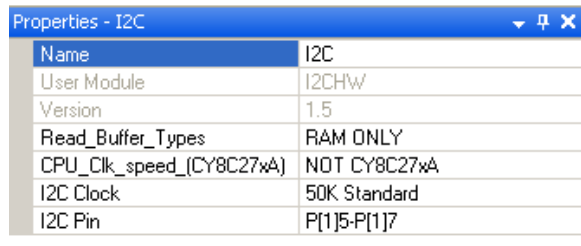
This user module provides support for the I2C hardware resource on the CY8C27443 PSoC device. It is capable of transferring data at 50/100/400 kbps when the CPU clock is configured to run at 12 MHz. There are two different selections for SDA and SCL providing direct access to the hardware resource.

The I2C resource supports data transfer at a byte by byte level. At the end of each address or data transmission or reception, the status is reported or a dedicated interrupt may be triggered. Status reports and interrupt generation depend on the direction of data transfer and the condition of the I2C bus as detected by the hardware. Interrupts may be configured to occur on byte-complete, bus-error detection and arbitration loss.

The I2C master User Module supports polled data transfers on a byte by byte basis, without using the interrupt or supplied ISR.

This module is responsible for transferring data between the RF Expansion Board and the add-on board connected externally.

Figure 4-22. I2C Master Hardware User Module



Properties - I2C	
Name	I2C
User Module	I2CHW
Version	1.5
Read_Buffer_Types	RAM ONLY
CPU_Clk_speed_(CY8C27xA)	NOT CY8C27xA
I2C Clock	50K Standard
I2C Pin	P[1]5-P[1]7

I2C_Clock

Specifies the desired clock speed at which to run the I2C interface. There are three possible clock rates available: 50K Standard, 100K Standard, and 400K Fast.

For the Node I2C Bridge, 50K Standard I2C clock is selected.

I2C_Pin

Selects the pins from Port 1 to be used for I2C signals. You do not have to select the proper drive mode for these pins, because PSoC Designer does this automatically. P[1]5 and P[1]7 are selected. On the RF Expansion Board, these pins are routed from the PSoC to the female header.

CPU_Clk_speed_(CY8C27xA)

The CY8C27xA refers to older CY27xxx devices that are no longer recommended for new designs. CY8C27xxx silicon revision A is not used in our design. The NOT CY8C27xA option is chosen.

Read_Buffer_Types

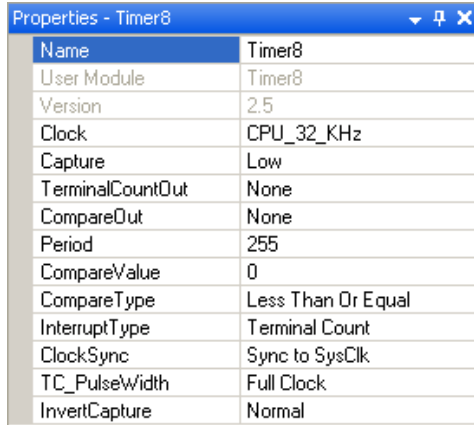
Selects what types of buffers are supported for data reads. Two selections are available: RAM ONLY or RAM OR FLASH. In the Node I2C Bridge design, only RAM is used for data reads and writes RAM ONLY.

4.3.3.3 *Timer8 User Module*

This user module implements an 8-bit timer that is clocked by a divider of SysClk. This is used to calibrate the sleep timer that is clocked by the 32 kHz system oscillator.

The Sleep Timer included as a part of the CYFISNP user module is clocked by the 32KHz oscillator. The 32 kHz oscillator provides very low accuracy in the order of 18-50% worst case. The Timer8 User Module is clocked by the 32 kHz oscillator and is used to calibrate the Sleep Timer (counts) against CPU execution time (1 ms) for better accuracy. The 32 kHz oscillator or the sleep timer interval is not changed here, but the sleep counts or number of sleep timer expirations.

Figure 4-23. Timer8 User Module



Properties - Timer8	
Name	Timer8
User Module	Timer8
Version	2.5
Clock	CPU_32_KHz
Capture	Low
TerminalCountOut	None
CompareOut	None
Period	255
CompareValue	0
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
TC_PulseWidth	Full Clock
InvertCapture	Normal

Capture

This parameter is selected from one of the available sources. A rising edge on this input causes the Count register to be transferred to the Compare register. The software capture mechanism cannot operate correctly if this parameter is set to a value of one or is held high externally. This is set to LOW.

TerminalCountOut

The terminal count output is an auxiliary Counter output. This parameter allows it to be disabled or connected to any of the row output buses. This parameter is also disabled.

CompareOut

The compare output may be disabled (without interfering with interrupt operations) or connected to any of the row output buses. It is always available as an input to the next higher digital PSoC block and to the analog column clock selection multiplexers, regardless of the setting of this parameter. This parameter appears only for members of the CY8C29/27/24/22/21xxx and CY8CLED04/08/16 families of PSoC devices.

Period

This parameter sets the period of the timer. For an 8-bit timer, the allowed values are between 0 and 255. This value is loaded into the Period register. The period is automatically reloaded when the counter reaches zero or the timer is enabled from the disabled state. This value may be modified using the API, and is set to 255.

CompareValue

This parameter sets the count point in the timer period when a compare event is triggered. This parameter is not used in this configuration.

CompareType

This parameter sets the compare function type "less than" or "less than or equal". This parameter is not used in this configuration.

InterruptType

This parameter specifies whether the terminal count event or the compare event triggers the interrupt. The interrupt is enabled using the API. The interrupt is not used in this configuration.

ClockSync

Sync to SysClk option to synchronize to SysClk. This option is chosen because there could be propagation delays and skew when routing the main system clock through the clock dividers.

4.3.3.4 *TX8 User Module*

Used for serial communication with host. This module is present for debugging purposes.

Port 0.2 is chosen to output the TX8 data and the clock selected is VC3 at approximately 920 kHz clock (VC3 clock = SysClk/1 & VC3 divider = 26). The purpose is to set the serial port to 115200,n,8,1 (which is about 920K/8)

4.3.4 **Firmware Model**

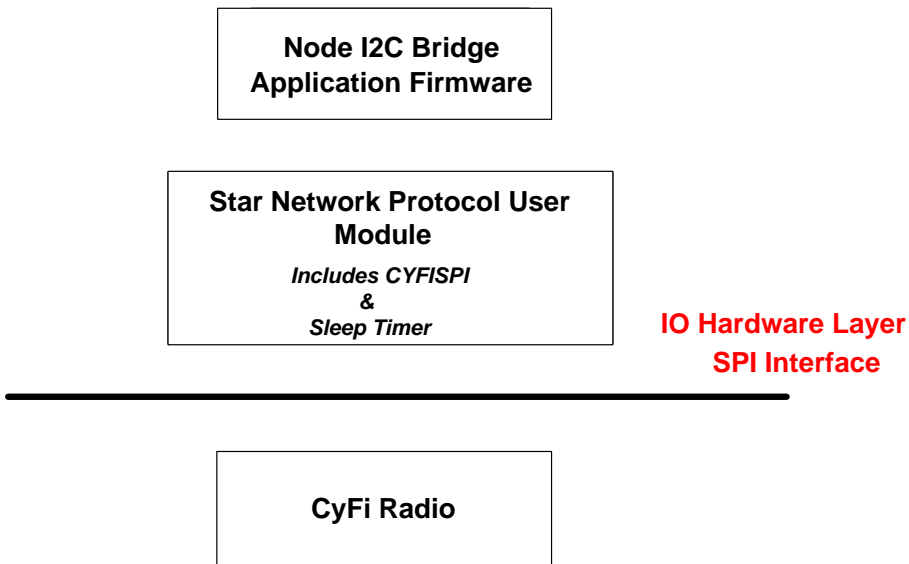
Header file main.h contains the following components:

- Macro Defines
- Global Variables
- Defines for I2C Maximum Payload, I2C Slave Address, and others
- Function Prototypes

4.3.5 **Architecture**

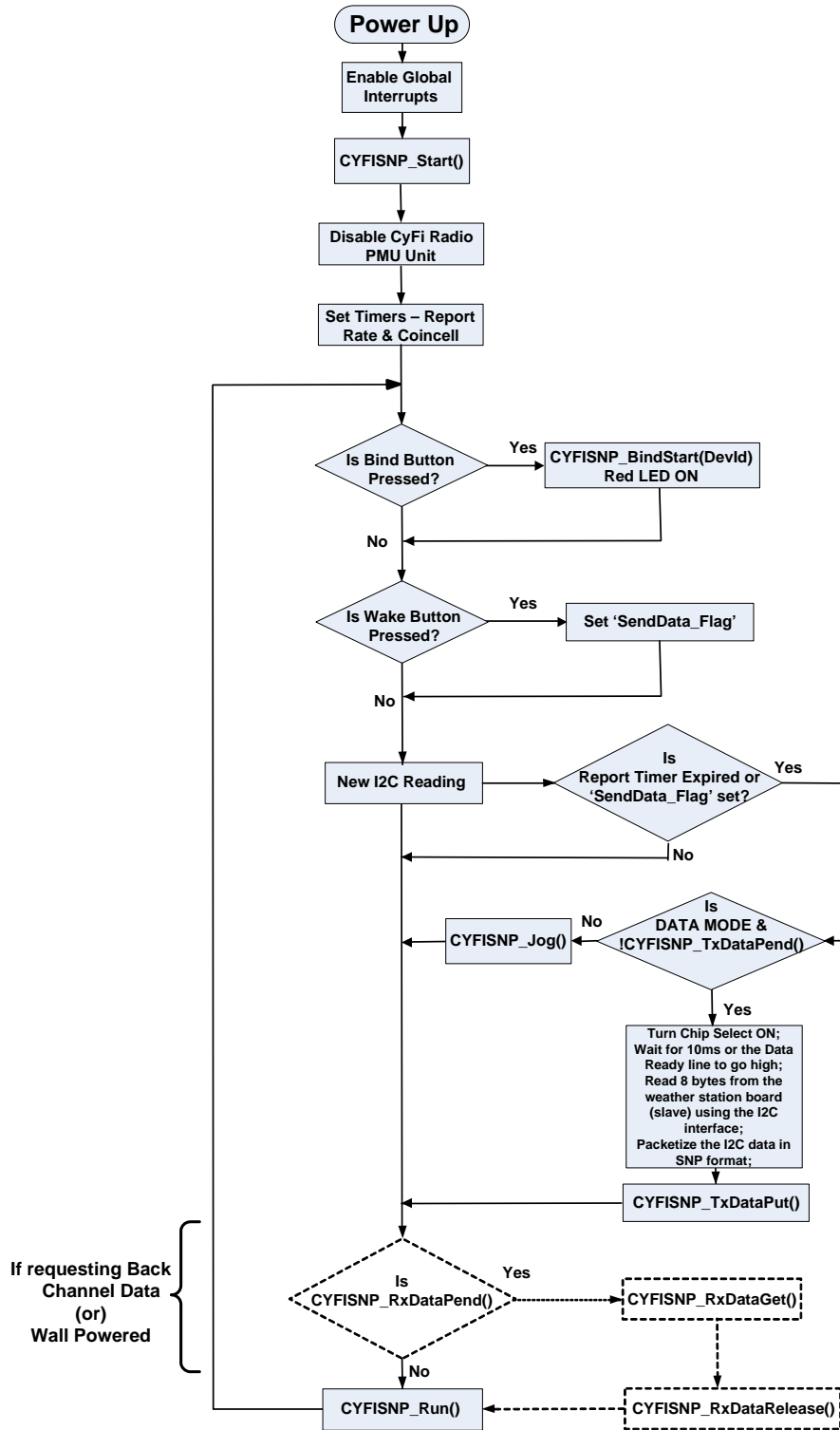
4.3.5.1 *Firmware Layering*

Figure 4-24. Firmware Layering



The main application firmware architecture is based on the functional block diagram. [Figure 4-25 on page 68](#) shows the flowchart.

Figure 4-25. Functional Block Diagram



4.3.5.2 *Main.c*

Main.c houses all the firmware for the I2C Bridge application.

The firmware can be broken into three main parts:

1. Power up initialization and setup
2. I2C Readings
3. Calibration routine for the timer
4. Main loop of the application with general tasks

Low-power considerations are done where applicable in the temperature sensor code. Debug functionality is achieved by using a serial TX8 interface and is included in the firmware code where necessary. The serial port pin is inverted and is connected to Port 0.2.

4.3.5.3 *Power Up Initialization and Setup*

This involves the following steps:

1. Turn the red and green LEDs ON for visual debug.
2. Enable global interrupts by calling the 'M8C_EnableGInt' macro. This step is mandatory before the CYFISNP protocol is invoked.
3. Turn TX8 ON for debug purposes.
4. Turn the CYFISNP Protocol ON. **CYFISNP_Start()**. This API starts and initializes the radio. Ensure global interrupts are enabled.
5. Disable the CyFi Radio power management unit. Savings of around 32 uA are observed.
6. The CYFISNP protocol has the sleep timer built into it. By calling the set of CYFISNP_TimeSet, timers can be set for events.
 - Report time interval: The Report time is configured by the 'NewTime' variable in the main.h file. This is set to a default of 0x05, or 5 seconds. This can be configured for the desired report time in seconds.

4.3.5.4 *SendNewTxMsg - I2C values*

In this section, the hardware block diagram ([Figure 5-5 on page 88](#)) is implemented in the firmware.

1. The I2C readings from the MultiFunction board are made only if there is a 'Wake button' press or if the report timer expires. Timer Calibration routine is also called here. The Timer Calibration is explained in detail in a later section.

```
if ((!CYFISNP_TimeExpired(&reportTimer)) && (!SendData_Flag))
{
    return;
}
```

2. Further, the application must ensure that it resides in the 'DATA MODE' and also ensure that the Tx data buffer is empty before starting with the temperature measurement process. If this condition is not satisfied, CYFISNP_Jog API is called to rouse the radio of any connect or ping mode timeouts

```
CYFISNP_eProtState == CYFISNP_DATA_MODE
&& CYFISNP_TxDataPend() == FALSE
```

I2C reading takes place only if this condition is true.

3. Call GetI2CData();
 - a. Enables the Chip Select
 - b. Starts a timer for 10 ms

```
CYFISNP_TimeSet(&dataDelay, DATA_READY_TIME);
```

- c. Waits for timer to expire or Data Ready pin to go high


```

while (CYFISNP_TimeExpired(&dataDelay) == 0)
{
    if (IS_DataReady_ON)
break;
}

```
 - d. Reads the I2C Data and stores it in rxBuffer


```
I2C_fReadBytes(SLAVE_ADDRESS, rxBuffer, 8, I2C_CompleteXfer);
```
 - e. Disables Chip Select
4. Configure Tx Buffer payload bytes - loadTxData();

Move I2C data from rxBuffer to SNP data packet.
 5. Load the Tx data to be transmitted.


```
CYFISNP_TxDataPut(&txApiPkt);
```

4.3.5.5 *Timer Calibration Routine*

The sleep timer is a part of the CYFISNP user module and is used to time various events in the protocol. The sleep timer is clocked by the 32 kHz internal low-power oscillator. The 32 kHz LPO is not very accurate and its accuracy can change as much as 50%. It is desired to calibrate the sleep timer (counts of 32 kHz oscillator) against the internal main oscillator (IMO) running at 24 MHz.

This calibration routine can be called every Report Timer expiration or as required.

The steps for calibration are:

1. Create a 1 ms delay loop based on CPU cycles.
2. Create an 8-bit timer clocked by the 32KHz source.
3. Disable global interrupts.
4. Start timer after loading max period value - 255.
5. Call the 1 ms delay loop.
6. Read the timer value and store it to a variable and stop timer counts.
7. Re-enable global interrupts.

The number of 32 kHz counts for 1 ms must be 32. The timer counts value is compared to 32.

If there is no count difference, the same time period is maintained. If there is a difference, the counts for the sleep timer are changed. Sleep timer expires every 16 ms. So, for 5 seconds, the sleep timer must expire (5000/16) times. This is the value that is calibrated depending on the difference in counts of the 8-bit timer.

4.3.5.6 *Main Application loop Tasks*

1. Check for Bind button press. Scan the bind button every cycle through the main application loop to detect a button bind request from the user.

Implemented by the API `checkBindButton()`

2. Check for Wake button press. Scan the wake button every cycle through the main application loop to detect a wake event by the user.

Implemented by the API `checkTestButton()`

3. Call the `SendNewTxMsg` function. This is explained in detail in the section [SendNewTxMsg - I2C values on page 69](#).
4. Call the `CYFISNP_Run()` API. This exercises the protocol and must be called for Bind, connect, ping, and data modes and is very important for maintaining the RF link between the nodes and the hub.
5. Call `M8C_Sleep` macro followed by 2 nops to put the microcontroller in a sleep state. This conserves power. The micro comes out of sleep when there is a sleep timer interrupt.

4.3.5.7 *Memory Management:*

The PSoC 27K parts have 16K of Flash memory. The Flash is organized into 2 banks with 64-byte blocks making up 8K in each bank. The Node firmware is stored and executed out of flash memory. The user need not be concerned with program store page boundaries, because the hardware automatically increments the 16-bit program counter on every instruction making the block boundaries invisible to user code. Upon reset, the program counter is reset to 0x00.

The PSoC 27K parts have 256 bytes of RAM. All memory allocation in the Node firmware uses the same 256 bytes of RAM. If the last byte in the stack is at address 0xFF, the Stack Pointer wraps RAM address 0x00. It is the firmware developer's responsibility to ensure the stack does not overlap with user-defined variables in RAM.

4.4 MultiFunction Expansion Card Light Sensor

4.4.1 MultiFunction Expansion Card CapSense Slider

The purpose of this project is to demonstrate the capacitive sensing capability of PSoC. You can change the color of the LED array by moving your finger across the CapSense slider. [Figure 2-9 on page 19](#) is an illustration of where to locate the project files. This project contains these driver elements:

- CSD Properties
- Slider
- LED

Figure 4-26. Driver Elements

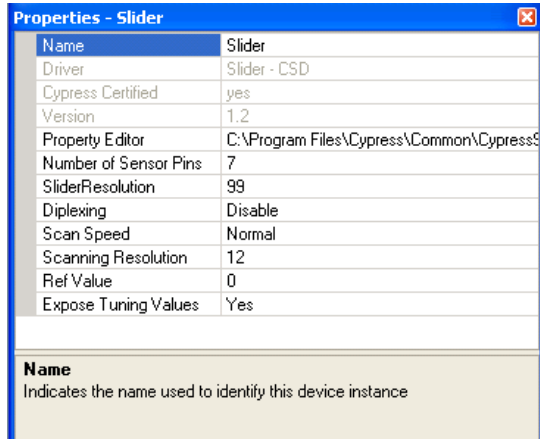


Configure the driver elements to control the color of the multicolor LED array. The drivers are configured with the properties shown in [Figure 4-27](#) to [Figure 4-30 on page 73](#).

Figure 4-27. CSD Properties

Properties - CSDProperties	
Name	CSDProperties
Driver	Properties - CSD
Cypress Certified	yes
Version	1.2
NoiseThreshold	40
NegativeNoiseThreshold	20
BaselineUpdateThreshold	200
Hysteresis	10
Debounce	3
LowBaselineReset	50
Sensors Autoreset	Disabled
Name Indicates the name used to identify this device instance	

Figure 4-28. Slider Properties



In the LED Properties screen, select the Current Mode as "Sourcing".

Figure 4-29. LED Properties

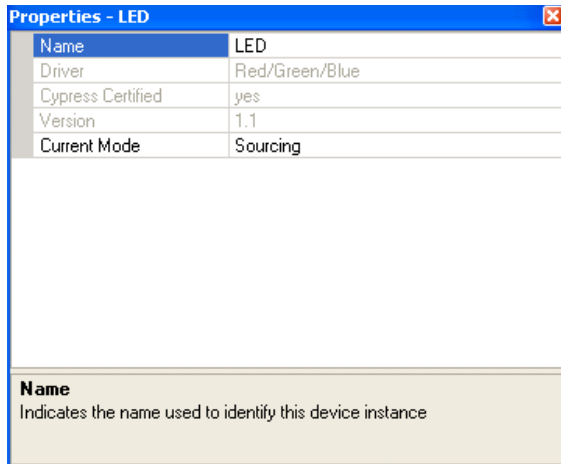
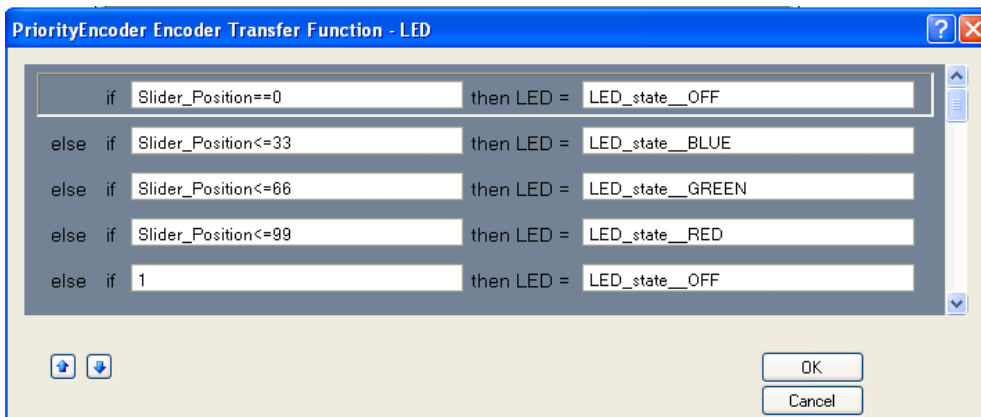


Figure 4-30. LED Transfer Function



This project can be explained through the following statements:

- When the finger position on the slider is at the origin, the LED is OFF.

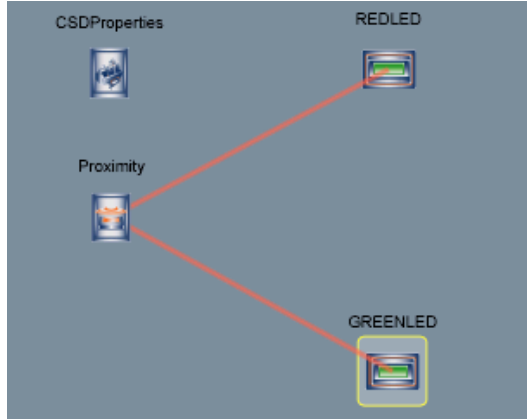
- When the finger position is in between the origin and position 33, the LED emits the color blue.
- When the finger position on the slider is greater than 33 and less than or equal to 66, the LED emits the color green.
- When the finger position is greater than 66 and less than or equal to 99, the LED emits the color red.
- For all other slider positions, the LED is OFF. This includes the absence of a finger on the slider.

4.4.2 MultiFunction Expansion Card Proximity Sensor

The purpose of this project is to demonstrate the capacitive sensing and proximity detection capability of Cypress's PSoC technology. As you move your finger near and far from the proximity detection antenna, the red and green LED turn ON and OFF. [Figure 2-9 on page 19](#) is an illustration of where to locate the application files. This project contains these driver elements:

- CSD
- Proximity
- Green and Red LED

Figure 4-31. Driver Elements



Configure the driver elements to control the turning ON and OFF of the two LEDs. The drivers are configured with the following default properties shown in [Figure 4-32 to Figure 4-37 on page 77](#).

Figure 4-32. CSD Properties

Properties - CSDProperties	
Name	CSDProperties
Driver	Properties - CSD
Cypress Certified	yes
Version	1.2
NoiseThreshold	10
NegativeNoiseThreshold	20
BaselineUpdateThreshold	200
Hysteresis	0
Debounce	3
LowBaselineReset	50
Sensors Autoreset	Enabled

Name
Indicates the name used to identify this device instance

Figure 4-33. Proximity Properties

Properties - CSDProperties	
Name	CSDProperties
Driver	Properties - CSD
Cypress Certified	yes
Version	1.2
NoiseThreshold	10
NegativeNoiseThreshold	20
BaselineUpdateThreshold	200
Hysteresis	0
Debounce	3
LowBaselineReset	50
Sensors Autoreset	Enabled

Name
Indicates the name used to identify this device instance

In the LED Properties screen, select the Current Mode as "Sourcing".

Figure 4-34. Green LED Properties

Properties - GREENLED	
Name	GREENLED
Driver	On/Off with blink - LED - On/Off/
Cypress Certified	yes
Version	1.2
Initial Value	OFF
BlinkRate	2
Current Mode	Sourcing

Name
Indicates the name used to identify this device instance

Figure 4-35. Red LED Properties

Properties - REDLED	
Name	REDLED
Driver	On/Off with blink - LED - On/Off/Blink
Cypress Certified	yes
Version	1.2
Initial Value	OFF
BlinkRate	2
Current Mode	Sourcing

Name
Indicates the name used to identify this device instance

Figure 4-36. Green LED Transfer Function

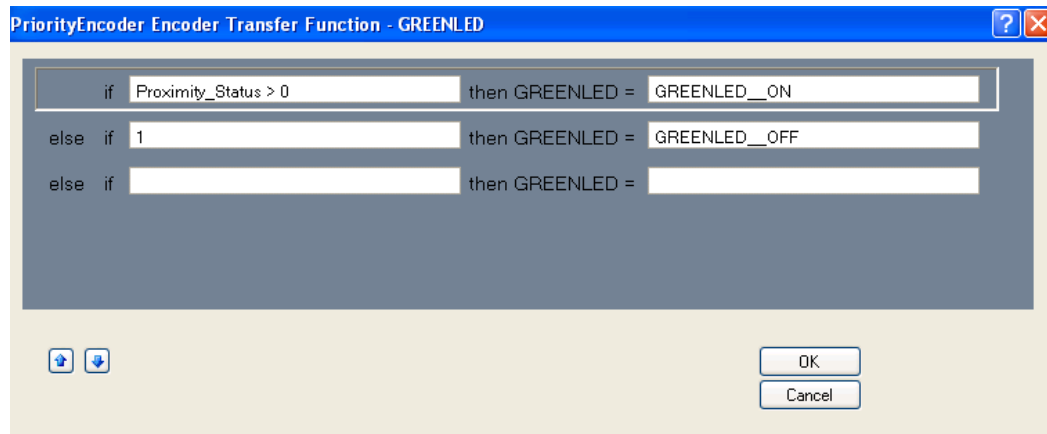
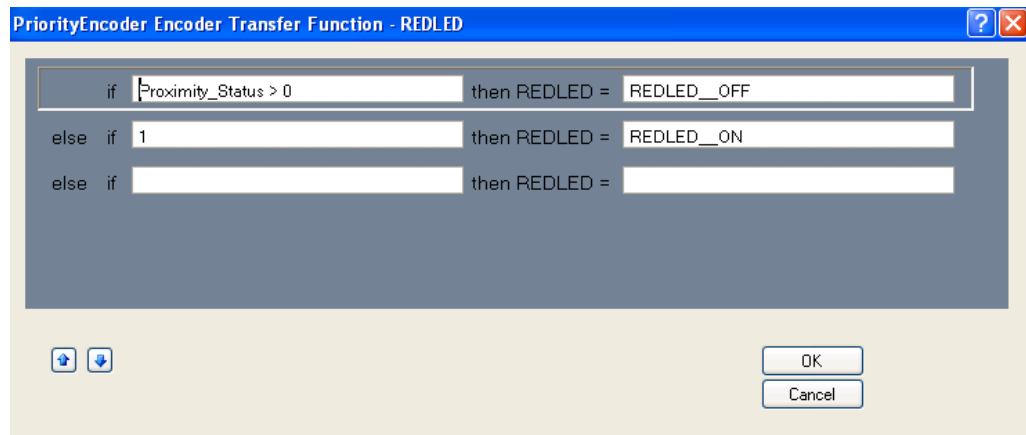


Figure 4-37. Red LED Transfer Function



This project can be explained through the following statements:

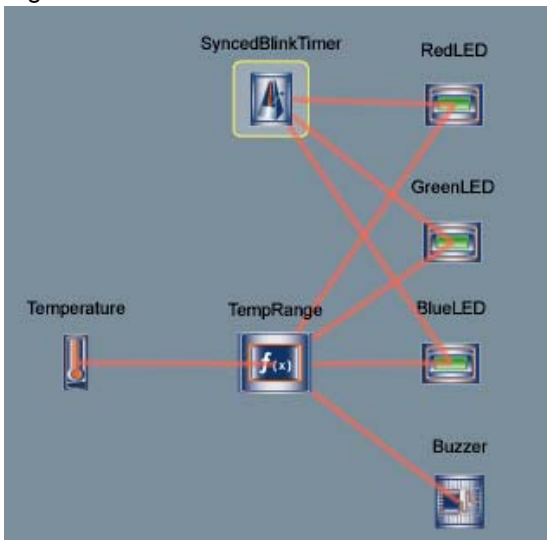
- When the finger is not close enough to the proximity antenna, the red LED is ON and the green LED is OFF.
- As the finger is brought closer and crosses the proximity threshold of the proximity antenna, the red LED turns OFF and the green LED turns ON.
- When the finger is gradually taken away again as it moves further than the proximity threshold, the red LED turns back ON and the green LED turns OFF.

4.4.3 MultiFunction Expansion Card Temperature Sensor

This project demonstrates the temperature sensing, thermistor reading, and calibrating capabilities of the PSoC device. Depending upon the temperature range within which a particular temperature reading is recorded, different colored LED blink (red, green, and blue). When the temperature goes above or below a certain threshold, a Buzzer is sounded out as an alert mechanism. [Figure 2-9 on page 19](#) is an illustration of where to locate the application files. This project contains these driver elements:

- **Temperature.** This is the thermistor driver.
- **TempRange.** This is a Set Point region driver. Setpoints convert a range of input values into a set number of regions. When a new setpoint threshold is added, it divides the region in which it lies into two regions. Setpoints are useful in converting a continuous range of values into a set number of discrete regions.
- **SyncedBlinkTimer.** This is an interval generator driver. This enables you to create a repeating timing interval.
- **LED.** Red, blue, and green LEDs are used to represent the different temperatures acquired by the thermistor.
- **Buzzer.** This is used to create a sound alert.

Figure 4-38. Driver Elements



Configure the driver elements and their associated transfer functions to blink the color coded LEDs for the appropriate temperature ranges and sound the buzzer when the temperature goes above or below extreme thresholds.

The drivers are configured with the following default properties shown in [Figure 4-39](#) to [Figure 4-50](#) on page 82.

Figure 4-39. Temperature Properties

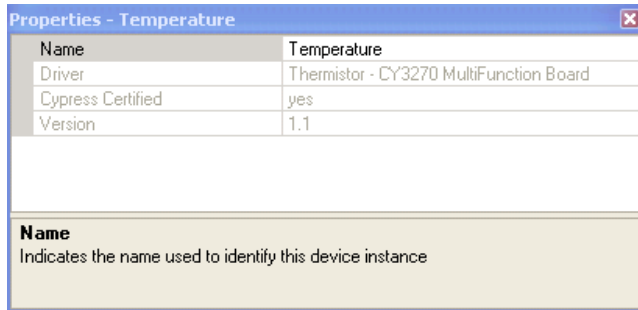


Figure 4-40. Temperature Range Properties

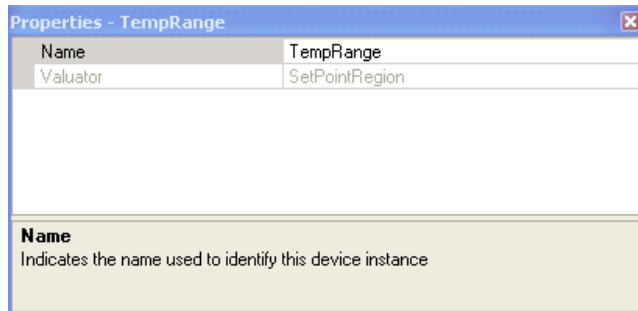
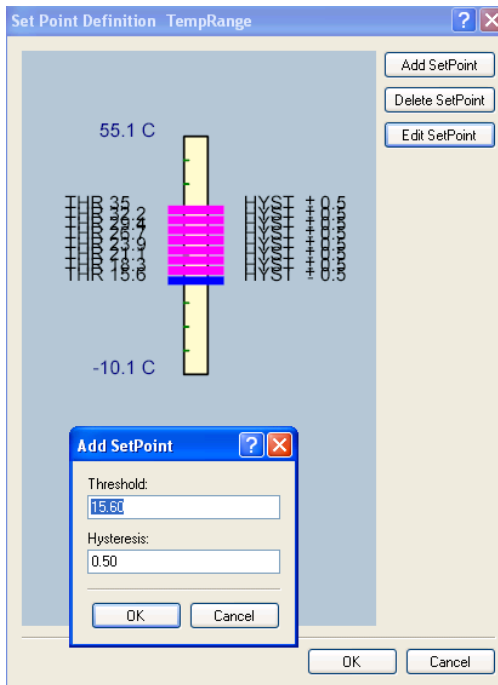


Figure 4-41. Green LED Transfer Function



Different set points are added at different points within the broad range, varying from -10.1°C to 55.1 °C, to demarcate different temperature regions. The points are 15.6°C, 18.3°C, 21.1°C, 23.9°C, 26.7°C, 29.4°C, 32.2°C, and 35°C. Each set point has an associated hysteresis of 0.5.

Figure 4-42. Temperature Range Properties

Properties - SyncedBlinkTimer	
Name	SyncedBlinkTimer
Driver	Interval Generator
Cypress Certified	yes
Version	1.1
IntervalTime	250
IntervalMode	TOGGLE
Name Indicates the name used to identify this device instance	

In the LED Properties screen, select the Current Mode as "Sourcing".

Figure 4-43. Red LED Properties

Properties - RedLED	
Name	RedLED
Driver	On/Off with blink - LED - On/Off/Blink
Cypress Certified	yes
Version	1.2
Initial Value	OFF
BlinkRate	1
Current Mode	Sourcing
Name Indicates the name used to identify this device instance	

Figure 4-44. Blue LED Properties

Properties - BlueLED	
Name	BlueLED
Driver	On/Off with blink - LED - On/Off/Blink
Cypress Certified	yes
Version	1.2
Initial Value	OFF
BlinkRate	1
Current Mode	Sourcing
Name Indicates the name used to identify this device instance	

Figure 4-45. Green LED Properties

Properties - GreenLED	
Name	GreenLED
Driver	On/Off with blink - LED - On/Off/Blink
Cypress Certified	yes
Version	1.2
Initial Value	OFF
BlinkRate	1
Current Mode	Sourcing
Name Indicates the name used to identify this device instance	

Figure 4-46. Buzzer Properties

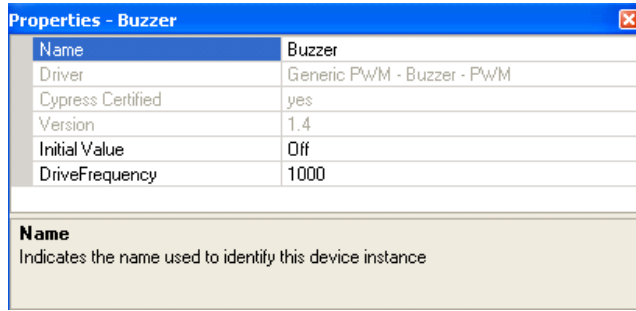


Figure 4-47. Red LED Transfer Function

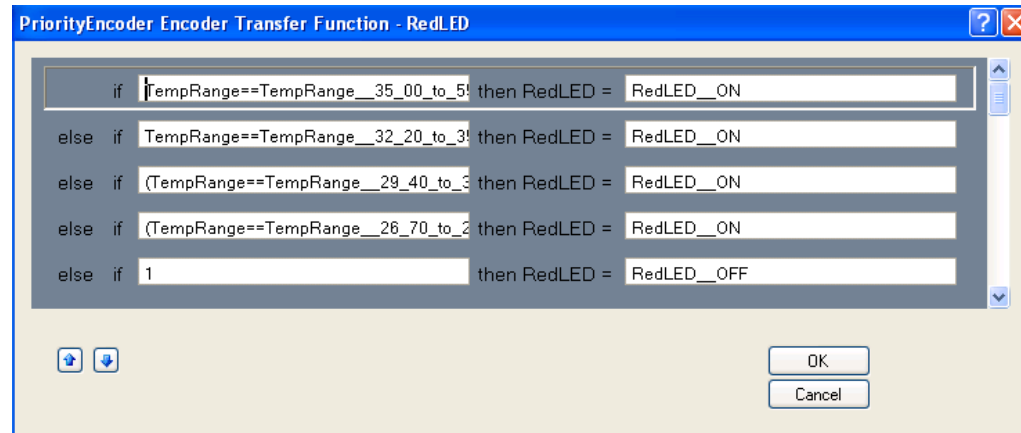


Figure 4-48. Green LED Transfer Function

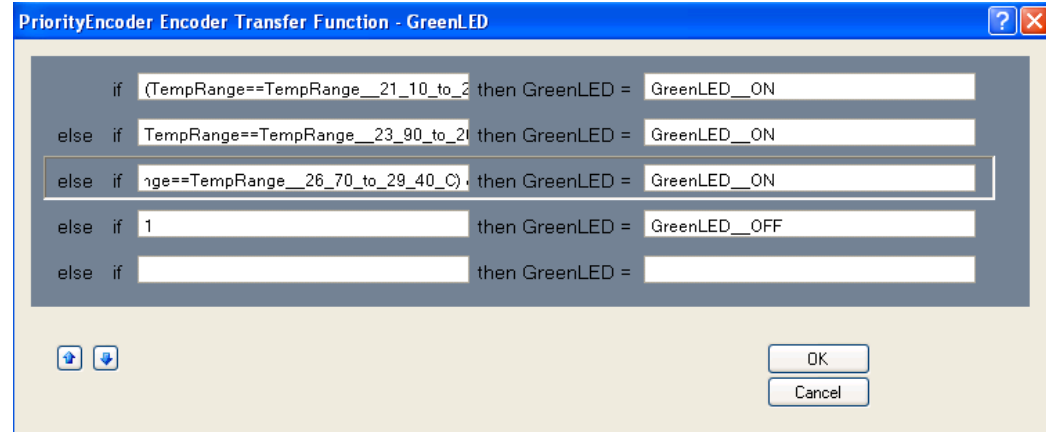
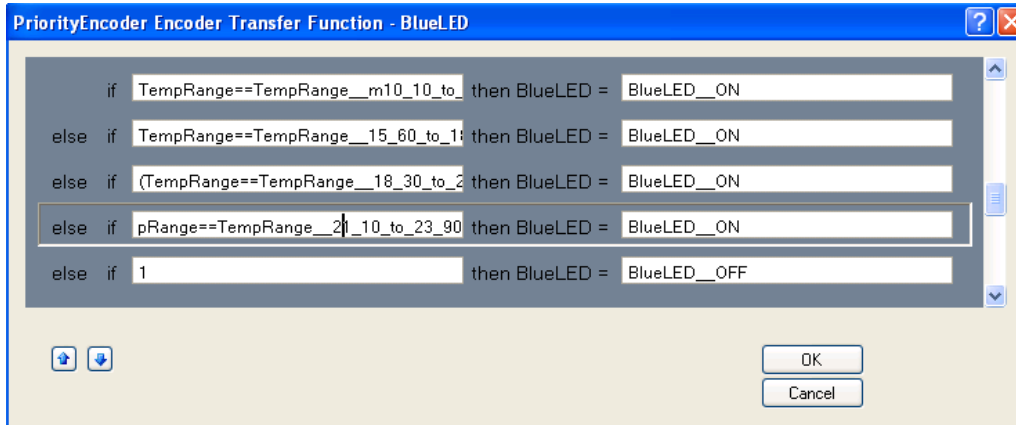


Figure 4-49. Blue LED Transfer Function



PriorityEncoder Encoder Transfer Function - BlueLED

```

if TempRange==TempRange__m10_10_to_ then BlueLED = BlueLED__ON
else if TempRange==TempRange__15_60_to_1: then BlueLED = BlueLED__ON
else if (TempRange==TempRange__18_30_to_2 then BlueLED = BlueLED__ON
else if pRange==TempRange__21_10_to_23_90 then BlueLED = BlueLED__ON
else if 1 then BlueLED = BlueLED__OFF

```

OK Cancel

Figure 4-50. Buzzer Transfer Function

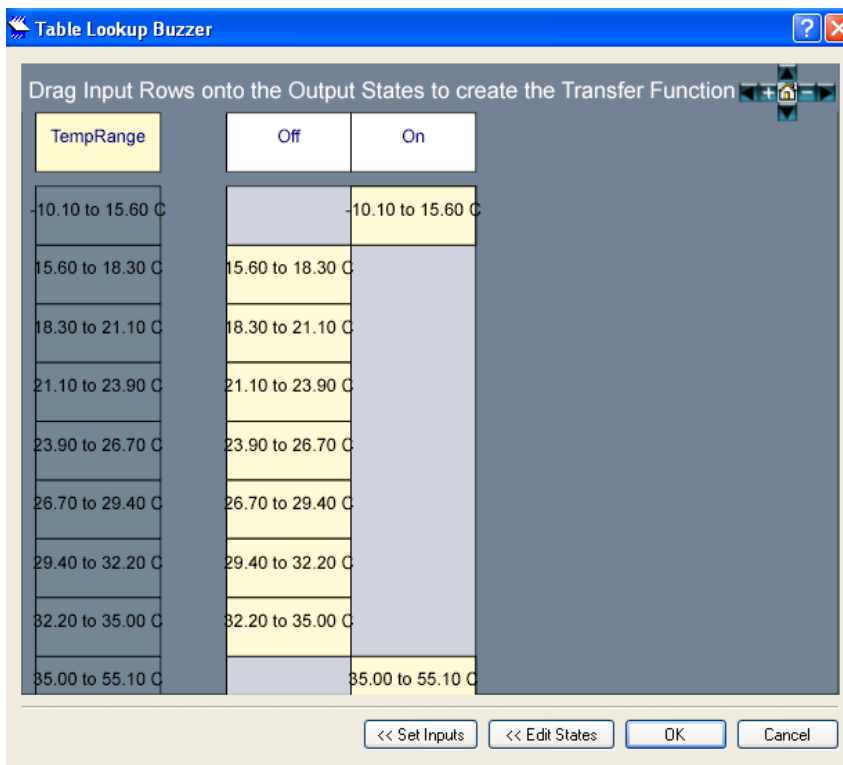


Table Lookup Buzzer

Drag Input Rows onto the Output States to create the Transfer Function

TempRange	Off	On
-10.10 to 15.60 C		-10.10 to 15.60 C
15.60 to 18.30 C	15.60 to 18.30 C	
18.30 to 21.10 C	18.30 to 21.10 C	
21.10 to 23.90 C	21.10 to 23.90 C	
23.90 to 26.70 C	23.90 to 26.70 C	
26.70 to 29.40 C	26.70 to 29.40 C	
29.40 to 32.20 C	29.40 to 32.20 C	
32.20 to 35.00 C	32.20 to 35.00 C	
35.00 to 55.10 C		35.00 to 55.10 C

<< Set Inputs << Edit States OK Cancel

These transfer functions can be explained through the following conditions:

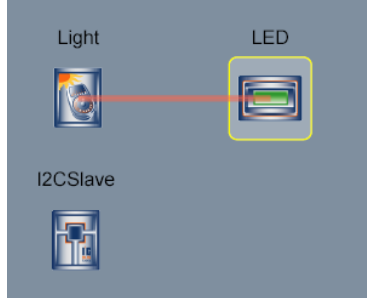
- The Buzzer is sounded only when the temperature is between -10.1°C and 15.6°C, and also when the temperature is between 35.0°C and 55.1°C
- The Red LED is ON only if the temperature is between 26.7°C and 55.1°C.
- The Green LED is ON only when the temperature is between 21.1°C and 29.4°C.
- The Blue LED is ON only when the temperature is between -10.1°C and 23.9°C.
- LEDs blink only if the Blink Timer is triggered.

4.4.4 CY3271 PSoC FirstTouch MultiFunction Expansion Card Light Sensor

The purpose of this project is to demonstrate a light sensor. In this project, the light sensor is used to control the brightness of the LED array. [Figure 2-9 on page 19](#) is an illustration of where to locate the application files. This project contains these driver elements:

- Light
- LED
- I2C Slave

Figure 4-51. Driver Elements



Configure the driver elements to control the brightness of the LED array. The drivers are configured with the following properties and Transfer function as shown in

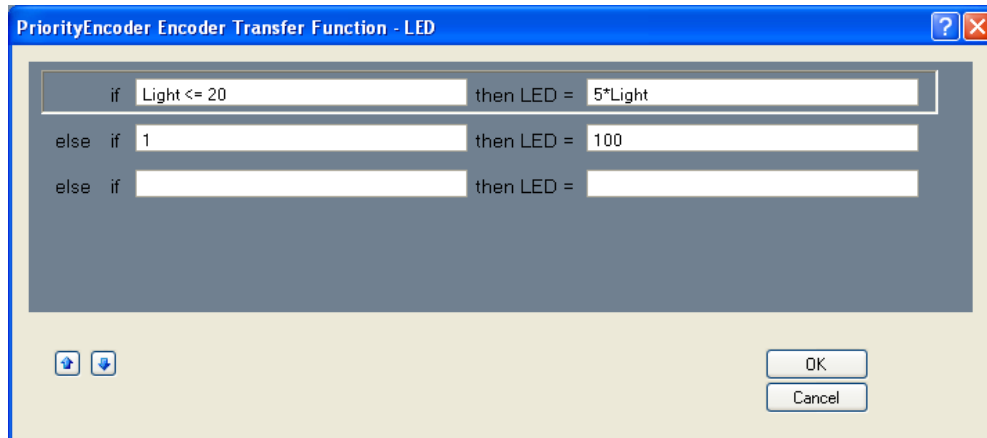
For the Light driver, use the default settings.

In the LED Properties screen, select the Current Mode as “Sourcing”. This is shown in [Figure 4-52](#) to [Figure 4-54 on page 84](#).

Figure 4-52. LED Properties

Properties - LED	
Name	LED
Driver	Intensity-Controlled, Software Con
Cypress Certified	yes
Version	1.1
Initial Value	
Current Mode	Sourcing

Figure 4-53. LED Transfer Function



if	then LED =
Light <= 20	5*Light
else if 1	100
else if	

Figure 4-54. I2C Slave Properties



Name	I2CSlave
Driver	I2C Slave
Cypress Certified	yes
Version	1.1
I2C_Address	5

5. Hardware

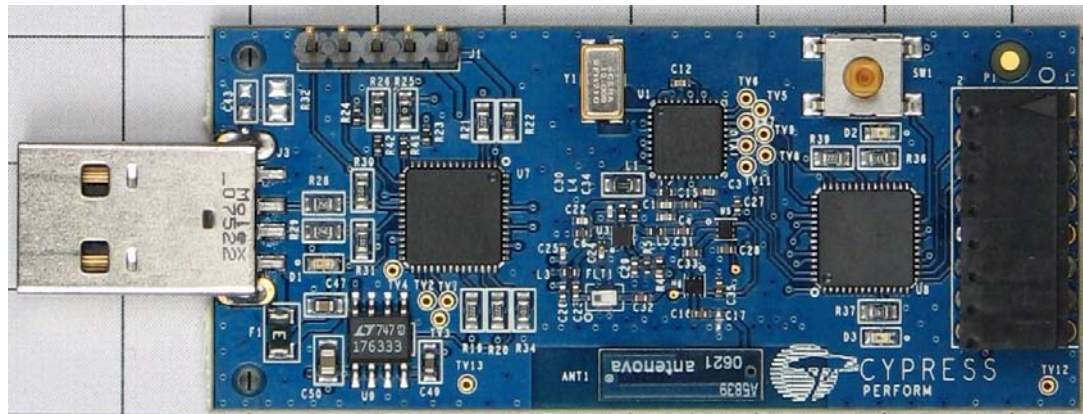


5.1 PC Bridge

The PC Bridge consists of the Hub CY8C24894, the Master CY8C24894, and the CYRF7936 2.4 GHz CyFi Transceiver. It contains a 16-pin connector to connect to the RF Expansion Board or the MultiFunction board, for application data exchange or ISSP programming. The FTPC Bridge is the interface bridge between the expansion cards, your PC, and the various applications.

Since the FTPC Bridge enumerates as a special type of 'composite device' that contains a PSoC Mini-Prog interface, the standard PSoC Programmer utility can identify and communicate with the FTPC bridge. This ensures that your FirstTouch RF Kit is automatically compatible with PSoC Designer.

Figure 5-1. PC Bridge



The master CY8C24894 also acts as a PSoC programmer and downloads the firmware hex file onto the application CY8C24894.

Schematics

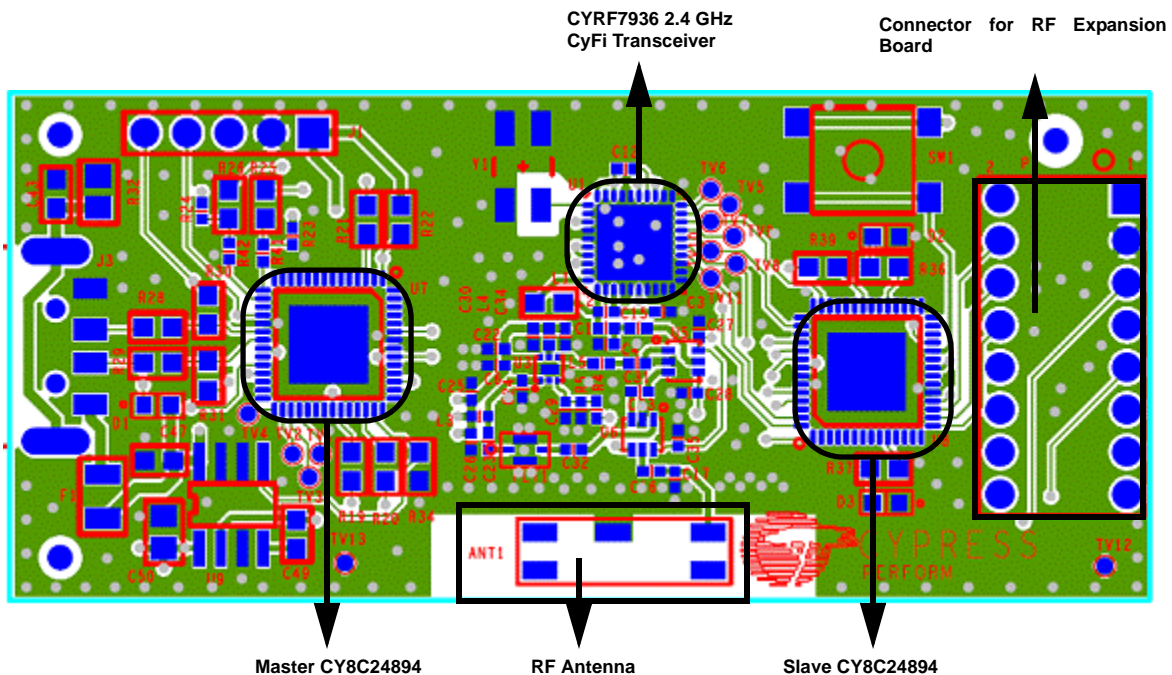
All hardware schematics can be found in C:\Cypress\CY3271\Hardware.

LED Usage

- **Blue LED** The blue LED blinks fast when the bridge is first connected to the USB port of a PC. After hot plug and play is established, it blinks at a periodic interval to indicate that the hub is enumerated and functioning normally.
- **Green LED**
 - The green LED lights up when the hub enters Bind mode. After a successful Bind, the green LED turns OFF. The green LED also turns Off when Bind mode times out. The green LED also blinks when an RF packet is received from any node.
- **Red LED**
 - The red LED indicates activity on the I²C bus when the SCD application is communicating with the CyFiSNP hub application. In the event that the packet buffer in the hub application files up, the red remains solid until the buffer is serviced. When in this condition, the hub application cannot receive additional RF packets from the nodes.

Figure 9-1 on page 103 shows the schematic of the PC Bridge. The layout is shown in Figure 5-2. These images are also available in the hardware directory in your installation directory.

Figure 5-2. PC Bridge Layout



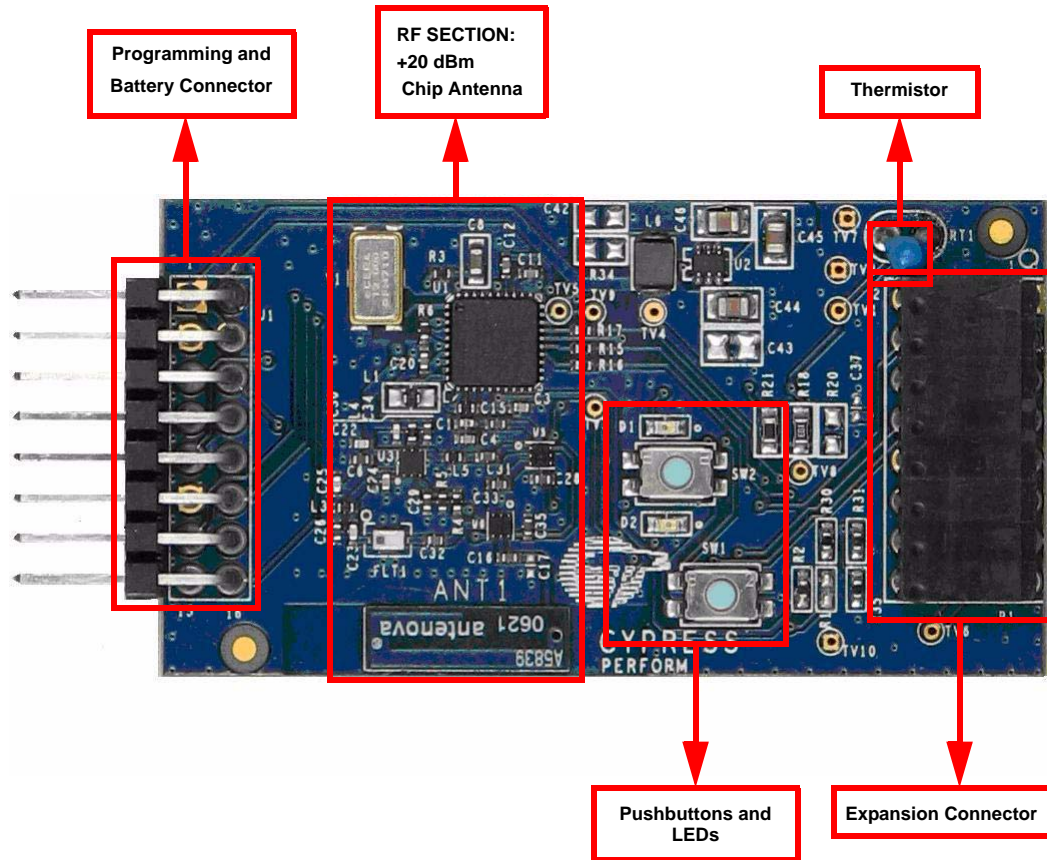
5.1.1 Programming the PC Bridge Application Processor

Select **FirstTouch RF** from the list of programmer devices and then choose the .HEX file to program the application processor. Make sure that there are no expansion cards connected to the expansion connector.

5.2 RF Expansion Card Overview

5.2.1 RF Expansion Card

Figure 5-3. Top View

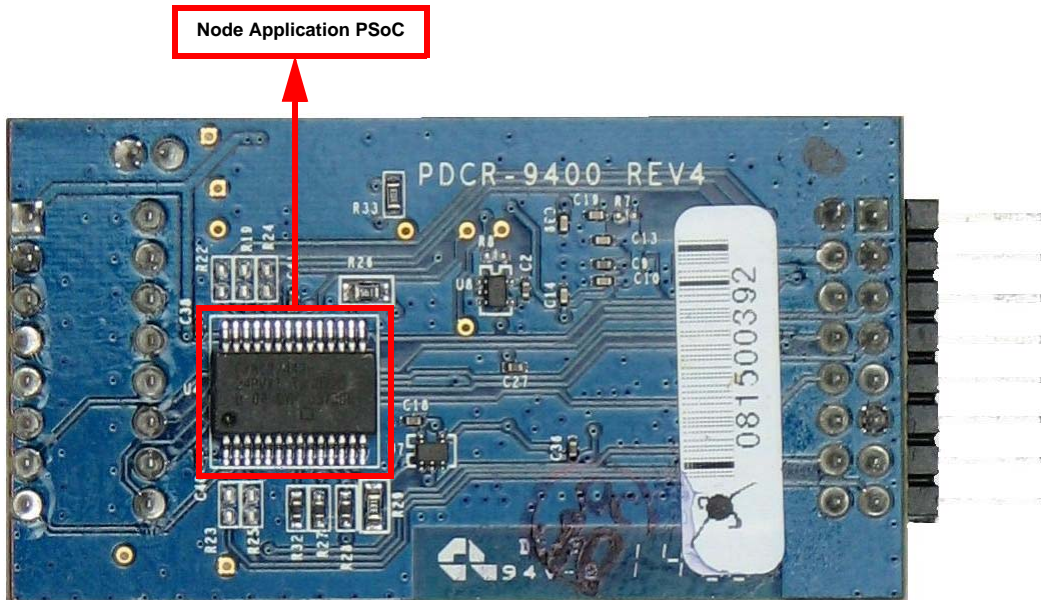


The RF Expansion card acts as the node device. It contains the PSoC CY8C27443, which is the application MCU that controls the radio transceiver CYRF7936 and sensors. The RF Expansion Board also contains the onboard thermistor.

The RF expansion card is designed to plug and play with the FTPC bridge. All power for the included expansion cards is provided by either the AA or the coin cell battery packs. Connection to the FTP Expansion Port is through the 8x2 or 5x2 pin header on the expansion card. The FirstTouch expansion cards have a dedicated host PSoC device installed. The particular PSoC installed is chosen as an example to indicate which PSoC is most suitable for the types of applications supported by a particular expansion card. This also makes it easier to transfer your design from the FirstTouch kit to your hardware.

The schematic is in [Figure 9-2 on page 104](#).

Figure 5-4. Bottom View



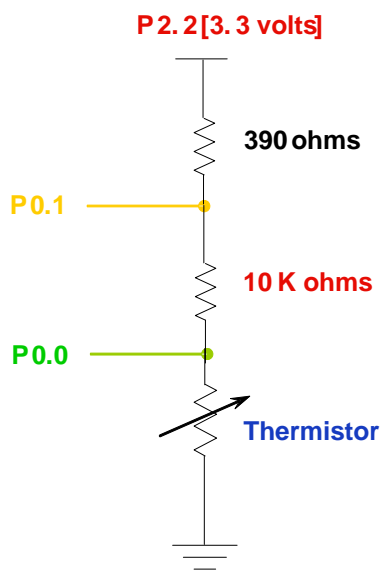
5.2.2 Programming the RF Expansion Card

Connect the RF expansion card into the PC dongle and then select **FirstTouch RF** from the list of programmer devices.

5.2.3 Hardware Design

Figure 5-5 shows the hardware design for a temperature sensor design using a thermistor.

Figure 5-5. Hardware Design



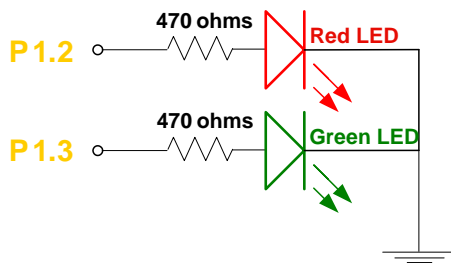
In this operation, P2.2 is supplied with 3.3V, by driving a high (logic '1') to the port pin. This drives the 390 ohm resistor, the reference resistor of 10K, and the thermistor. P0.1 and P0.0 (shown in [Figure 5-5 on page 88](#)) are the analog inputs to the ADC. The resistance of the thermistor changes with respect to temperature and it is about 10K at 25°C.

The 390 ohm resistor helps to prevent the input signal to the ADC from exceeding the supply voltage rails. The ratio of the voltages at P0.0 and P0.1 is proportional to the absolute temperature. This ratiometric reading eliminates inaccuracies because of supply voltage.

5.2.4 LED Connections

The user is provided with two LEDs on the FTRF Expansion Board. The Green LED is connected to P1.3 and the Red LED is connected to P1.2 on the CY8C27443 on the RF Expansion Board. P1.2 and P1.3 are configured as active LOW outputs.

Figure 5-6. LED Connections



After the CD is installed, all hardware schematics are located in a folder such as illustrated in [Figure 2-9 on page 19](#).

LED Usage (As controlled by the Node firmware)

Green LED: The green LED is turned ON when the node enters Bind mode. The green LED is turned OFF if the bind is successful or when the Bind mode times out.

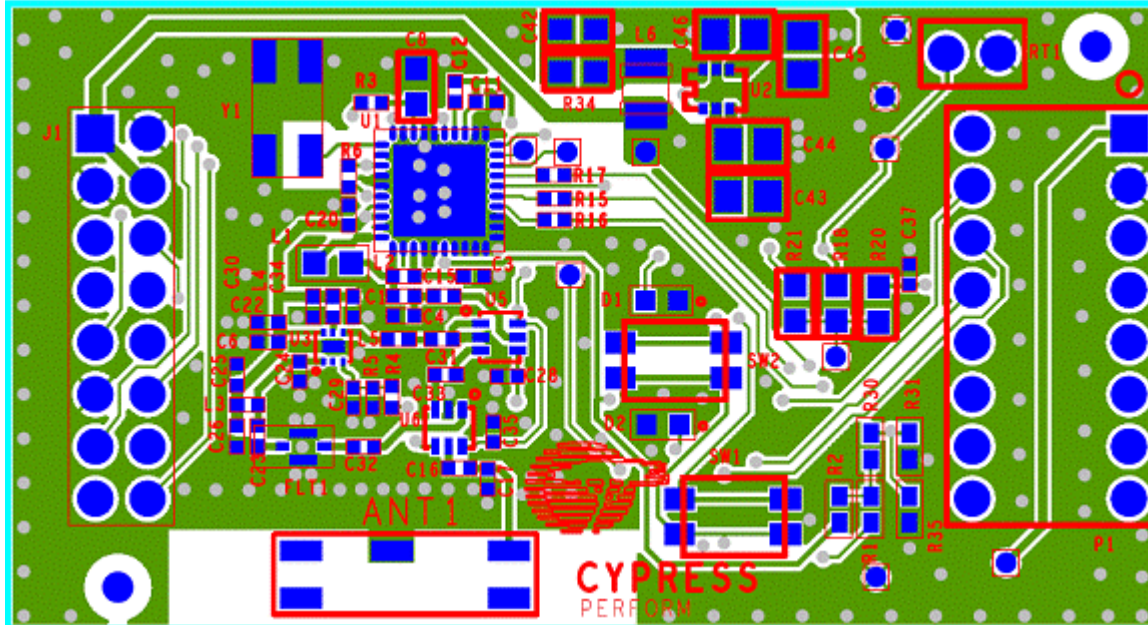
The node firmware causes the green LED to blink upon successful transmission of a data packet. Success of a transmission is determined by reception of the 'ACK' packet from the hub when a data packet is sent.

Red LED: The red LED blinks at a five second interval when bound. When SW2 is pressed for more than two seconds the red LED illuminates solid indicating that the report interval was advanced to the next interval. When SW2 is released the red LED flashes according to the selected interval:

- 1 = 1second
- 2 = 5 seconds
- 3 = 30 seconds
- 4 = 1 minute
- 5 = 5 minutes

The power on default is five seconds.

Figure 5-7. RF Expansion Layout



5.3 MultiFunction Card (FTMF Expansion Card)

The FTMF Expansion Card contains a CY8C21434 PSoC that acts as the 'host' for various demonstrations. The FTMF Expansion Card has hardware to support the following PSoC powered peripheral applications:

- CapSense 'Touch Button'
- CapSense '7-Element Touch Slider'
- CapSense 'NonTouch/Proximity Detection'
- Ambient Light-Level Detection
- Thermistor-based Temperature Measurement

In addition to the above input sensors, the FTMF card also provides the following output devices:

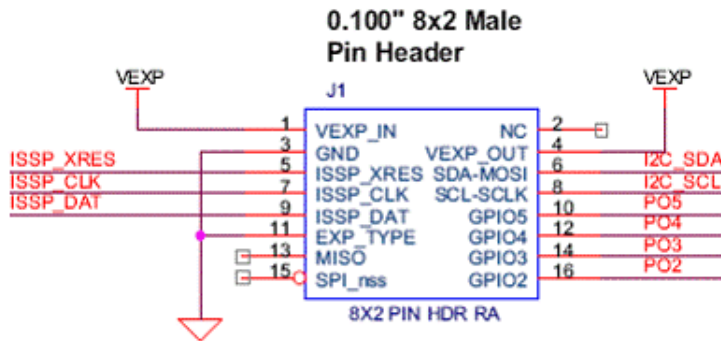
- Red-Green-Blue Triple LED Cluster
- Audible Magnet Transducer or Speaker, or both
- I2C Digital Communications
- Four Unused A/D GPIO Lines for User Functions

The dedicated sensors and output devices on the FTMF Expansion Card help you quickly evaluate and experiment with a variety of PSoC applications, without having to build any hardware. Your PSoC Express or PSoC Designer project completely determines the remaining FTMF Expansion Card functions. The kit installation contains demonstration projects that use the following input sensors:

- CapSense Slider
- Temperature Sensor
- Ambient Light Sensor
- CapSense Proximity Sensor

The FTMF Expansion Card uses a standard FirstTouch expansion header to connect to the FirstTouch RF Expansion Board or other target hardware.

Figure 5-8. FTMF Expansion Card

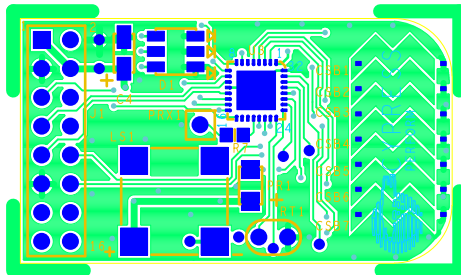


NOTE: This Expansion Board Does Not Have An Onboard Voltage Regulator - DO NOT Power With > 5Vdc

Notice that the 8x2 pin expansion header also includes four General Purpose IO connections labeled P02-P05. These are hard wired to four unused Port 0 IO pins on the CY8C21434 host and allow you to easily connect the FTMF Expansion Card to your specific hardware or sensors. These IO pins are specifically chosen because they can operate as analog outputs, analog inputs, digital inputs, digital outputs, or any combination of the four types. This pin selection makes them true analog or digital GPIO.

You can use the sensors and output devices in any way you want within your project, but you must ensure to always assign the correct pins within your project. Failure to do so may cause unpredictable or unplanned project results.

Figure 5-9. FTMF Layout

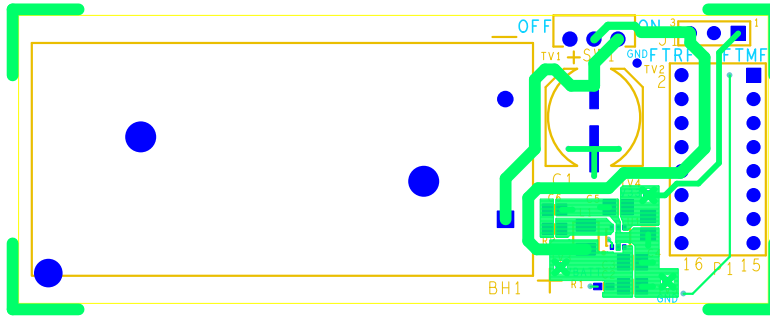


The schematic is shown in [Figure 9-3 on page 105](#). The layout for the MultiFunction Expansion Card is shown in [Figure 5-9](#). These images are also available in the CDROM that is shipped with the kit.

5.4 AAA Power Pack

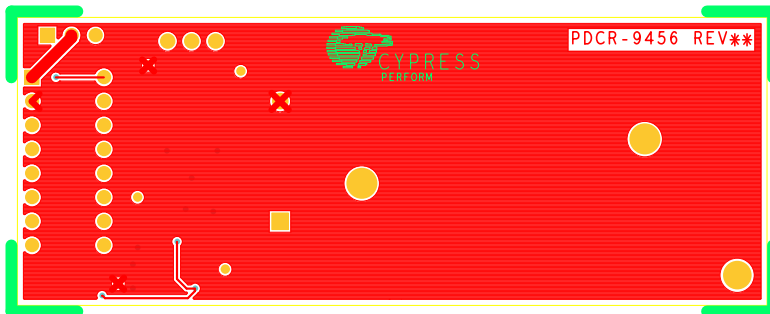
The AAA power pack can hold two AAA batteries and is used to power the RF Expansion cards. It also contains a 16-pin connection header to connect with the RF board. The schematic is shown in [Figure 9-4 on page 106](#).

Figure 5-10. AAA Layout



PDCR-9456 REV ** PRIMARY SIDE

PDCR-9456 REV ** PRIMARY SILKSCREEN



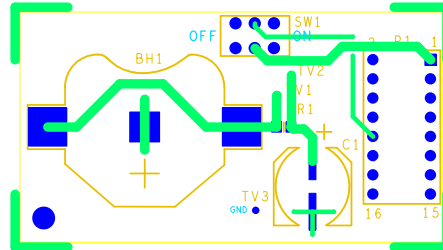
PDCR-9456 REV ** SECONDARY SIDE

PDCR-9456 REV ** SECONDARY SILKSCREEN

5.5 CR2032 Power Pack

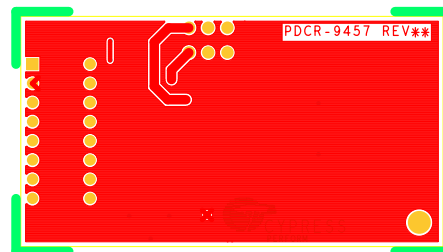
The CR2032 power pack can hold one CR2032 coin cell battery and is used to power the RF Expansion cards. This enables coin cell ultra low-power operation. It also contains a 16-pin connection header to connect with the RF board.

Figure 5-11. CR2032 Layout



PDCR-9457 REV ** PRIMARY SIDE

PDCR-9457 REV ** PRIMARY SILKSCREEN



PDCR-9457 REV ** SECONDARY SIDE

PDCR-9457 REV ** SECONDARY SILKSCREEN

The schematic is shown in [Figure 9-5](#) on page 107.

6. Specifications



6.1 General RF

FTRF operates in the unlicensed worldwide industrial, scientific, and medical (ISM) band:

- 2.400 - 2.483 GHz, up to 0 dBm and 2.412 - 2.460 GHz at +20 dBm (PA enabled)
- Less than 240 mA operating current (Transmit at 20 dBm)
- Transmit power up to +20 dBm (Europe and Japan limited to +10 dBm)
- Receive sensitivity up to -93 dBm
- Operating range of up to 1 km or more
- DSSS data rates up to 250 kbps, GFSK data rate of 1 Mbps

6.2 RF Expansion Card

- Operating voltage from 2.4 to 3.6V
- Operating temperature from 0 to 50°C
- Expansion connector can supply up to 100 mA at 3.3VDC
- Support for I2C and up to 5 General purpose IOs

6.3 PC Bridge

- USB powered
- Support for I2C and up to 5 General purpose IOs
- Operating temperature from 0 to 50°C
- Support programming of the following PSoC families:
 - 21xxx
 - 24xxx
 - 27xxx
 - 29xxx

6.4 MultiFunction Expansion Card

- Operating voltage from 2.4- 5.25V
- Operating temperature from 0 to 50°C

6.5 Certifications

This kit is designed to implement wireless device links operating in the worldwide 2.4 GHz ISM frequency band. It is intended for systems compliant with worldwide regulations covered by:

- Europe ETSI EN 301 489-1, ETSI EN 301 489-7, & ETSI EN 300 328-1
- USA FCC Part 15
- Industry Canada RSS-210 standards.

This device complies with part 15 of the FCC rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference.
2. This device must accept any interference received, including interference that may cause undesired operation.

Note The manufacturer is not responsible for any radio or TV interference caused by unauthorized modifications to this equipment. Such modifications could void the user's authority to operate the equipment.

RF Exposure Warning This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This device must be installed in accordance with the provided instructions and must be operated with minimum 20 cm spacing between the antennas and a person's body during wireless mode of operation. Further, this transmitter must not be co-located or operated in conjunction with any other antenna or transmitter.

7. Frequently Asked Questions



1. What kind of range can I expect from the CY3271?

The CY3271 features CyFi transceivers with power amplification on both the PC Dongle and the RF Expansion Card, for a maximum possible RF output of +20 dBm. This translates to over 400m of line of sight (LOS) range in open space.

However, the firmware projects that are included with the CY3271 limit the RF output power to lower than +20 dBm:

- a. In the Ultra Low Power Temperature Sensor Demo, which uses the CR2032 coin cell battery, the power amplifier circuitry is disabled to extend battery life, and the CyFi transceiver output power is limited to a maximum of 0 dBm. However, because the RF signal still passes through the transmit and receive switches, there is a path loss of approximately 3 dBm. This translates roughly to LOS range of 30m in open space, which is slightly worse than the 50m LOS range that is typically expected from the CyFi transceiver with 0 dBm output power. This is an issue with this specific implementation, and not a reflection on the range that can be achieved by the CyFi transceiver in real implementations.

The following figure shows the difference between the implementation in the CY3271 and a typical implementation.

Figure 7-1. 0 dBm Configuration in CY3271 Sustains 3 dBm Loss Due To Passing Through T/R Switches.

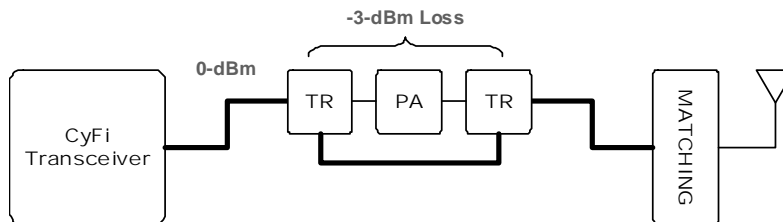
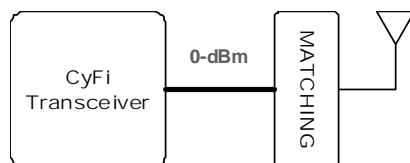


Figure 7-2. In a Typical Non[power Amplified Implementation, the Output of The Transceiver does Not Pass Through T/R Switches, and No Loss Is Sustained.



- b. In the Wireless I2C Bridge project, the power amplifier is limited to a maximum of +10 dBm. This is due to regulatory compliance requirements by ETSI (Europe) and TELEC (Japan). This translates roughly to a range of 150m LOS. However, customers that are using the kit in the United States may choose to modify the included projects, according to the instructions in this User Guide, to increase the maximum RF output power to +20 dBm for prototyping purposes. In that case, the maximum range that the kit is capable of can be achieved.

The following figure illustrates the maximum power output that the kit is programmed for international compliance, vs. what is possible for customers in the United States:

Figure 7-3. Out-Of-Box Maximum Output Power Configuration For International Regulatory Compliance

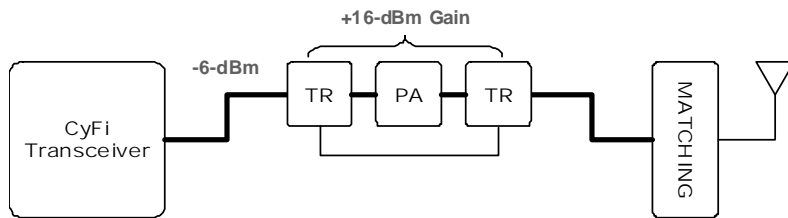
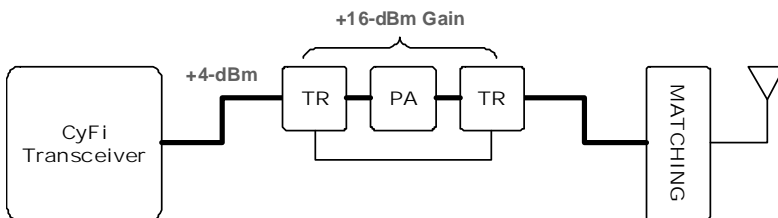


Figure 7-4. Maximum Allowed Output Power Configuration For US Customers



2. What does the red LED on the PC Bridge indicate.

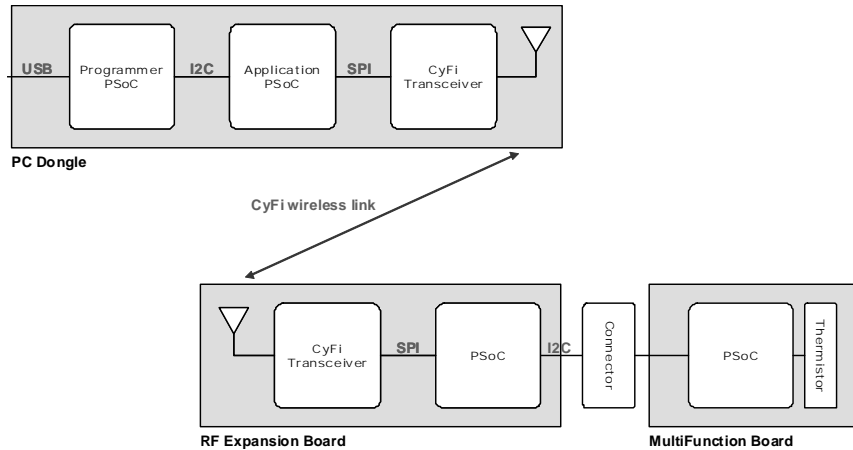
The red LED indicates I2C activity between the SCD software application and the RF Hub application. It should be blinking any time the SCD application is running. If the LED is on solid, this means that the RF Hub packet buffer is full. This will happen when the SCD application is not running but nodes are continuing to send data to the RF Hub. The red LED will also go solid if the SCD software application is unable to keep up with the rate of packets arriving from the nodes.

1. How many devices do I need to implement a typical sensor application?

The CY3271 was built for modularity and adaptability, and therefore uses more PSoC devices to accomplish this goal than would be typically necessary to create a sensor application.

For example, the conceptual block diagram for the Wireless MultiFunction Temperature Sensor Demo, which is one of the many demos that can be performed with the CY3271, is shown in the following figure:

Figure 7-5. Wireless Thermistor System Using CY3271

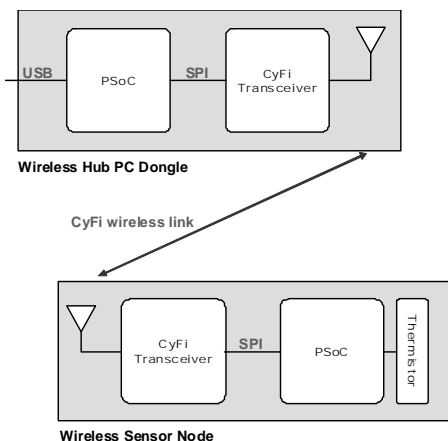


The modularity of the CY3271 system means that sensor boards such as the MultiFunction Board have their own PSoC devices, so that they can operate either standalone (when connected directly into a battery power pack), or in conjunction with the RF Expansion Card (the usage model is shown in [Figure 7-5](#)). The RF Expansion Card has its own PSoC so that it can operate standalone as a wireless node, or optionally accept sensor boards such as the MultiFunction Board (the usage model is shown in [Figure 7-5](#)). This requires PSoC-to-PSoC communication across the expansion connector, which is achieved by using I2C.

The CY3271 PC Dongle was designed to act as the interface for the CY3271 into the PC, as a programmer for all target PSoCs within the CY3271 system, and as the CyFi wireless hub for wireless demos. To accomplish all these goals, two PSoC devices are provided in the PC Dongle. The first PSoC provides a USB-to-I2C bridge to the PC, and also acts as the programmer. The firmware on this PSoC is protected to ensure seamless operation for the entire kit at all times. The second PSoC is a target device, which combined with the on-board CyFi transceiver, acts as the hub of the CyFi network. This PSoC is open for experimentation by the kit user.

The additional functionality that is required from the PC Dongle and the RF Expansion Card necessitates the additional PSoC components discussed here. Contrast this to what a typical implementation in a real system would appear:

Figure 7-6. Wireless Thermistor System in a Typical Implementation



In this implementation, a single PSoC device can be used on the wireless node end to read the thermistor measurement and run the CyFi network protocol stack. Similarly on the wireless hub side, a single PSoC can act as the USB interface into the PC and run the CyFi network protocol stack.

Note that both systems in [Figure 7-5 on page 99](#) and [Figure 7-6](#) accomplish the same ultimate purpose when it comes to reading the value of the thermistor and reporting it wirelessly to a PC. However, because the system in [Figure 7-6](#) does not have the burdens of the CY3271 in terms of the additional functionality it must accomplish, its component count is smaller.

3. Can I evaluate the power consumption of PSoC and CyFi Low-Power RF with the CY3271?

As described in the answer to question 2 in this FAQ, the CY3271 was designed to demonstrate a wide range of functionality in a very modular way.

Because of this, there was a need to accommodate a wide range of voltage and current requirements that make up the CY3271 system. This necessitated the use of several voltage converters that ensure that the various boards in the CY3271 work both standalone and in tandem.

For example, because the CY3271 enables the user to connect sensor boards into the expansion header of the RF Expansion Card, the power configuration of the AAA Power Pack and the RF Expansion Card was designed to supply a fairly large amount of current.

This makes the CY3271 an ideal system to experiment with the functionality of PSoC devices and CyFi Low-Power RF, but not an ideal system to evaluate their power consumption.

However, the CY3271 does indeed demonstrate low power operation with the Ultra Low Power Temperature Sensor demo, which uses the CR2032 Power Pack.

4. What if I wanted more than one wireless node?

The CY3271 ships with one wireless node, the RF Expansion Card, out of the box. Customers that are interested in creating wireless networks with a larger number of nodes can order the CY3271-RFBOARD, which features two additional RF Expansion Cards, and two additional AAA Power Packs.

5. What other kits work with the CY3271?

The modularity of the CY3271 enables a large variety of sensor boards to be added to the kit to showcase more applications.

Cypress offers the CY3271-EXP1 Sensor Expansion Pack, which features two sensor expansion boards:

- Pigtail Thermistor Board
- Weather Station Board, featuring temperature, pressure, humidity, and ambient light sensors.

The CY3271-EXP1 is only the first of many CY3271-EXP* kits that will be released by Cypress to extend the functionality of CY3271.

6. Can I create my own sensor boards to work with the CY3271?

Yes, you can create a sensor board that connects into the RF Expansion Card, which both supplies your board with power, and enables you to transmit sensor data to a PC using CyFi Low-Power RF.

To do so, follow these steps:

- a. Your sensor board must use a 8x2 0.100" male header that mates with the RF Expansion Card's 8x2 0.100" female receptacle.
- b. The header signals must match the specification provided as part of the RF Expansion Card schematic in the Appendix of this User Guide. These include an I2C interface, 5 GPIOs, VDD, and ground.
- c. Limit the current consumption on your sensor board to 50 mA.

7. What else can I do with the CY3271 hardware?

After you are done experimenting with the CY3271 kit itself, you can re-use the RF Expansion Card as a wireless module for development and prototyping purposes.

The RF Expansion Card's 8x2 0.001" male header, which features an I2C interface, 5 GPIOs, and connections for power and ground mates easily with common prototyping hardware. The connector specification is available in the RF Expansion Card schematic, located in the Appendix of this User Guide.

8. I have evaluated PSoC and CyFi Low-Power RF, and I want to continue with development. What are the next steps?

The recommended next step is to acquire the CY3210-CYFI Low-Power RF Development Kit.

The CY3210-CYFI features:

- Two PSoC Evaluation Pods (29xxx family and 24x94 family)
- Two CyFi transceiver RF modules
- Two development baseboards with ZIF sockets, RF module headers, generous prototyping area, flexible power options, LCD interface, in-system programming, and RS232 interface
- Firmware tutorials and examples
- PSoC Designer software
- Sense and Control Dashboard software

Though the CY3210-CYFI ships with PSoC Evaluation Pods for the 29xxx and 24x94 families, it enables development with most other PSoC families, allowing you to choose the PSoC device that is most suited for your application.

In addition, visit <http://www.cypress.com/training> to find training on PSoC devices and CyFi Low-Power RF.

Figure 9-1. PC Bridge Schematic

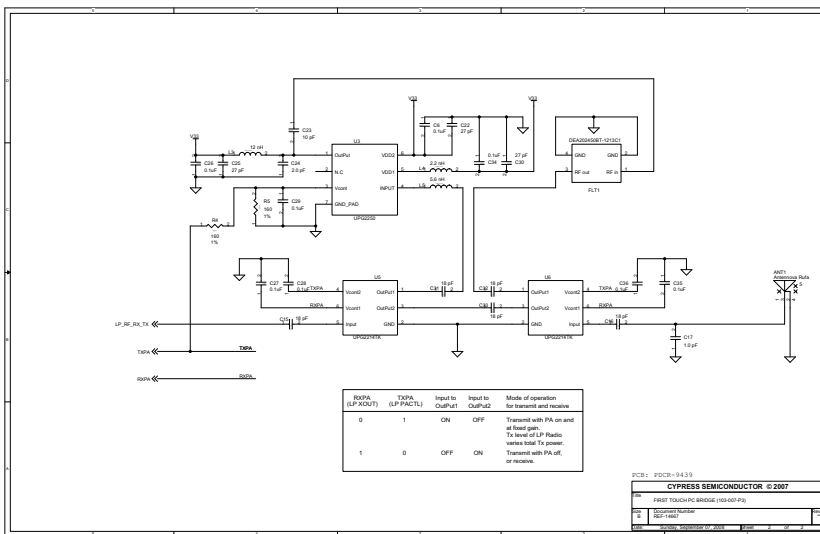
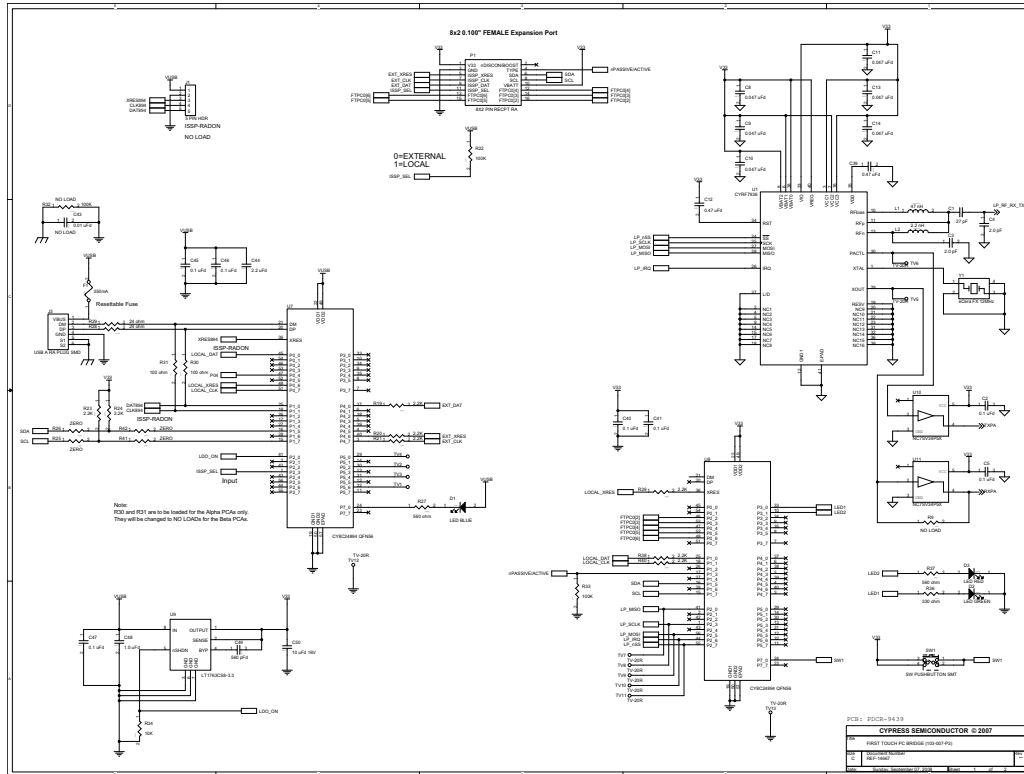


Figure 9-2. RF Expansion Card Schematic

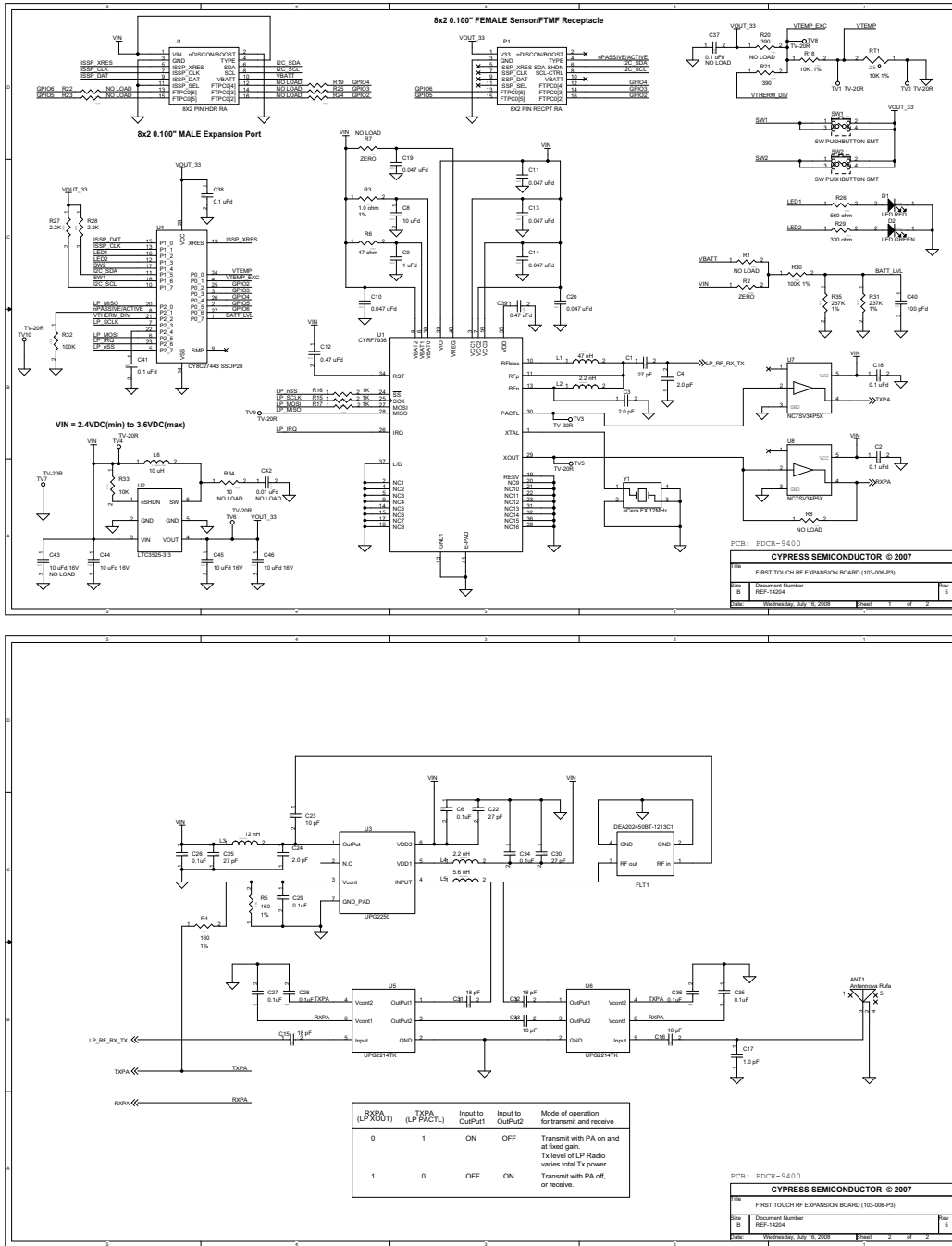


Figure 9-3. FTMF Expansion Card Schematic

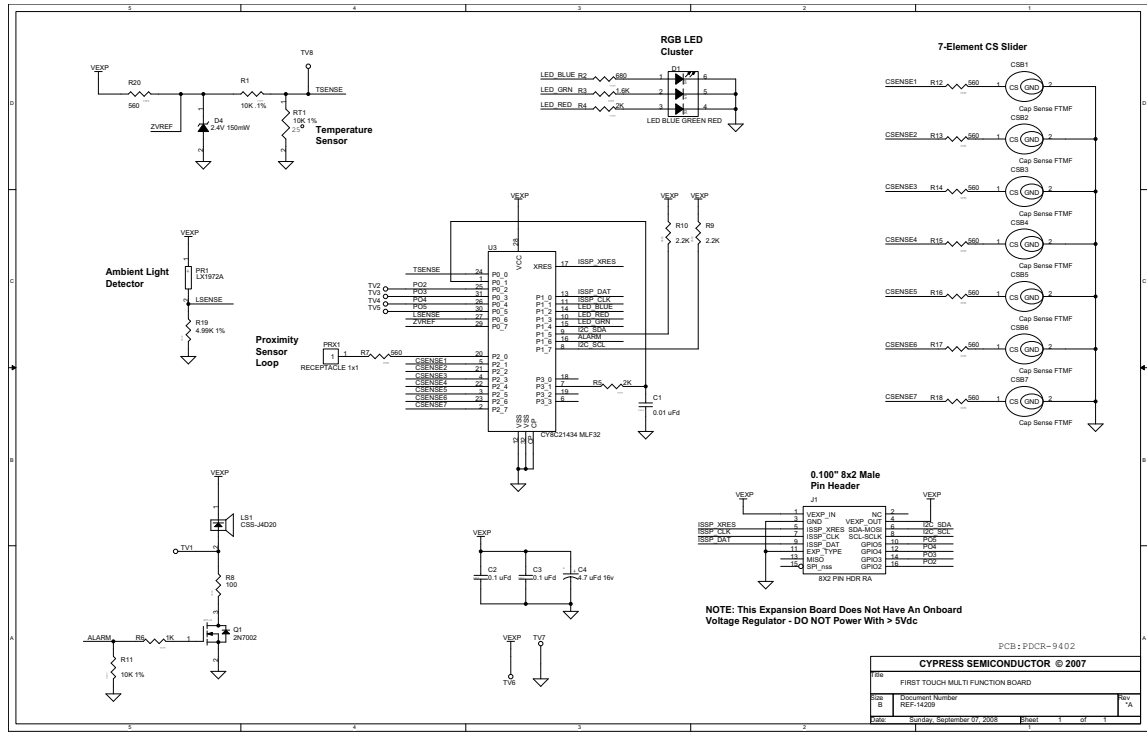


Figure 9-4. AAA Power Pack Schematic

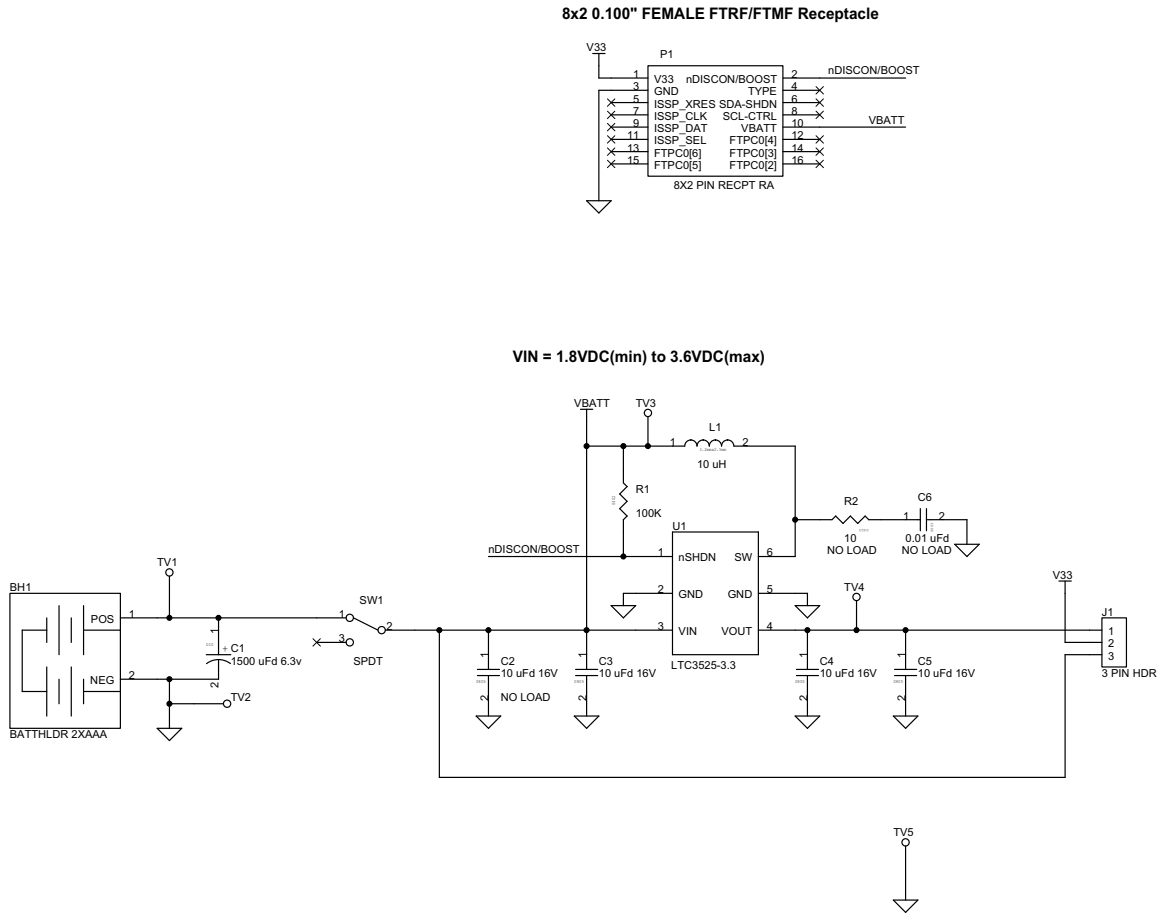


Figure 9-5. CR2032 Power Pack Schematic

8x2 0.100" FEMALE FTRF/FTMF Receptacle

