Freescale ZigBee™ Application

User's Guide

Document Number: ZAUG
Rev. 1.3
01/2008

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only*:**
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Contents

Chapter 6
Example Applications

About This Book

The *ZigBee Application User's Guide* explains how to install and run the ZigBee Feature Set example applications that are included in Freescale's BeeKit Wireless Connectivity Toolkit.

Audience

This document is the operator's manual for the ZigBee example applications in BeeKit. Anyone needing to demonstrate the applications or to become familiar with their behavior should read this manual. ZigBee application developers should read this manual along with the *ZigBee Application Development Guide* to understand what the applications do and how they do it.

Organization

This document is organized into the following sections.

Chapter 1 Introduction – introduces the Freescale implementation of ZigBee wireless sensor networks.

Chapter 2 Freescale Development Boards – provides detailed installation and device configuration information using the Freescale BeeKit Wireless Connectivity Toolkit tools.

Chapter 3 BeeKit and CodeWarrior – provides a simple home lighting control network to introduce users to simple ZigBee applications.

Chapter 4 Starting and Running a Simple ZigBee Network – provides a quick tutorial to form a ZigBee network using code built and loaded into two development boards in previous chapters.

Chapter 5 Creating a Wireless UART Application - shows how to create a Freescale Wireless UART application using the Freescale BeeKit Wireless Connectivity Toolkit.

Chapter 6 Example Applications – provides several examples to allow users to configure and run ZigBee wireless home control applications.

Conventions

This document uses the following conventions:

`Courier` Is used to identify commands, explicit command parameters, code examples, expressions, data types, and directives.

Italic Is used for emphasis, to identify new terms, and for replaceable command parameters.

All source code examples are in C.

Definitions, Acronyms, and Abbreviations

APS	Application Support sub-layer, a ZigBee stack component
APL	Application Layer, a ZigBee stack component
BDM	Background Debug Mode: The HCS08 MCUs used here have a BDM port that allows a computer to program its flash memory and control the MCU's operation. The computer connects to the MCU through a hardware device called a BDM pod. In this application, the pod is the P&E USB HCS08/HCS12 Multilink
BeeKit	Freescale Wireless Connectivity Toolkit networking software
Binding	Associating two nodes in a network for specific functions (e.g., a light and switch)
Cluster	A collection of attributes accompanying a specific cluster identifier (sub-type messages.)
EVB	Evaluation Board, a Freescale development board.
GUI	Graphical User Interface: BeeKit and CodeWarrior, the two Windows tools discussed here, each uses a GUI
HCS08	A member of one of Freescale's families of MCUs
IDE	Integrated Development Environment: A computer program that contains most or all of the tools to develop code, including an editor, compiler, linker, and debugger
MAC	IEEE 802.15.4 Medium Access Control sub-layer
MCU	Micro Controller Unit: A microprocessor combined with peripherals, typically including memory, in one package or on one die
NCB	Network Coordinator Board, a Freescale development board
Node	A device or group of devices with a single radio
NWK	Network Layer, a ZigBee stack component
OUI	Organizational Unique Identifier (The IEEE-assigned 24 most significant bits of the 64-bit MAC address)
PAN	Personal Area Network
Profile	Set of options in a stack or an application
1320x-QE128EVB	1320x-QE128 Evaluation Board, a Freescale development board with an MC1320x transceiver and an MC9S08QE128 MCU
SARD	Sensor Application Reference Design, a Freescale development board
SMAC	Freescale Simple MAC, a very simple, very small proprietary wireless protocol that uses the Freescale IEEE 802.15.4 radios
SRB	Sensor Reference Board, a Freescale development board
SCI	Serial Communication Interface. This is a hardware serial port on the HCS08. With the addition of an external level shifter, it can be used as an RS232 port
SPI	Serial Peripheral Interface. This is a serial port intended to connect integrated circuits that are together on one circuit board
SSP	Security Service Provider, a ZigBee stack component

Stack	ZigBee protocol stack
Toggle	A toggle switch moves from one state to its other state each time it is toggled. For instance, if the first toggle takes the switch to Off, the next toggle will be to On, and the one after that will be to Off again. In the applications this document describes, the switches are momentary push buttons with no memory of their states. The HCS08 maintains each switch's state
UART	Universal Asynchronous Receiver Transmitter, an MCU peripheral for access to devices not on the same circuit board. With level shifting, the UART implements RS-232
UI	User Interface
ZC	ZigBee Coordinator: one of the three roles a node can have in a ZigBee network
ZED	ZigBee End Device: one of the three roles a node can have in a ZigBee network
ZR	ZigBee Router: one of the three roles a node can have in a ZigBee network
802.15.4	An IEEE standard radio specification that underlies the ZigBee specification

References

The following documents were referenced to build this document.

1. Freescale *BeeStack Software Reference Manual*, Document BSSRM, February 2007.
2. Document 053474r13, *ZigBee Specification*, ZigBee Alliance, December 2006
3. Document 075123r00, *ZigBee Cluster Library Specification*, ZigBee Alliance, July 2007
4. Document 053520r24, *Home Automation Profile Specification*, ZigBee Alliance, September 2007
5. The data sheets for the MC13193, MC13203, MC1321x radios
6. Freescale *MC9S08GB/GT Data Sheet*, Document MC9S08GB60, December 2004

Revision History

The following table summarizes revisions to this manual since the previous release (Rev. 1.2).

Revision History

Location	Revisions
Entire document	Updated for software release and added QE128 information.

Chapter 1
Introduction

The Freescale BeeKit Wireless Connectivity Toolkit includes a set of example applications for the ZigBee Feature Set using the Home Automation application profile, supplemented by Accelerometer and Wireless UART example applications. This user's guide describes how to

- Configure the applications in BeeKit for any of the Freescale development boards that BeeKit supports
- Export the configured applications from BeeKit
- Import the configured applications into CodeWarrior
- Build the applications in CodeWarrior
- Load the applications into Freescale development boards using a BDM pod
- Run the applications on the boards

These applications require the installation of the BeeKit Wireless Connectivity Toolkit, including the BeeStack Codebase, and CodeWarrior for the HCS08, version 6.1 or later. They also require a P&E Multilink Background Debug Mode pod or its equivalent to program the development boards. This document assumes that all of these are correctly installed.

This user's guide assumes familiarity with the purpose and major features of ZigBee Wireless Networks. It explains only enough of BeeKit and CodeWarrior to get the applications into the development boards. These much larger topics are explained in the references.

1.1 What This Document Describes

This document explains how to:

- Get the example ZigBee applications in BeeKit into Freescale development boards
- Run the applications

1.2 What This Document Does Not Describe

This document does not explain how to:

- Install BeeKit or CodeWarrior
- Understand or modify the application code
- Port the applications or BeeStack to a platform that BeeKit does not already support
- Learn about ZigBee. For a tutorial on ZigBee, go to www.freescale.com/zigbee and click on Online Training

Chapter 2
Freescale Development Boards

Freescale has these development boards:

- Network Coordinator Board (NCB)
- Sensor Reference Board (SRB)
- Evaluation Board for QE128 MCUs (1320x-QE128EVB)
- Axiom GB60 (AXM-0308)
- Sensor Application Reference Design (SARD)
- MC1319x Evaluation Board (EVB)

Table 2-1. Freescale Development Boards

Board Number	Device Type	Serial Interface	Board Features
MC1321x-NCB	MC13213 (SiP using MC68HCS908GT60)	USB and RS232	LCD with two 16-character rows, PWM-driven speaker
MC1321x-SRB	MC13213 (SiP using MC68HCS908GT60)	USB	3-axis accelerometer, temperature sensor, PWM-driven speaker
1320x-QE128EVB	MC13202 and socket for QE128 MCUs such as Flexis 8-bit MC9S08QE128	USB and RS232	LCD with two 16-character rows
AXM-0308	MC13193 and MC68HCS908GB60	RS232	RF is on a daughter card
SARD	MC13193 and MC68HCS908GT60	RS232	3-axis accelerometer
EVB	MC13193 and MC68HCS908GT60	USB and RS232	

2.1 HCS08 Board Details

Each of these boards has:

- a Freescale HCS08 MCU:
  - 60KB flash for GB/GT60 MCUs or 128 KB flash for MC9S08QE128
  - 4KB static RAM for GB/GT60 MCUs or up to 8KB for MC9S08QE128
  - Two Serial Communication Interface (SCI) ports
    - If a board has an RS232 connector, it uses SCI 1
    - If a board has a USB connector or a 2nd RS232 connector, it uses SCI 2
  - One Serial Peripheral Interface (SPI) port connected to the ZigBee radio

- Four push button switches (S1-S4 or SW2-SW5 on 1320x-QE128EVB) on MCU input pins. Pin assignments are not identical across all boards
- Four LEDs (LED1-LED4) on MCU output pins. Pin assignments are not identical across all boards
- A six-pin Background Debug Mode (BDM) connector for programming the flash and using the debugger

The applications use the four switches and the four LEDs identically on all boards even though the pin assignments differ, as explained in the next chapter.

The 1320x-QE128EVB, EVB, NCB, and SRB have a USB-UART translator circuit. That does not affect the code on the MCU (the USB port looks like an RS232 port), but the USB port can power the board, and it is easier to connect more than a few boards to a computer if they use USB. The boards are shown in the following figures.

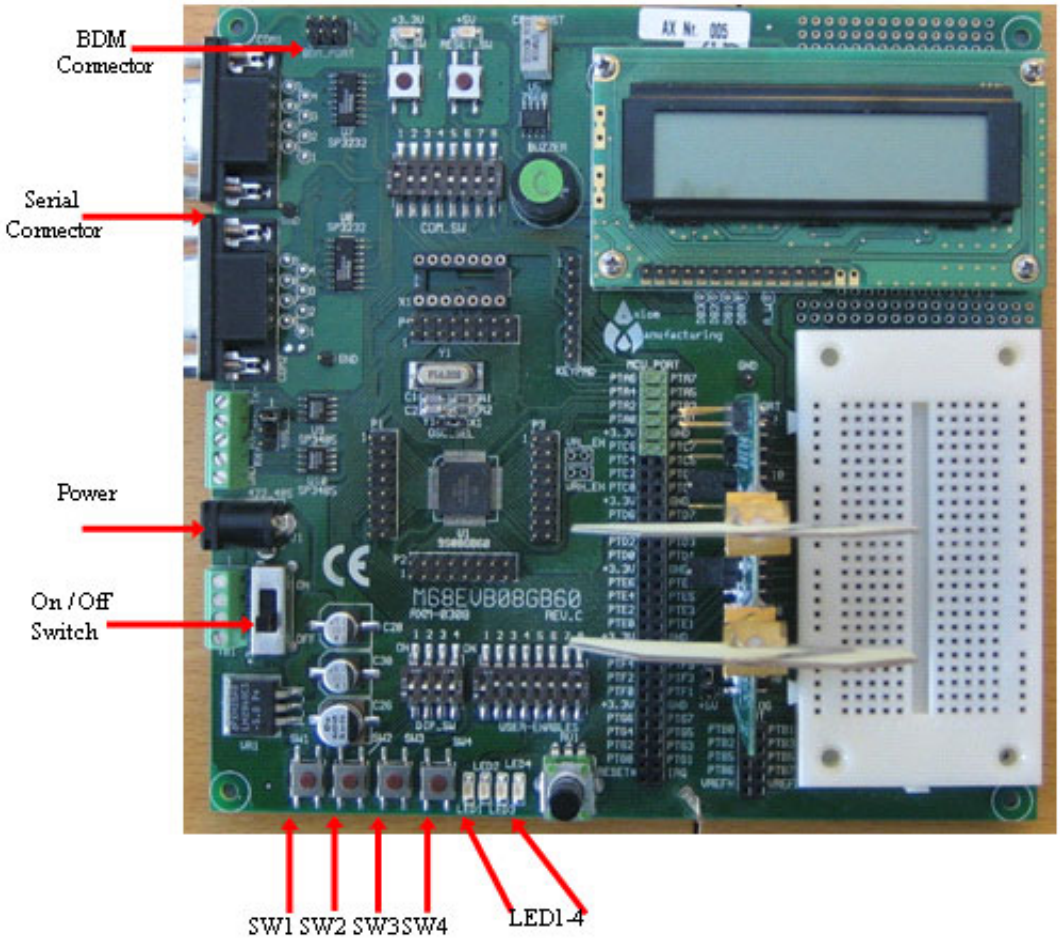

Figure 2-1. Axiom AXM-0308 Evaluation Board

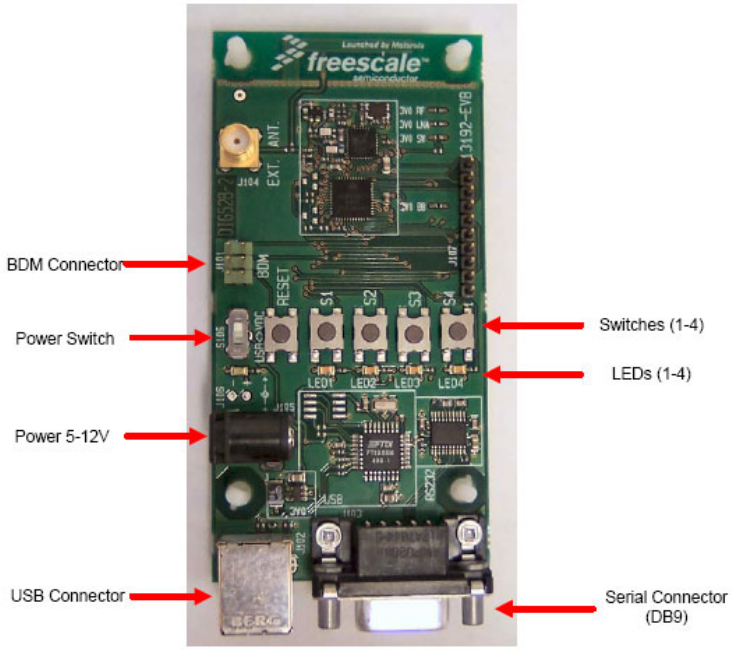

Figure 2-2. EVB (DIG528-2) Evaluation Board

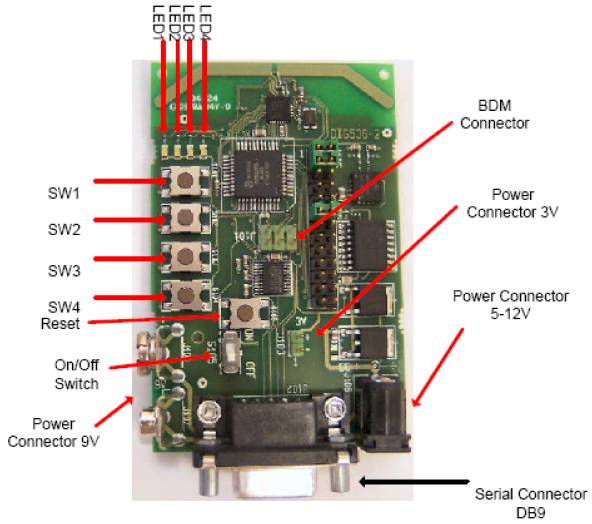

Figure 2-3. SARD (DIG536-2) Evaluation Board

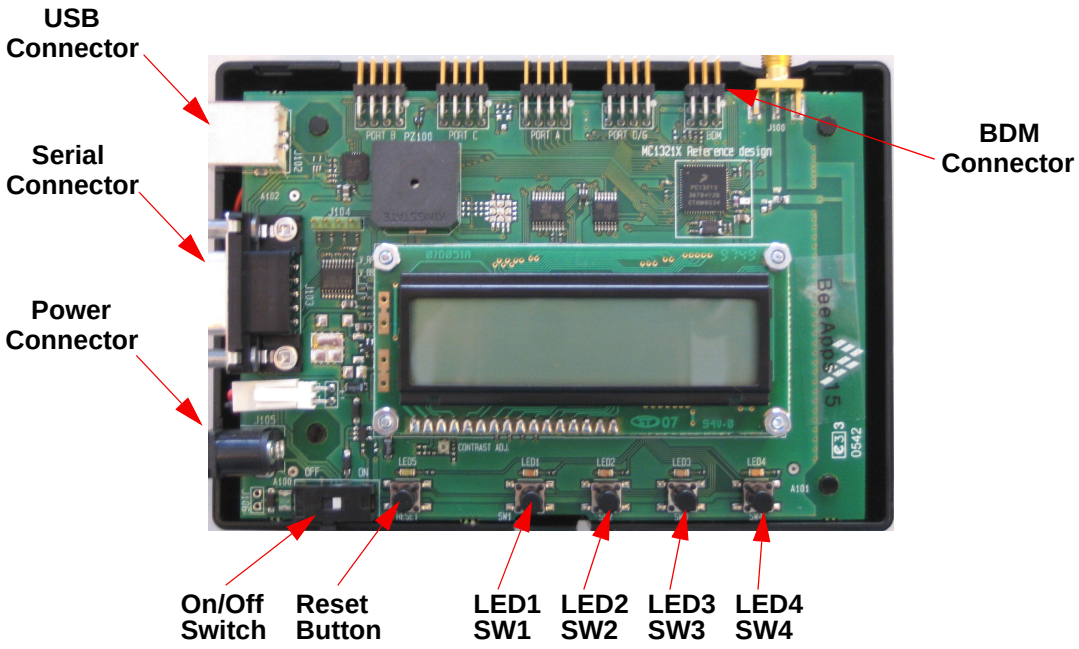

Figure 2-4. NCB Evaluation Board

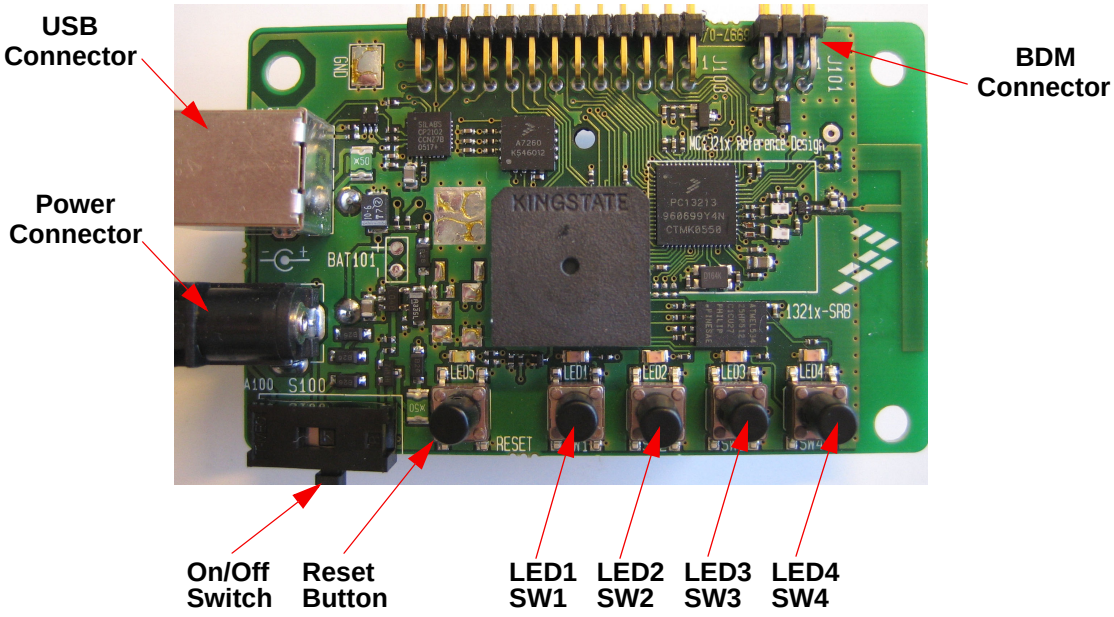

Figure 2-5. SRB Evaluation Board

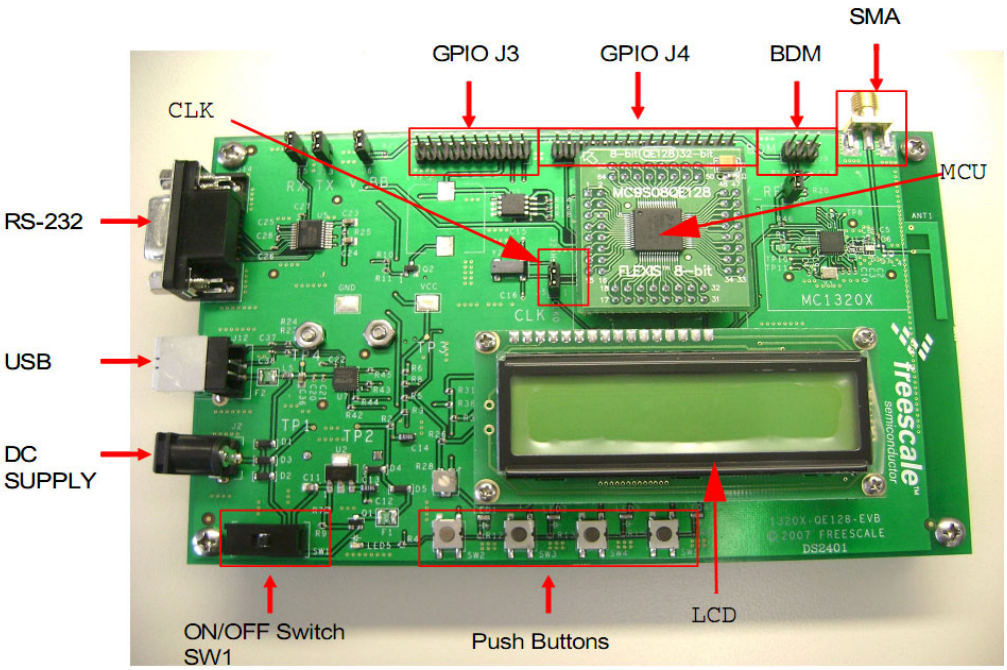

Figure 2-6. 1320x-QE128EVB

Chapter 3
BeeKit and CodeWarrior

The Freescale BeeKit Wireless Connectivity Toolkit is a comprehensive Codebase of wireless networking libraries, application templates, and sample applications. The BeeKit Graphical User Interface, part of the BeeKit Wireless Connectivity Toolkit, allows users to create, modify, and update various wireless networking configurations.

After the user has configured a networking project, BeeKit can generate an XML file that CodeWarrior is able to import. Once CodeWarrior has the project contents, the user can build the target files and load them into the Freescale development boards.

3.1 Creating a BeeKit Project

Follow these steps to create a network project and configure the individual devices.

1. Start BeeKit
2. If another Codebase (MAC or SMAC) is selected, perform the following:
   a) Select File -> Select Codebase… or click the “Select Other Codebase...” link.
   b) Choose the BeeStack Codebase version to use from the Codebase list.
   c) Select OK.
3. From the menu, create a new *project* to configure a new device by selecting the following:
   a) File -> New Project…

The New Project window appears as shown in Figure 3-1.

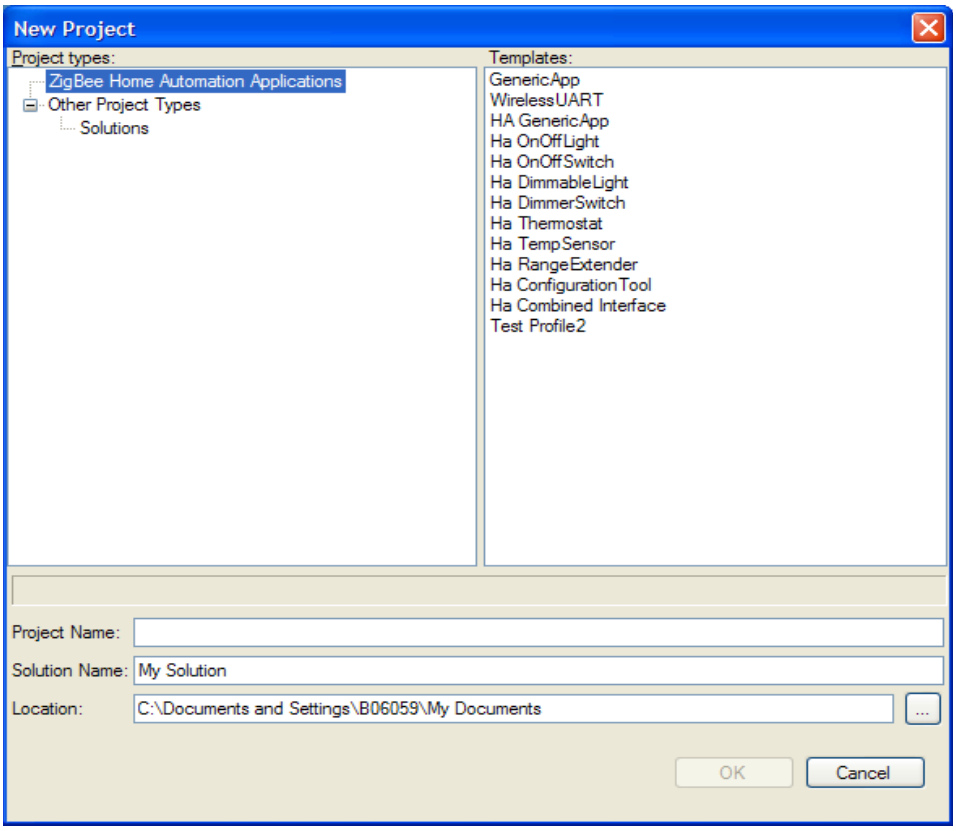

Figure 3-1. BeeKit New Project Window

4. Select the ZigBee Home Automation Applications project type from the left side of the window.

5. As shown in Figure 3-2, select the HaOnOffLight template.

   For the small network being built in this guide, fill in the text boxes for the template application as follows:

   Project name: ZcSrbHaOnOffLight
   Solution Name: HaLightingSolution
   Location: BeeKitSolutions (sub directory on host PC)

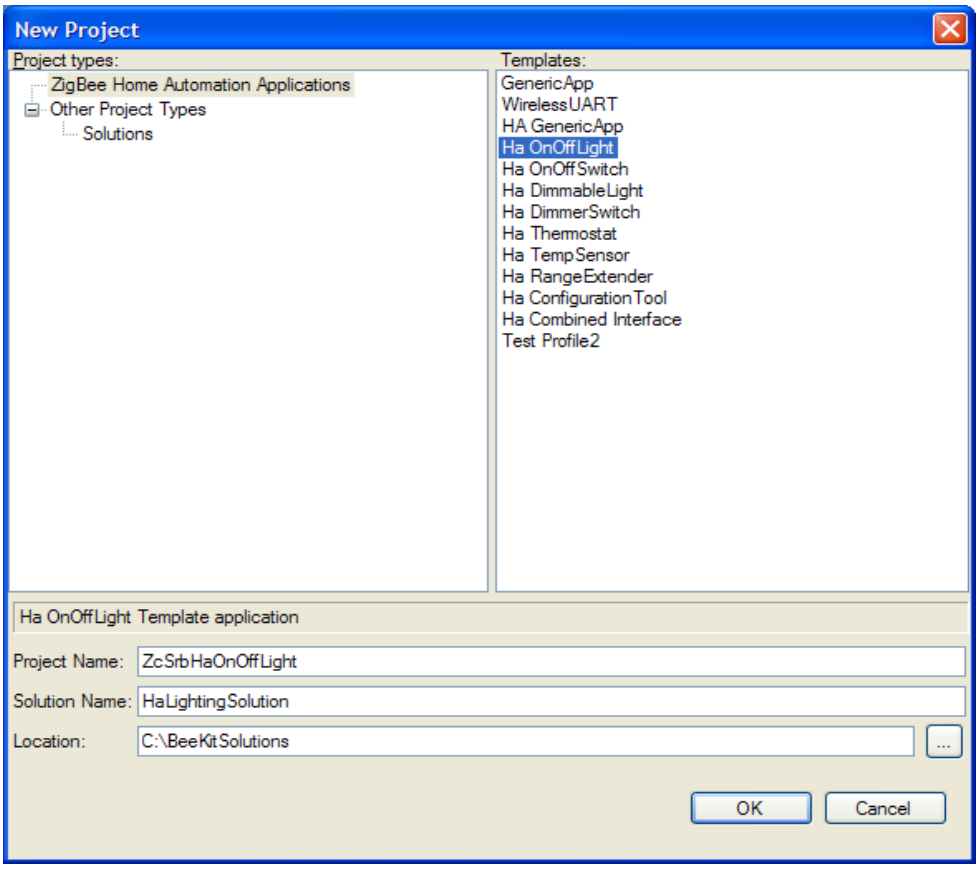

Figure 3-2. Project/Template Select

6. Click the OK button to create the project for the first device.

3.1.1 Basic Options

After the New Project window closes, the BeeKit Project Wizard Welcome window opens as shown in Figure 3-3.

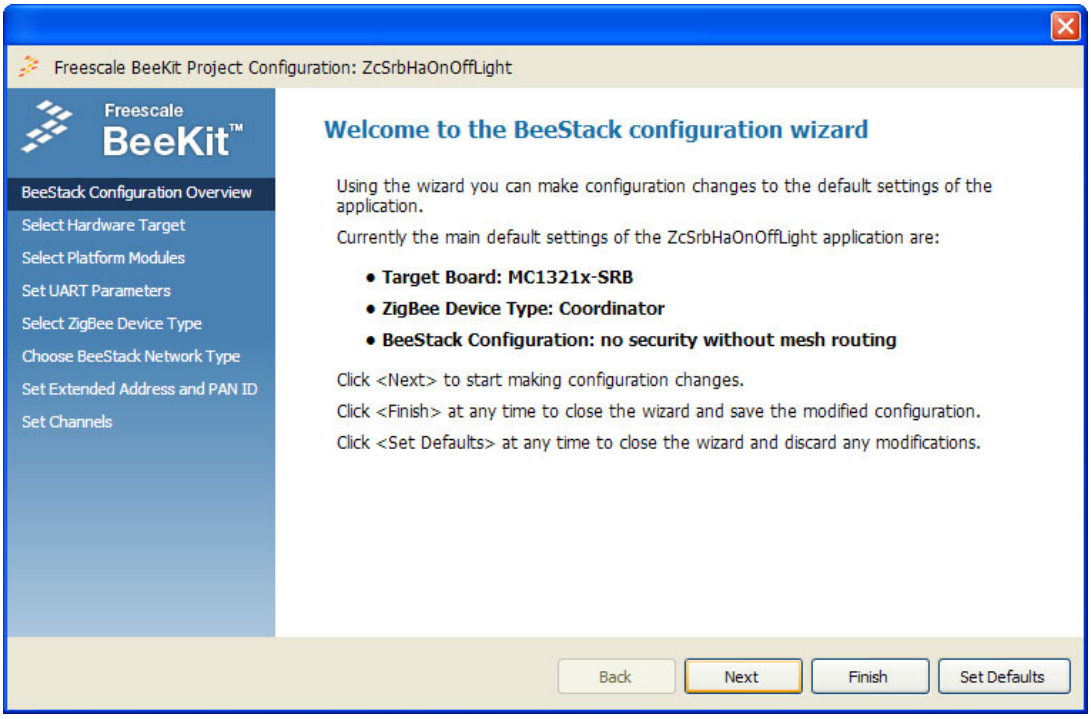

Figure 3-3. BeeKit Project Wizard Welcome Page

There are three ways to proceed from the Wizard welcome window.

Review the current project settings. In this example, the board is an SRB, the ZigBee node type is coordinator, and there is no security or mesh routing enabled. (The default settings depend on the project type.)

- If users accept all of these settings, they can select Finish now without any more configuration
- If users know the settings they want to change, they can go directly to them and select them from the choices on the left
- If users do not know what choices are available, they can click on the Next button. This is the choice described in Section 3.1.2, “Custom Configuration Options”

3.1.2 Custom Configuration Options

Users can modify the device configurations by clicking on the Next button in the Welcome window. The Hardware Target selection page appears as shown in Figure 3-4.

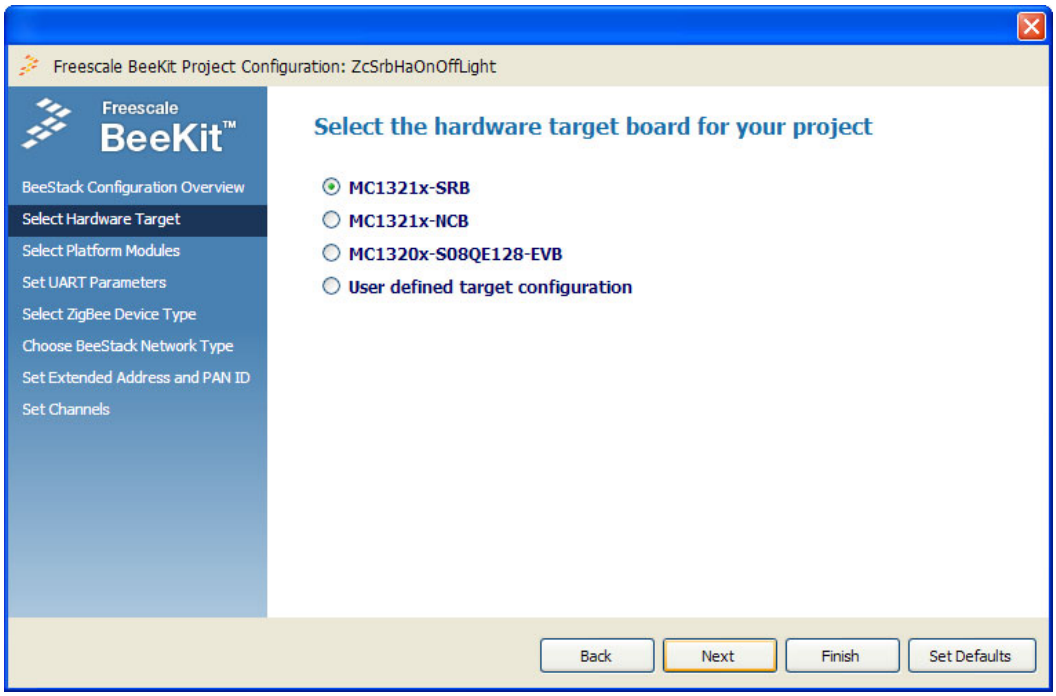

Figure 3-4. Hardware Target Page

1. Change the device to a specific platform. This example uses the MC1321x-SRB.
2. Select the MC1321x-SRB radio button and click on the Next button. The Platform Modules page appears as shown in Figure 3-5.

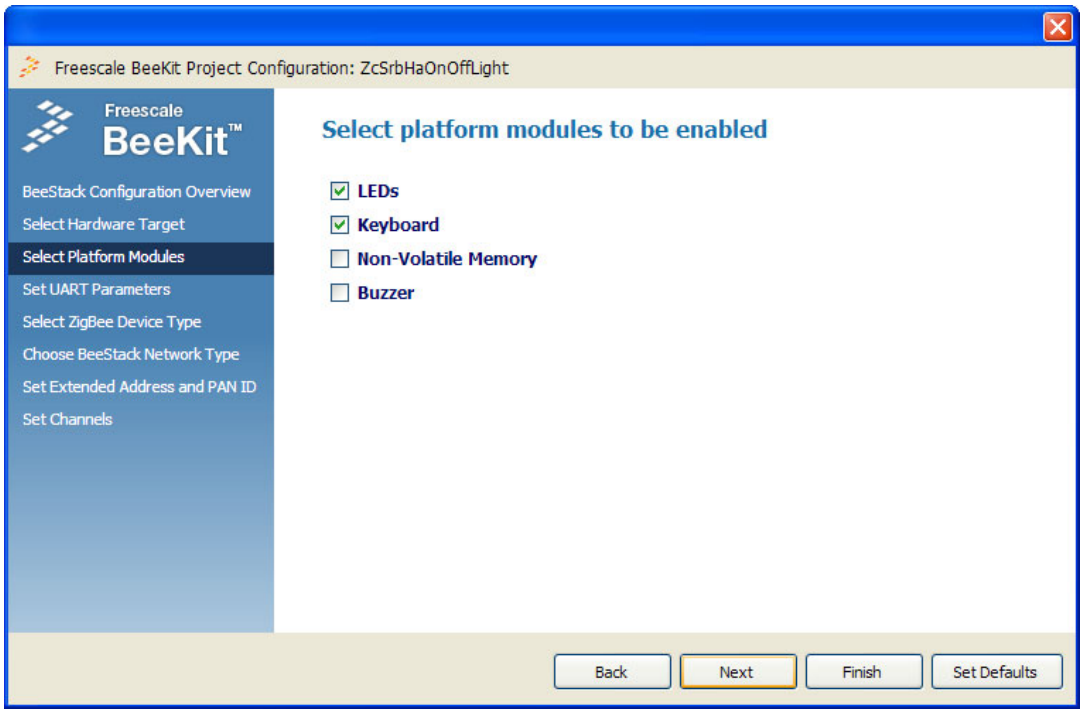

Figure 3-5. Platform Modules Page

4. Leave the default platform modules settings unchanged. LEDs and Keyboard modules on the SRB board will be enabled. Click Next to go to the UART Parameters page shown in Figure 3-6.

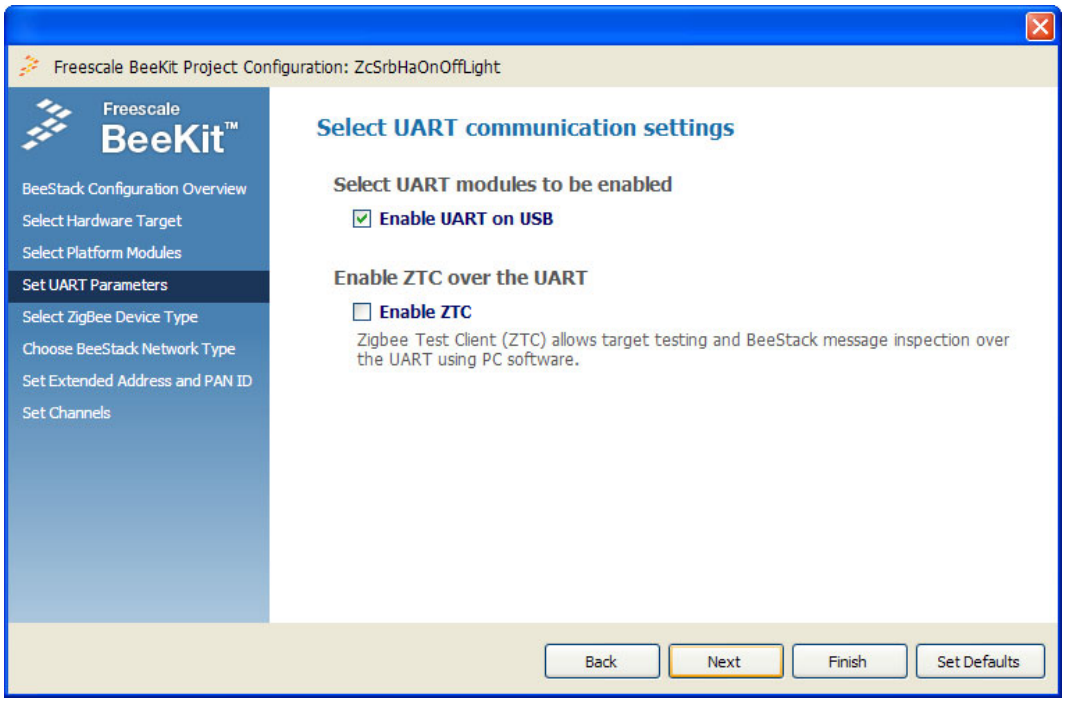

Figure 3-6. UART Parameters Page

6. Leave the default UART settings unchanged. UART module will be enabled on the USB port of the SRB and ZTC will be disabled. Click Next to go to the ZigBee Device Type Selection wizard page shown in Figure 3-7.

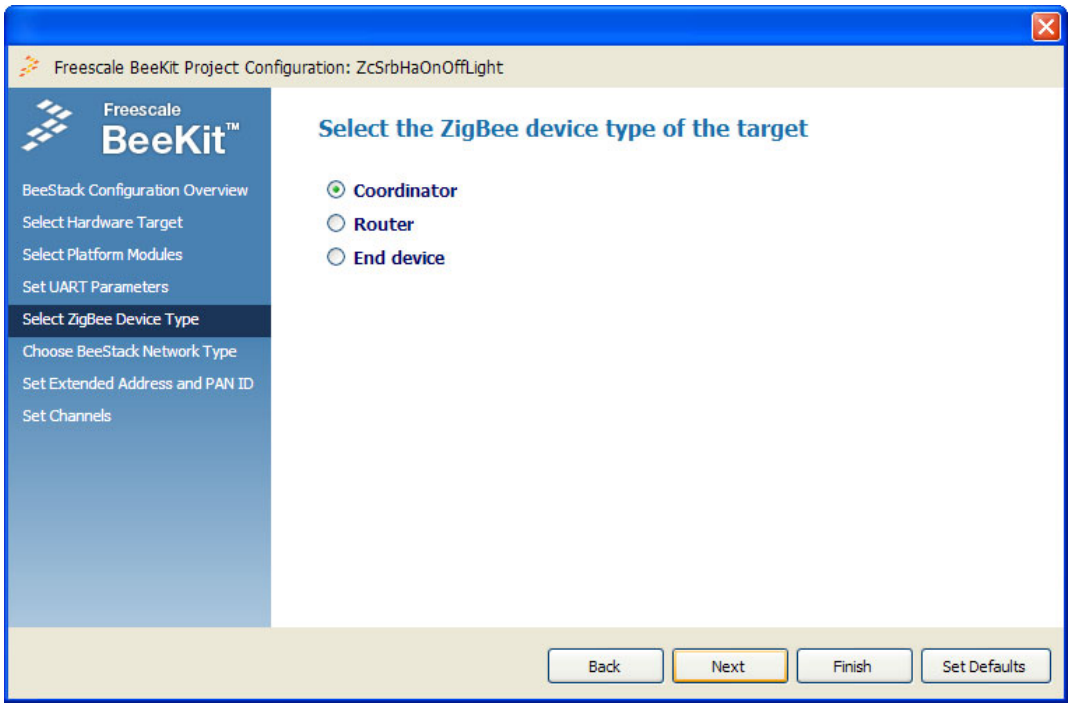

Figure 3-7. ZigBee Device Type Selection Page

7. Make sure the Coordinator is selected as Device type.

NOTE

Each network requires only one coordinator. When repeating these steps for additional devices select the correct Device type (End Device) being configured.

8. Click the Next button. The BeeStack Network Type page appears as shown in Figure 3-8.

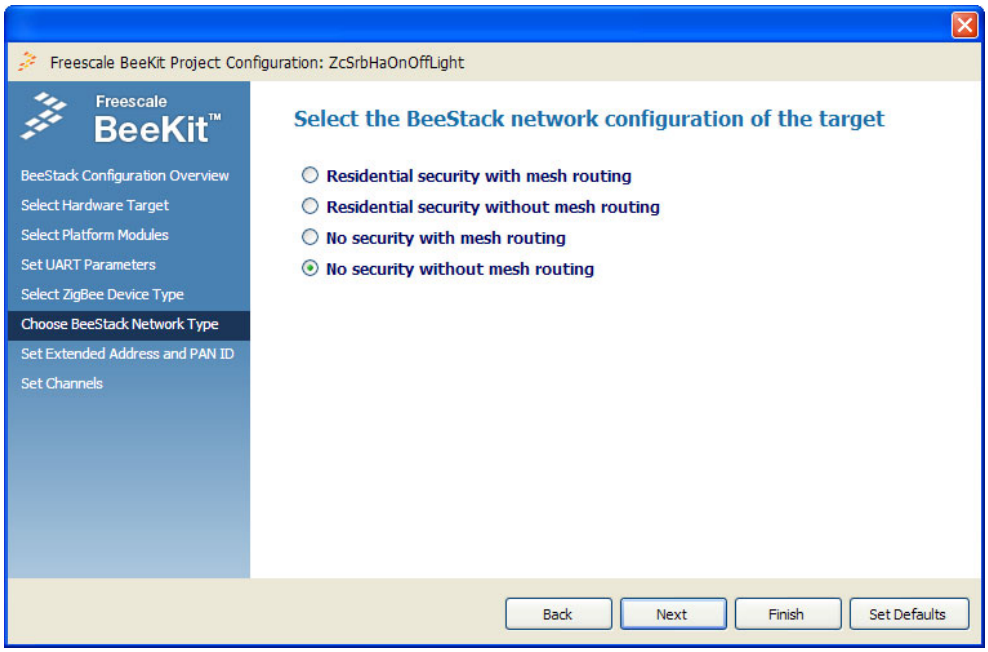

Figure 3-8. BeeStack Network Type Selection Page

9. Make sure “No security without mesh routing” is selected as the BeeStack network configuration.
10. Click Next. The Extended address and PAN ID page appears as shown in Figure 3-9.

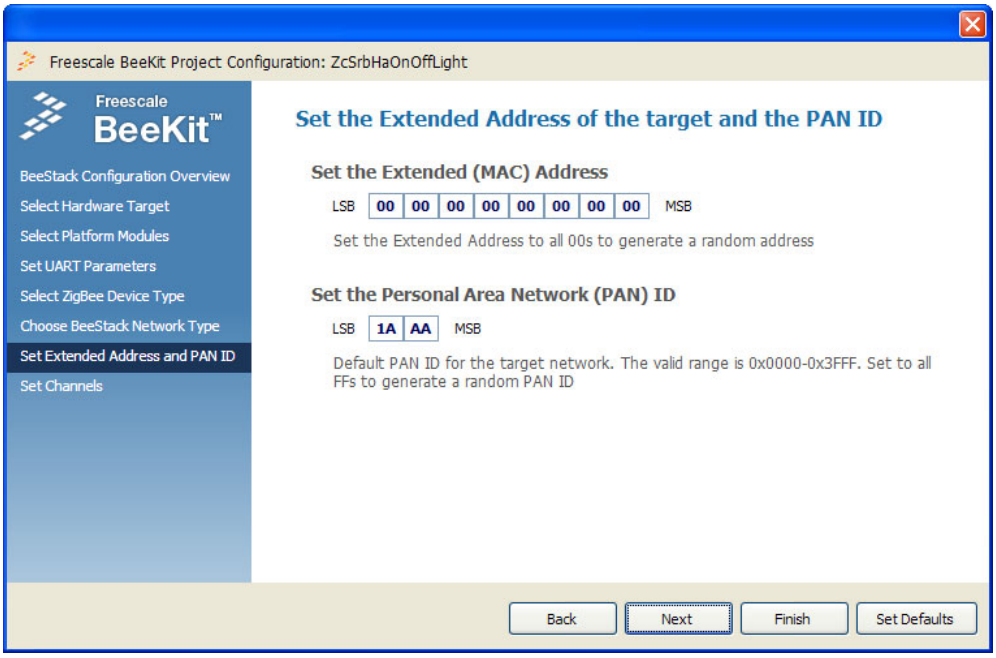

Figure 3-9. Extended Address and PAN ID Selection Page

11. For this example, leave the Extended Address option with all zeros. BeeKit will automatically generate a random (but not guaranteed unique) address. Alternatively, enter the full MAC address from the label on the development board. Also leave the default PAN ID set to 0x1AAA.
12. Click Next. The Channels page appears as shown in Figure 3-10.

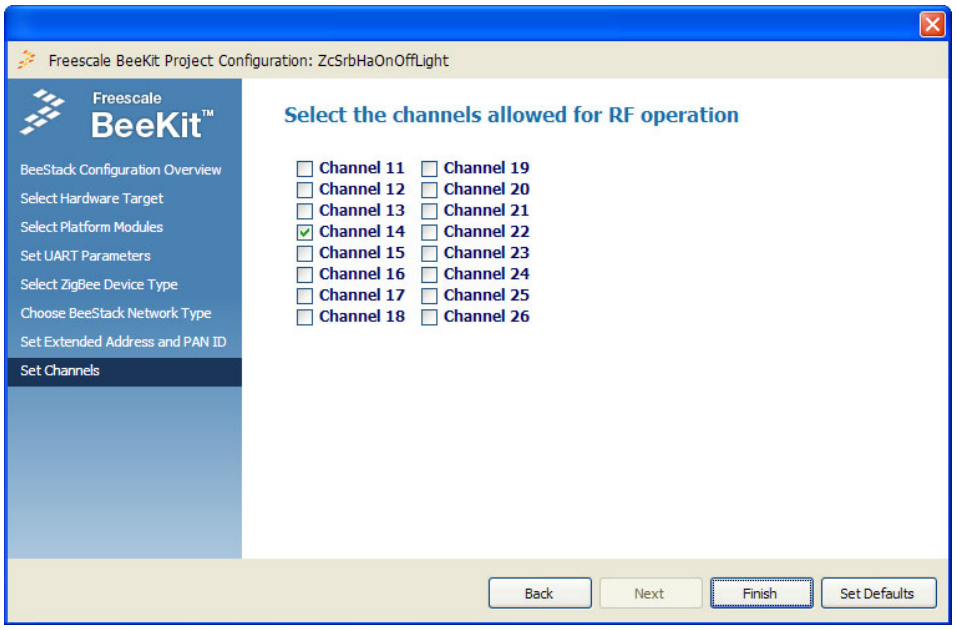

Figure 3-10. Channels Page

WARNING

The channel and PAN ID must be the same for all devices on the network. If users set the channel to something other than the default, verify in the setup for subsequent devices that the channel is the same one selected for this initial device. If users select a PAN ID, a colleague might assign the same value to another PAN later. Expect odd behavior. If users choose 0xFFFF for their PAN ID, the coordinator selects one when it forms its network. The nodes that join this network must be able to identify the correct network by some method that does not involve a known PAN ID. This is not a BeeKit issue; this is a real-world ZigBee issue. User products must be able to cope with this.

13. Click on the Finish button.

The setup concludes when BeeKit returns to the main Project window.

3.1.3 Creating Additional Devices

To set up a ZigBee wireless network, the coordinator needs other devices to communicate with and control. Create additional devices by performing the following steps:

1. From the main menu, Select Solution -> Add Project…
2. This opens the Add Project window.
3. Select ZigBee Home Automation Applications the same as was done for the coordinator. This time, select the HA OnOffSwitch.
4. Repeat the steps as described in Section 3.1.1, “Basic Options” and Section 3.1.2, “Custom Configuration Options”. Adjust the settings to suit the application. Note that the current settings for the switch are different than the light.
5. After selecting Finish for the switch application, the BeeKit main window appears as shown in Figure 3-11.

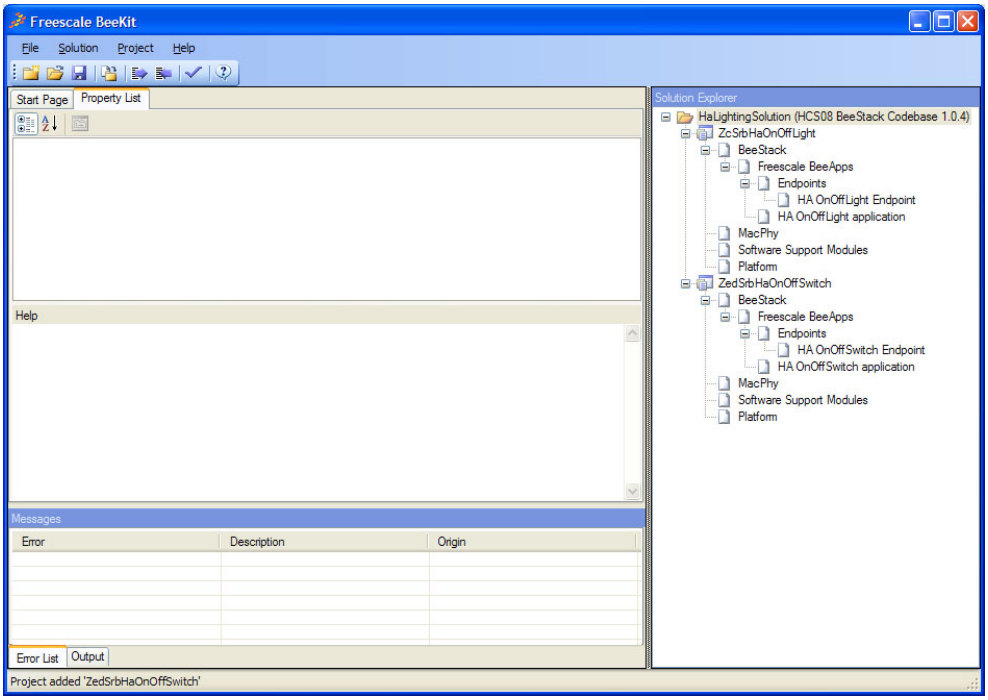

Figure 3-11. BeeKit Main Window

When users click on a component in the Solution Explorer area, useful information about that item appears under the Property List tab. Users can also change their settings there. For example,

- To change the development board:
  - — Select the Platform component in the Explorer window
  - — Select the Hardware Target property in the Property List
  - — Use the pull-down menu to select the board
- To change the device type:
- Select the MacPhy component in the Explorer window
  - — Select the Device Type property in the Property List

— Use the pull-down menu to select coordinator, router, or end device

3.1.4 Exporting Created BeeKit Projects

Once all the devices to be used in the network have been created as BeeKit Projects and saved as a BeeKit Solution, the files must be exported into a format for importing into the CodeWarrior IDE used for compiling and debugging.

To export the saved solution:

1. From the Solution menu, select Export Solution. BeeKit verifies the internal consistency of the configuration, looking for such errors as two endpoints with the same number. If the verification succeeds, the window that opens will display all the created devices, each with a checked box as shown in Figure 3-12.

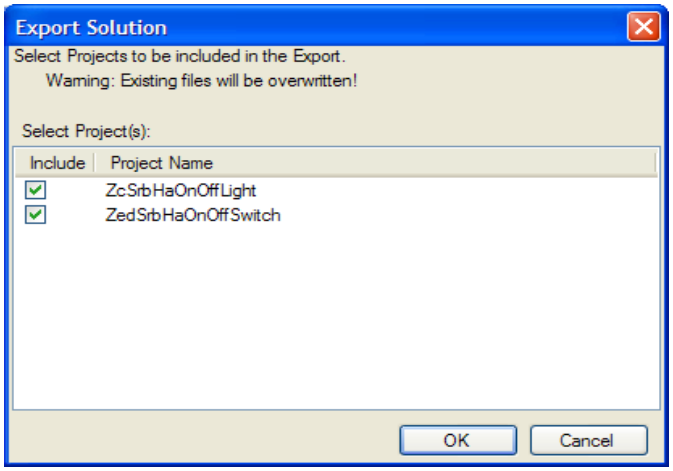

Figure 3-12. Export BeeKit Project Solution Window

2. Click on the OK button to start the export process. While BeeKit executes this, it displays the window shown in Figure 3-13, and the steps it takes scroll by in the Messages window pane.

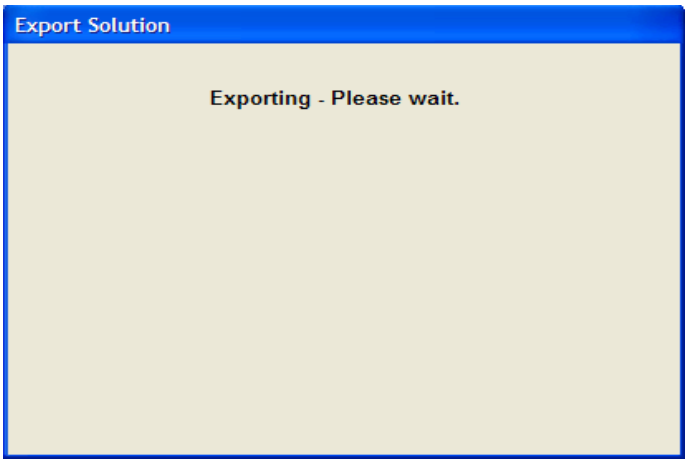

Figure 3-13. Please Wait – Exporting Project Window

3. When BeeKit finishes exporting, the Wait window disappears. The Messages window still contains the export steps, which can be scrolled through. The export process is now complete.
4. Exit BeeKit by choosing File -> Exit from the menu bar.

3.2 Importing the Project into CodeWarrior

The project files created in BeeKit and saved as Solutions must now be imported into CodeWarrior to build the binary output file suitable for programming into each MCU's flash.

To import the code for each device, follow these steps:

1. Start *CodeWarrior*, which opens to a blank window and the menu at the top.
2. Select File -> Import Project… as shown in Figure 3-14.

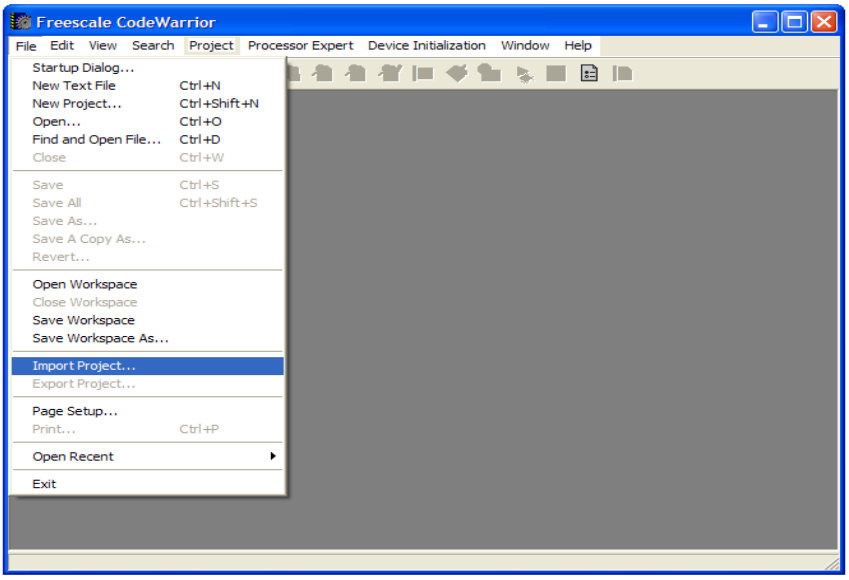

Figure 3-14. CodeWarrior Import Project…

3. Navigate to the directory created in the BeeKit export procedure. In this example, it is:

```
C:\BeeKitSolutions\HaLightingSolution
```

4. This directory contains a directory for each device created earlier in BeeKit. In this example, the directories are `ZcSrbHAOnOffLight` and `ZedSrbHAOnOffSwitch`. Users will need to import each device's project separately
5. Navigate into the `ZcSrbHAOnOffLight` directory as shown in Figure 3-15.

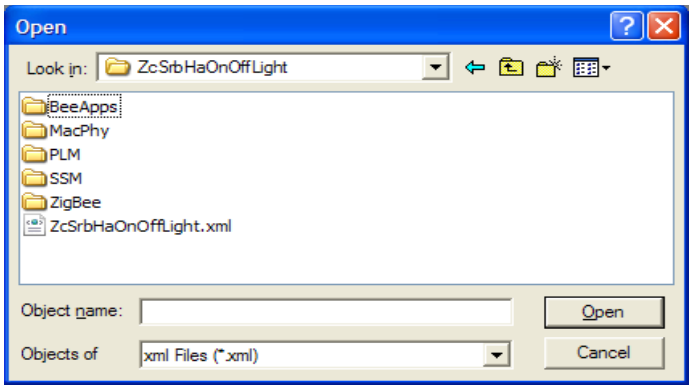

Figure 3-15. CodeWarrior Open XML Window

6. Click on the BeeKit `.xml` file to be imported and select Open. The Open window is replaced by a Save window in the same directory. Type in the name for the CodeWarrior project file without an extension. This example uses the same name as the `.xml` file as shown in Figure 3-16. Click the Save button.

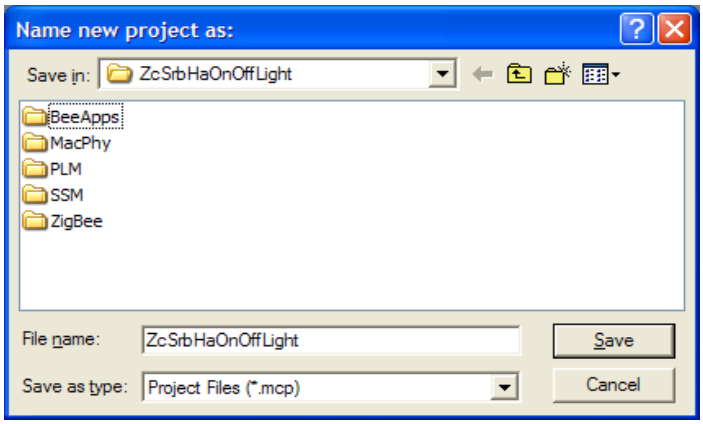

Figure 3-16. CodeWarrior Name New Project Window

7. CodeWarrior creates and loads the project file as shown in Figure 3-17.

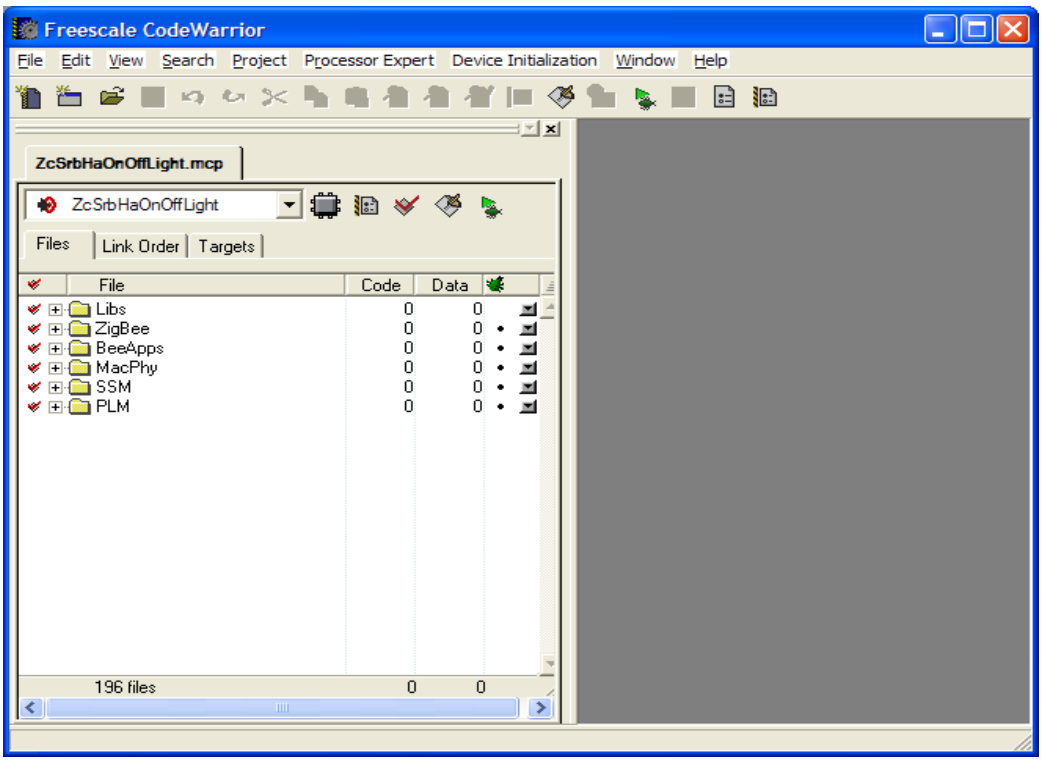

Figure 3-17. CodeWarrior ZcSrbHAOnOffLight Project Window

3.3 Building a Code Image Using CodeWarrior

Build the application binary in one of three ways:

1. Press the Make hot key F7
2. Click the “make” icon (Looks like paper with a pen in hand.)
3. From the menu select Project -> Make

CodeWarrior reports the build progress in a window it opens and then closes as shown in Figure 3-18.

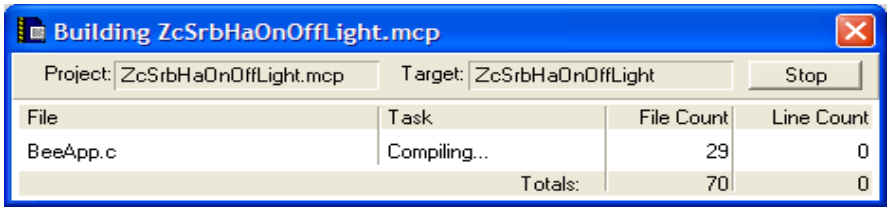

Figure 3-18. CodeWarrior Build Window

3.4 Loading the Code Image into a ZigBee Device

Connect the P&E BDM pod to the host computer using the USB cable. A lighted blue LED indicates the BDM has power and a successful USB connection.

1. Connect the BDM pod to the device. Align pin 1 of the BDM port connector with the red wire of the flat cable connector
   - EVB, SARD, and SRB: The connector is J101, and pin 1 is marked with a ‘1’
   - NCB: The connector is marked “BDM”, and pin 1 has a white dot beside it
   - 1320x-QE128EVB: The connector is labeled J9, and pin 1 is the one next to the label
2. Turn on the board.
3. The amber LED on the BDM will now light up, and the LED on the development board will also light up. If not, switch the power off and on again on the board. Recheck the connection to the BDM port. Check the power adapter connection to the board or verify that the batteries are charged.
4. Download the compiled image to the board by choosing one of three ways
   a) Press the Debug hot key F5
   b) Click the “Debug” icon (Looks like a bug with a green triangle.)
   c) From the menu select Project -> Debug

CodeWarrior opens the Debug window as shown in Figure 3-19.

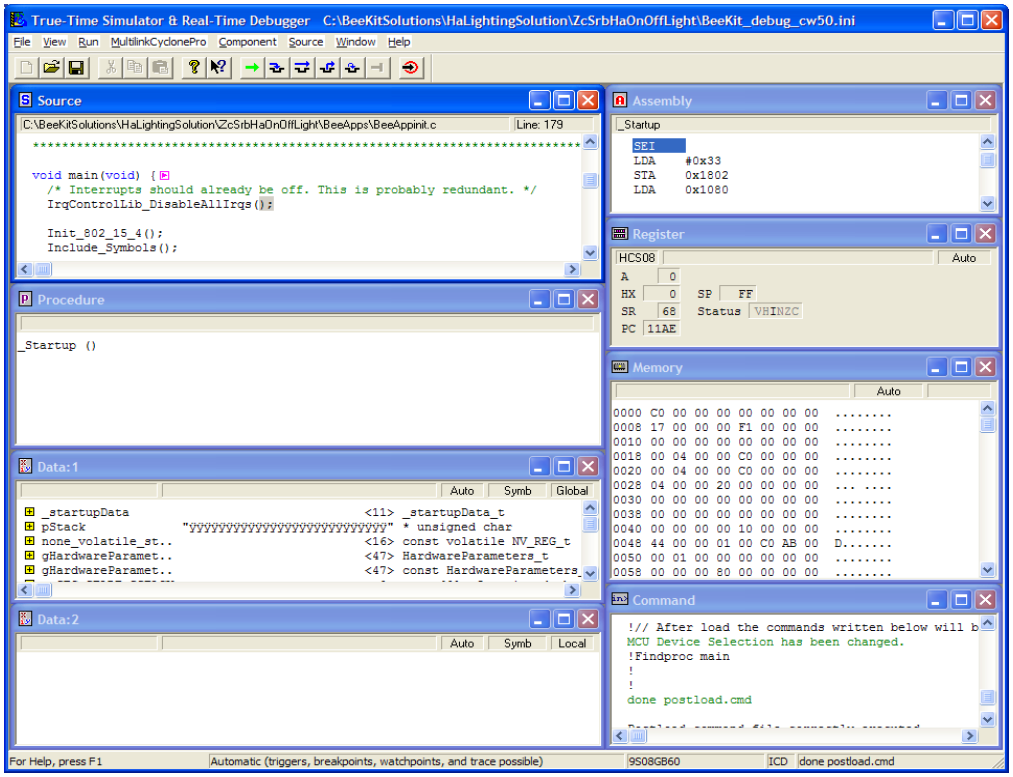

Figure 3-19. CodeWarrior Debugger Window

CodeWarrior then opens the connection manager window, if this is the first programming/debug event since it was opened, as shown in Figure 3-20.

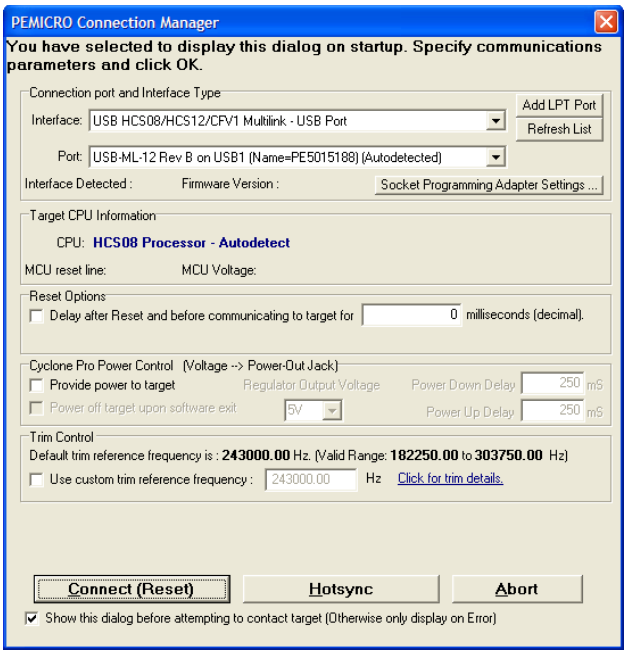

Figure 3-20. CodeWarrior Connection Manager Window

5. Click on the Connect button. The programmer window displays the running status in a window as shown in Figure 3-21 and then closes automatically.

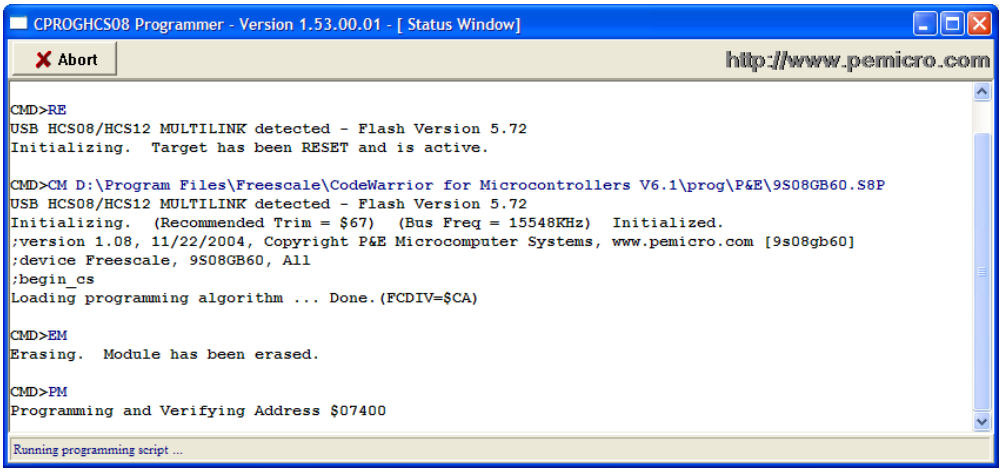

Figure 3-21. Status Window for Connection Manager

NOTE

If the Status window does not display the line “programming and verifying Address,” either the BDM or USB cables are not properly connected. Correct the problem before repeating the step.

6. Close the debugger by selecting File -> Exit.

WARNING

Exit the debug mode to avoid multiple debug appearances; having multiple appearances while setting up boards can produce unexpected results.

7. Disconnect the board and exit the CodeWarrior project, leaving CodeWarrior running.
8. Power cycle the board or press the reset button to get the board ready for use.
9. Repeat these steps to put the ZedSrbHAOnOffSwitch code into another board.

Chapter 4
Starting and Running a Simple ZigBee Network

This chapter goes through the steps to establish a small Home Automation network using the two nodes programmed in the previous chapters. The network consists of the HA OnOffLight and the HA OnOffSwitch.

NOTE

When referring to switches SW1-SW4 in this guide, it refers to the four push-buttons on the development boards. On the 1320x-QE128EVB the buttons are labeled SW2-SW5 and on the SARD they are labeled S101-S104. See Section 2.1, “HCS08 Board Details”.

4.1 Starting the Network

1. Turn on the power for the OnOffLight.

   LED1 will flash to indicate that the board is not on a network (if a ZR or ZED) or that a network has not been formed (ZC).
2. Press SW4 to select a channel other than the one selected in BeeKit (optional).

   With each key press, the four LEDs will light up briefly to indicate the channels from 11 to 26. The LEDs display the offset from channel 11 in binary: 0000 is channel 11, and 1111 is channel 26.
3. Repeat steps 1 and 2 for the OnOffSwitch. The network will not form if the two nodes are on different channels after step 2, because the example applications are configured to look on only one channel for a network.

Start the network with the following steps:

1. Press SW1 on the OnOffLight, which is configured as the ZC.

   The LEDs 1-4 will light up in sequence and then go out until only LED1 remains lighted. This is the first step in creating a small network, since this step serves to *form* the network.
2. Press SW1 on the OnOffSwitch, which is configured as the ZED.

   All the LEDs on this board will flash in a series, and finally only LED1 will light to indicate that ZED has joined the network that the OnOffLight has formed.
3. Press SW3 first on one board, then within ten seconds, press SW3 on the other board, to bind the devices.

   LED3 starts flashing, then goes solid when binding completes. If LED3 flashes, but then goes out (unlighted), binding failed. Turn off the boards and try again.
4. Long press SW1 to go into run mode on both boards. Holding a switch down for more than one second is a long press.

4.2 Running the Network: Remotely Controlling a Light

1. Press SW1 on the board defined as the ZED (OnOffSwitch) to toggle the light.
2. LED2 on the ZC (OnOffLight) changes state: it turns on.
3. Press SW1 on the ZED again to toggle the light.
4. LED2 on the ZC changes state again: it turns off.
5. SW1 on the ZC can also toggle the light. Press it and ZC's LED2 changes state
6. Press SW1 on the ZED again to toggle the light again.

This completes setting up and using a small Home Automation Lighting Control network. For additional example applications, see Chapter 5, "Creating a Wireless UART application".

Chapter 5
Creating a Wireless UART application

This chapter shows how to create a Freescale Wireless UART application using the Freescale BeeKit Wireless Connectivity Toolkit. The Freescale Wireless UART application is not part of the ZigBee Alliance Home Automation (HA) profile and is not part of any other ZigBee public profiles. The Wireless UART application is considered by the ZigBee Alliance (www.zigbee.org) as a “Manufacturer Specific” profile, in that these applications are only meant to function with devices from a single manufacturer.

The Wireless UART runs in a fully buffered interrupt driven mode and is well suited for large file transfers. However, the example used in this chapter only sends a small amount of text to show Wireless UART basic functionality.

5.1 Creating the Wireless UART BeeKit Project

Follow these steps to create a BeeKit project and configure the Coordinator and End Node devices.

1. Start the BeeKit Wireless Connectivity Toolkit.
2. If another Codebase (MAC or SMAC) is selected, perform the following:
   a) From the tool bar select File -> Select Codebase… or
   click the “Select Other Codebase...” button.
   b) From the Codebase list, choose a BeeStack Codebase version.
   c) Click the OK button.
3. From the menu, create a new project to configure a new device by selecting File -> New Project… The New Project window appears as shown in Figure 5-1.

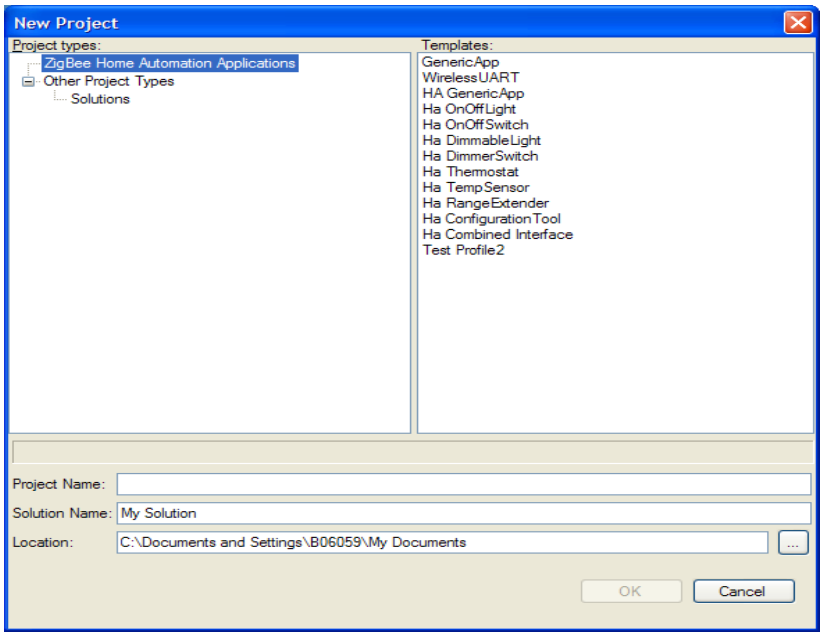

Figure 5-1. BeeKit New Project Window

4. Select the ZigBee Home Automation Applications project type from the left side of the window.
5. As shown in Figure 5-2, select the Wireless UART template.
6. For the small network being built in this guide, fill in the text boxes for the template application as follows:

   Project name: ZcSrbWirelessUART
   Solution Name: WirelessUART Solution
   Location: c:\WirelessUART (or other sub directory on host PC)

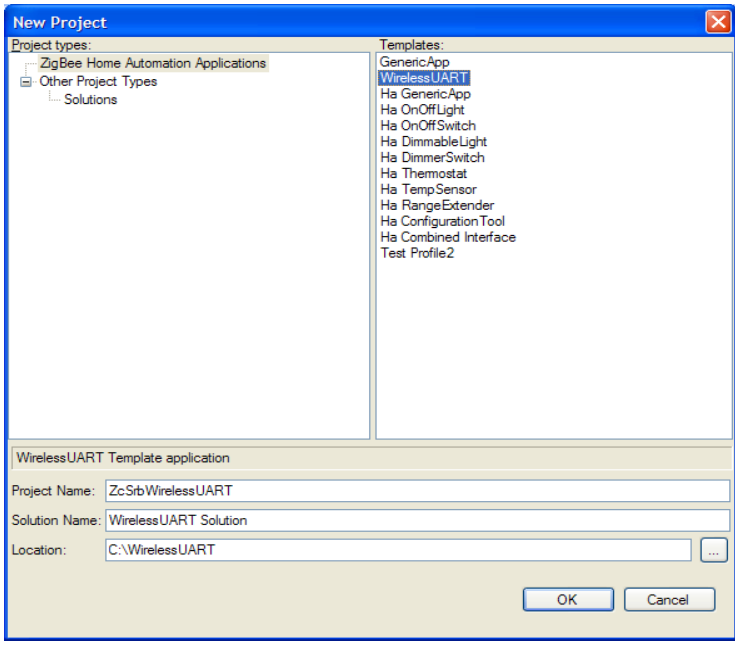

Figure 5-2. Project/Template Select

7. Click the OK button to create the project for the first device.Basic Options

After the New Project window closes, the BeeKit BeeStack Configuration Overview window appears as shown in Figure 5-3.

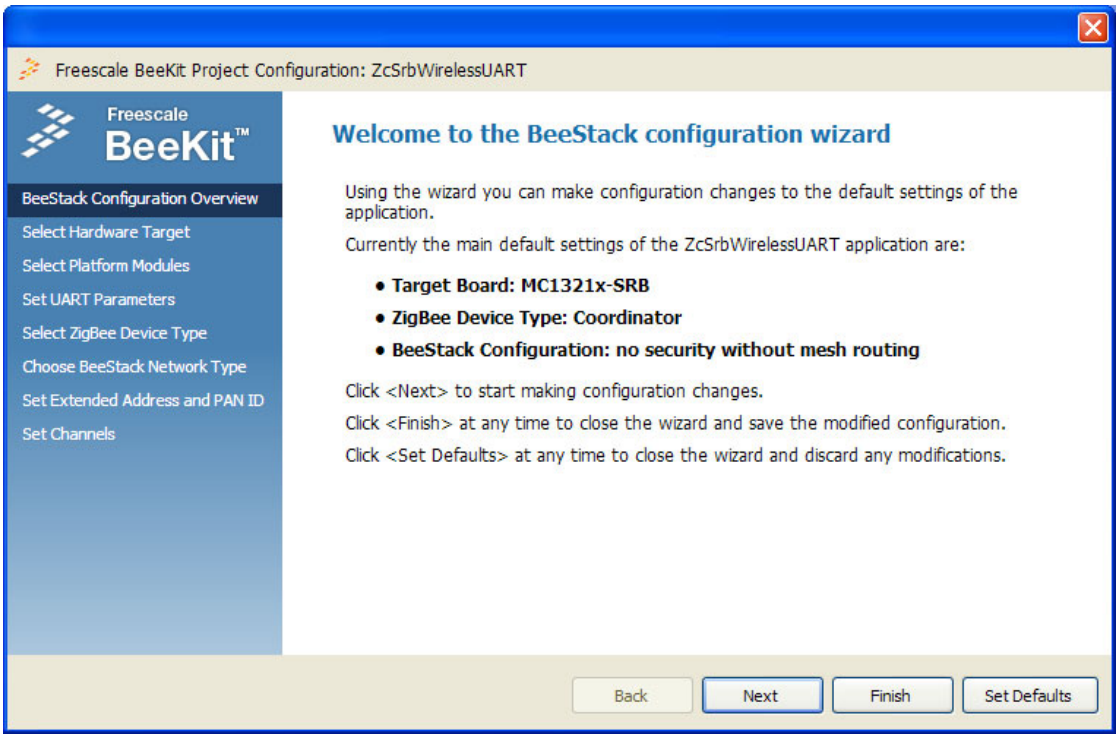

Figure 5-3. BeeKit BeeStack Configuration Overview Window

There are three ways to proceed from this window.

Review the current project settings. In this example, the board is an SRB, the ZigBee node type is coordinator, and there is no security or mesh routing enabled. (The default settings depend on the project type.)

- If users accept all of these settings, they can select Finish now without any more configuration
- If users know the settings they want to change, they can go directly to them and select them from the choices on the left
- If users do not know what choices are available, they can click on the Next button. This option is described in Section 5.1.1, “Custom Configuration Options”

5.1.1 Custom Configuration Options

After clicking the Next button in BeeKit BeeStack Configuration Overview window, the Select Hardware Target window appears as shown in Figure 5-4.

1. Change the device to a specific platform. This example uses the MC1321x-SRB.

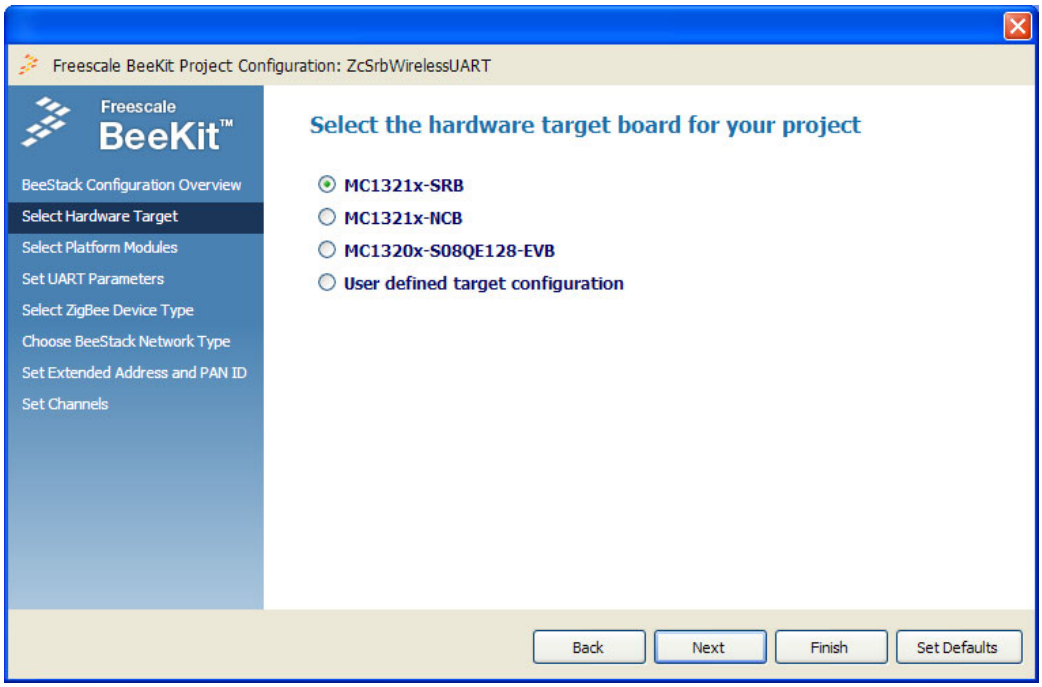

Figure 5-4. Hardware Target Page

2. Select the MC1321x-SRB option and click on the Next button. The Select Platform Modules window appears as shown in Figure 5-5.

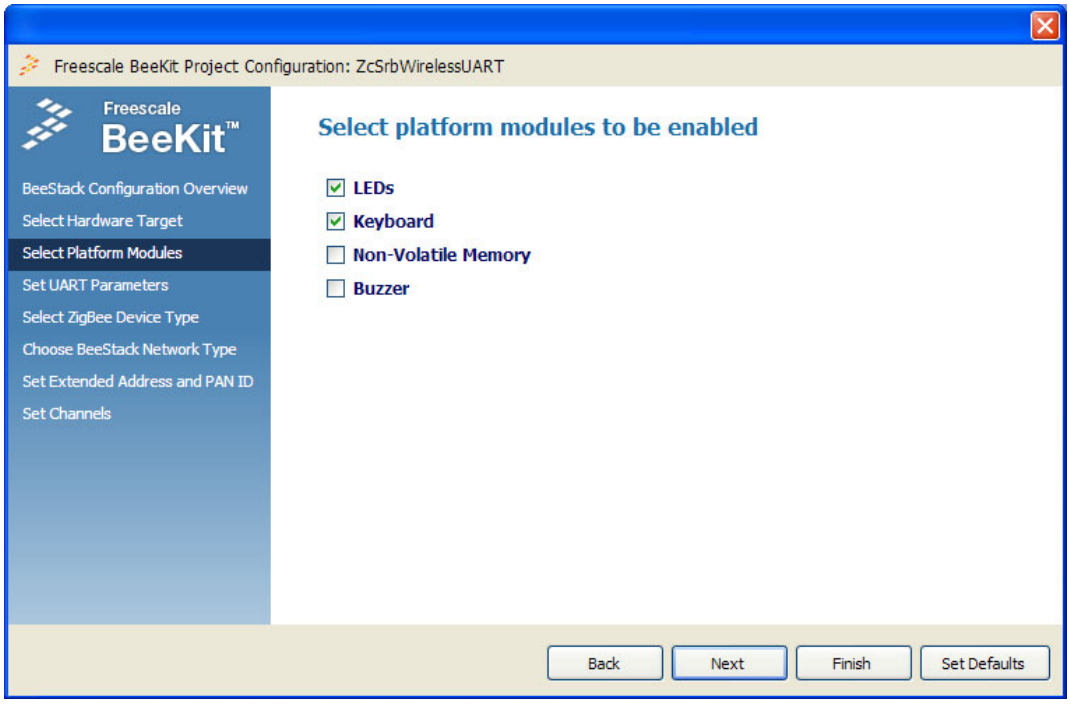

Figure 5-5. Platform Modules Page

4. Leave the default platform modules settings unchanged. (LEDs and Keyboard modules on the SRB board enabled.) Click the Next button and the Set UART Parameters window appears as shown in Figure 5-6. Leave the default UART settings unchanged. (The UART module enabled on the USB port of the SRB and the ZigBee Test Client (ZTC) disabled.)

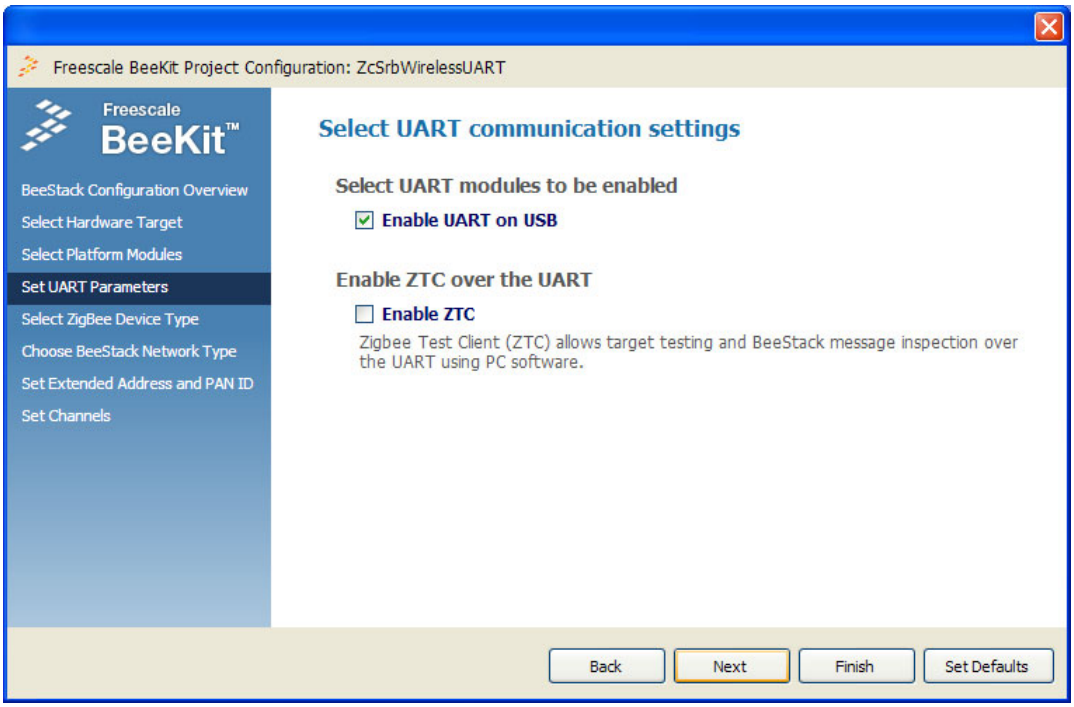

Figure 5-6. UART Parameters Page

6. Click the Next button and the Select ZigBee Device Type window appears as shown in Figure 5-7. Ensure that the “Coordinator” option is selected.

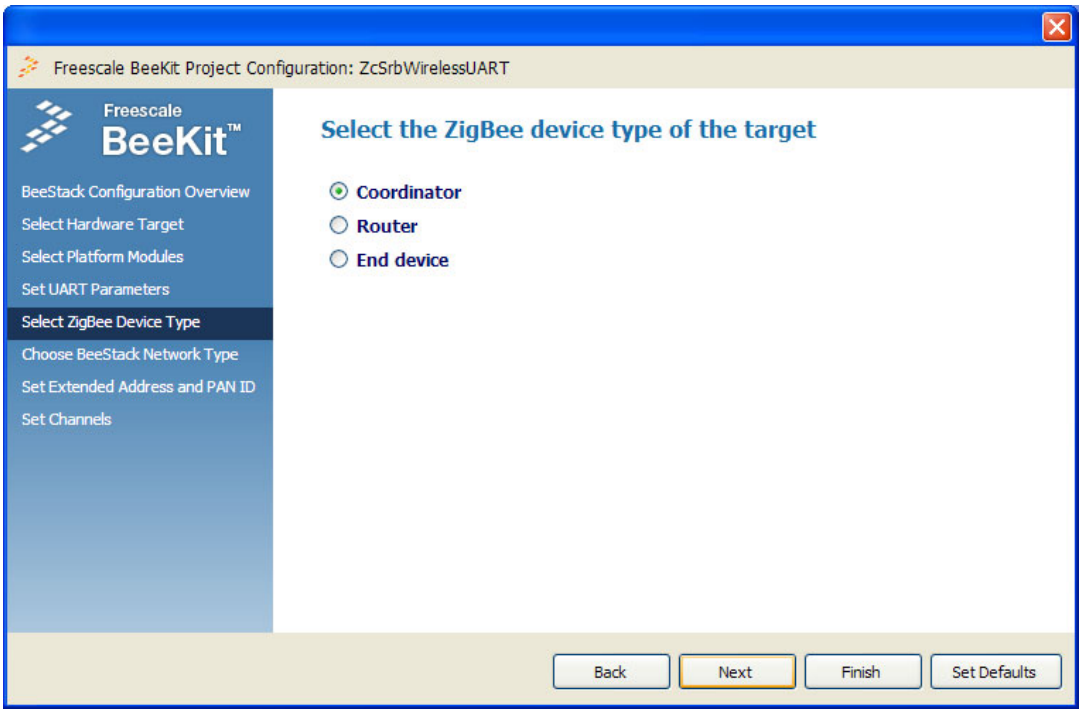

Figure 5-7. ZigBee Device Type Selection Page for Coordinator Device

7. Click the Next button and the Choose BeeStack Network Type window appears as shown in Figure 5-8. Select the “No security without mesh routing” option.

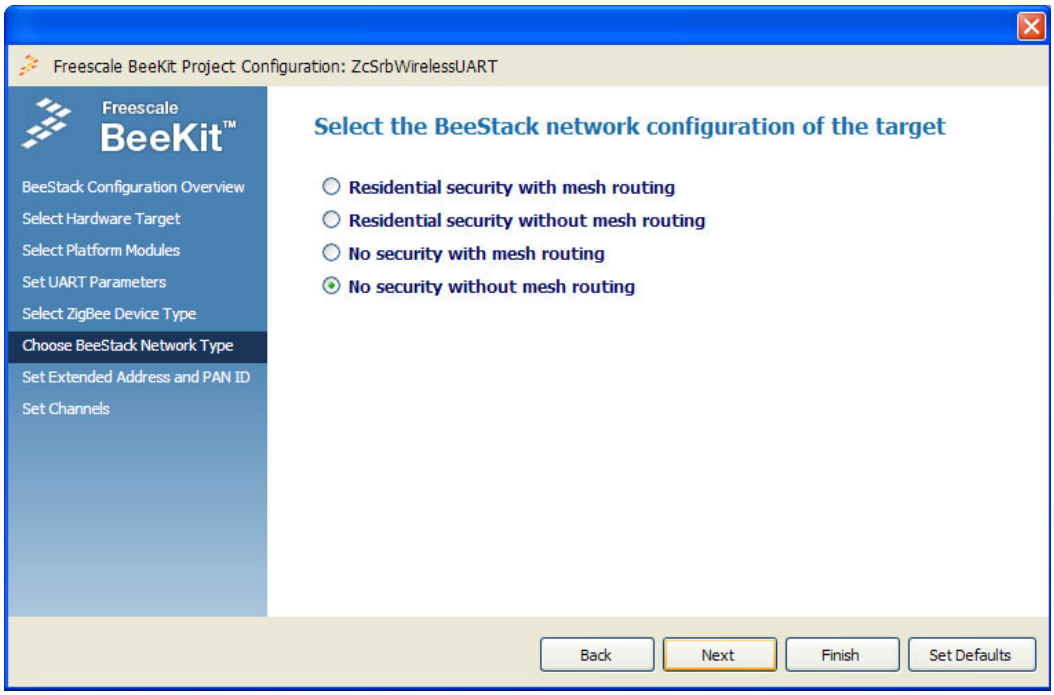

Figure 5-8. BeeStack Network Type Selection Page

8. Click the Next button and the Set Extended Address and PAN ID window appears as shown in Figure 5-9. For this example, leave the Extended Address option with all zeros. BeeKit automatically generates a random (but not guaranteed unique) address. Alternatively, enter the full MAC address from the label on the development board. Leave the default PAN ID set to 0x1AAA.

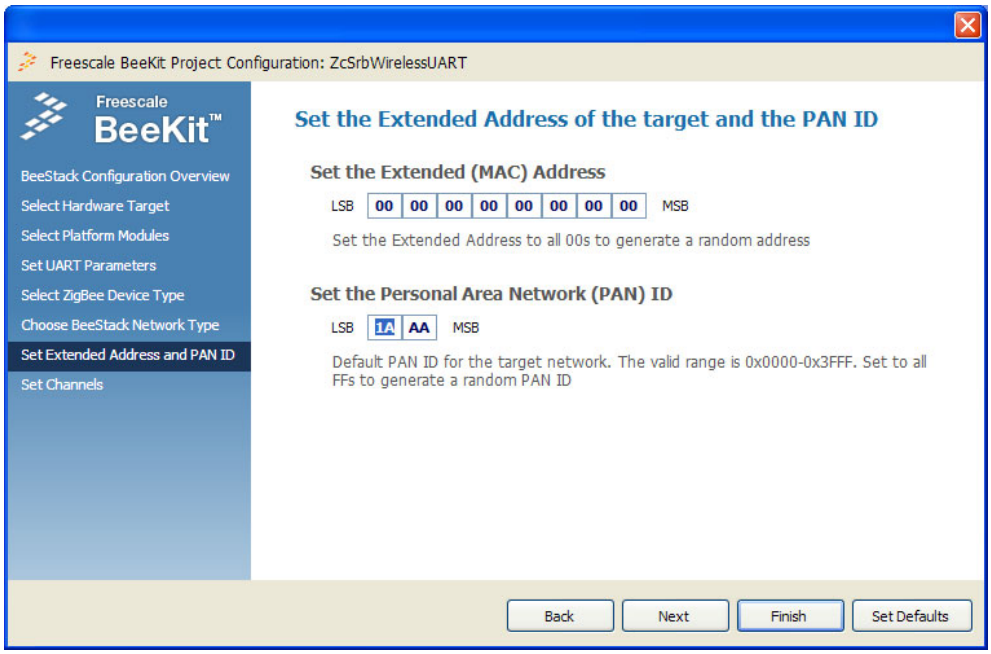

Figure 5-9. Extended Address and PAN ID Selection Page

WARNING

The channel and PAN ID must be the same for all devices on the network. If users set the channel to something other than the default, verify in the setup for subsequent devices that the channel is the same one selected for this initial device. If users select a PAN ID, a colleague might assign the same value to another PAN later. Expect odd behavior. If users choose 0xFFFF for their PAN ID, the coordinator selects one when it forms its network. The nodes that join this network must be able to identify the correct network by some method that does not involve a known PAN ID. This is not a BeeKit issue; this is a real-world ZigBee issue. User products must be able to cope with this.

9. Click the Next button and the Set Channels window appears as shown in Figure 5-10. Select ‘Channel 14’.

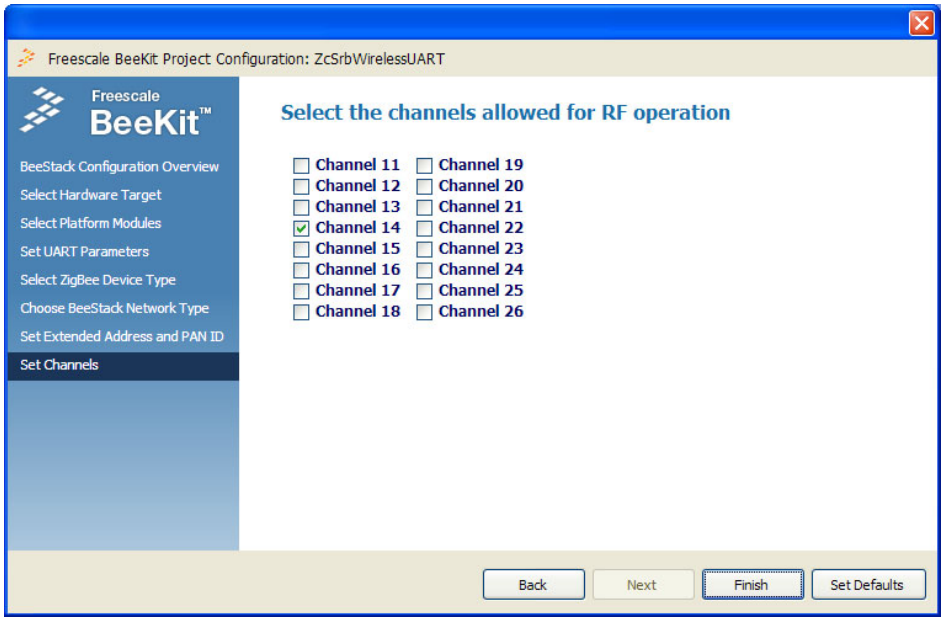

Figure 5-10. Channels Page

10. Click the Finish button.

5.1.2 Adding a Project for the End Node

Now that the coordinator is added, users need to add a project for the End Node. To add a project for the End Node using BeeKit, perform the following tasks.

1. From the BeeKit main window, select Solutions -> Add Project. The Add Project window appears as shown in Figure 5-11.

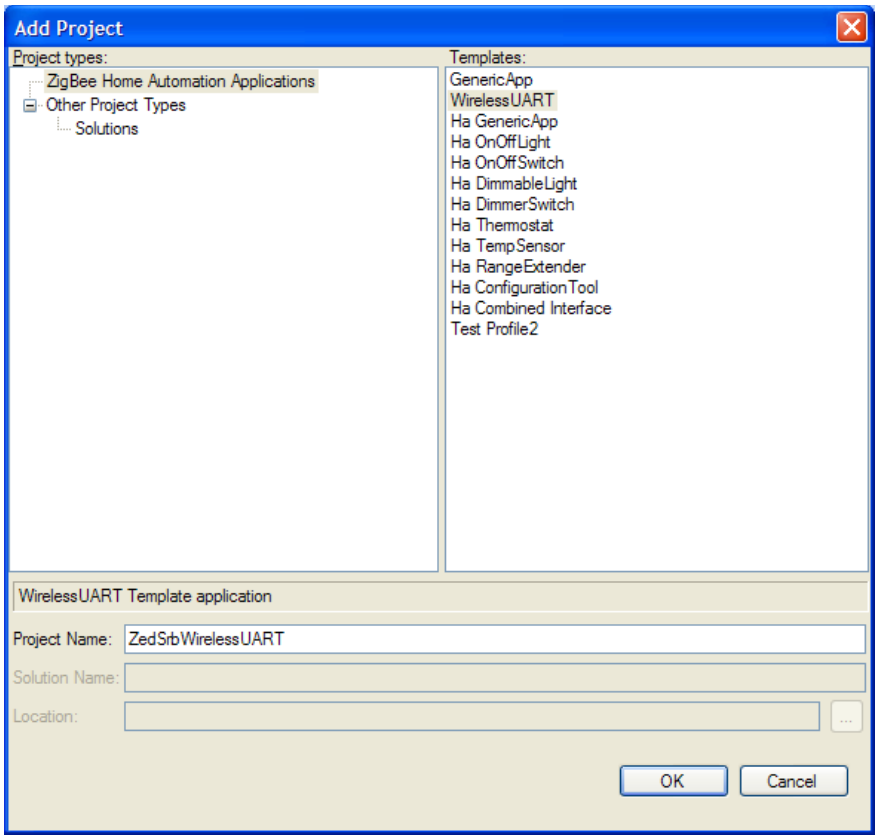

Figure 5-11. Add New Project Page

2. Select “WirelessUART” and name the project WirelessUART_EndPoint as shown in Figure 5-11.
3. Continue clicking the Next button until reaching the “Select ZigBee Device Type” window as shown in Figure 5-12. Select the “End device” option and click on the Next button.

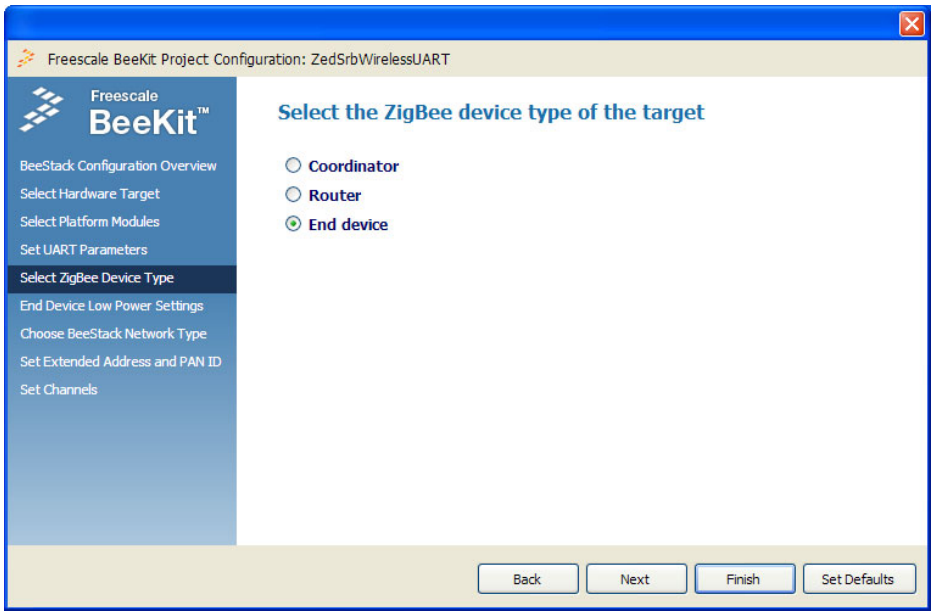

Figure 5-12. Selecting the End Device as the Device Type

4. The End Device Low Power Settings window appears as shown in Figure 5-13. Leave all settings at their default values as shown in Figure 5-13.

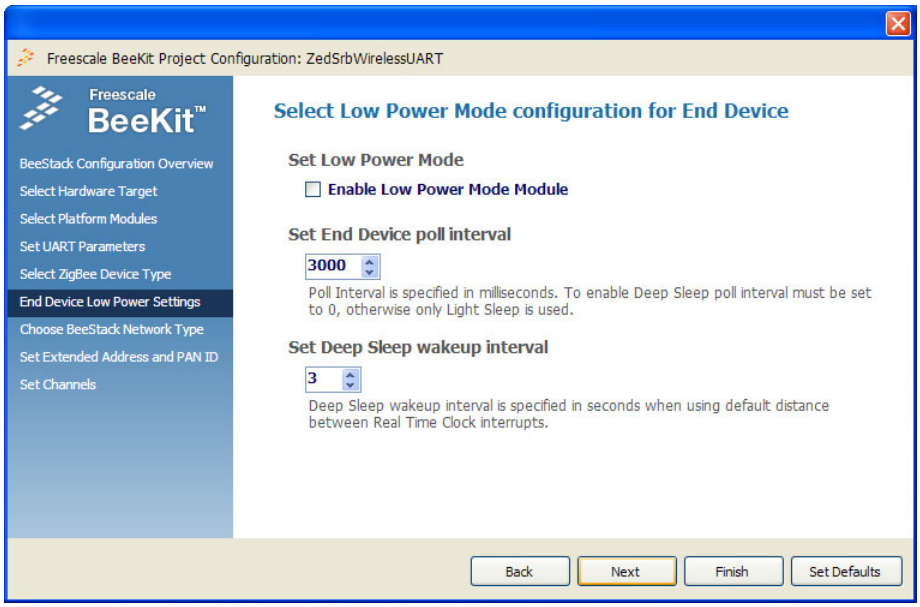

Figure 5-13. Select Low Power Mode

5. Click the Next button and the Choose BeeStack Network Type window appears. Select the same network type using the same settings used for the Coordinator as shown in Figure 5-8. (No security without mesh routing).
6. Click the Next button and set the Extended address and PAN ID the same as it was for the Coordinator as shown in Figure 5-9.
7. Click the Next button and select the same channel as the Coordinator as shown in Figure 5-10.

8. Click on the Finish button.

This completes the steps needed to create a solution/project for the Wireless UART application for a Coordinator and an End Node. The solution is now ready for export for use in CodeWarrior. The BeeKit main window should appear as shown in Figure 5-14.

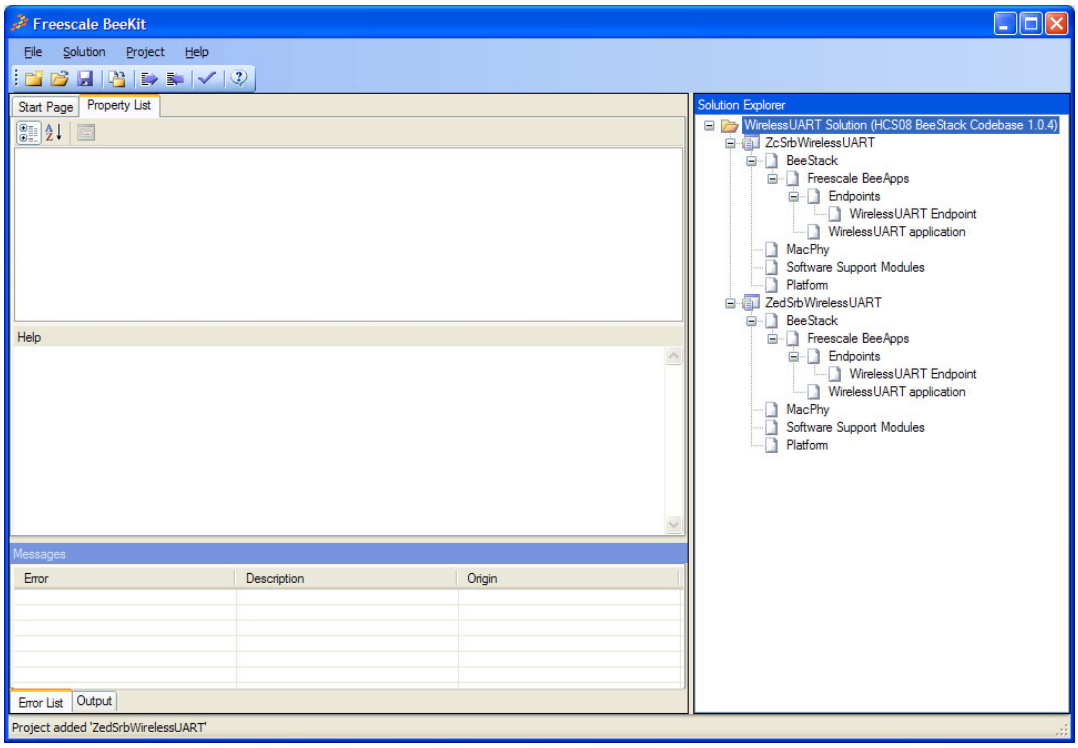

Figure 5-14. Main Wireless UART Project Page

5.1.3 Exporting the BeeKit Projects

Once the Coordinator and End Node devices to be used in the application are created as BeeKit Projects and saved as a BeeKit Solution, the files must be exported into a format that can be imported by CodeWarrior, in this case, a `.xml` file. The CodeWarrior IDE is used for compiling and debugging.

To export the saved solution, perform the following tasks:

1. From the BeeKit main window tool bar, click on Solution -> Export Solution. BeeKit verifies the internal consistency of the configuration and looks for errors such as two endpoints with the same number. If verification succeeds, the window that appears displays the created device, with a checked box as shown in Figure 5-15.

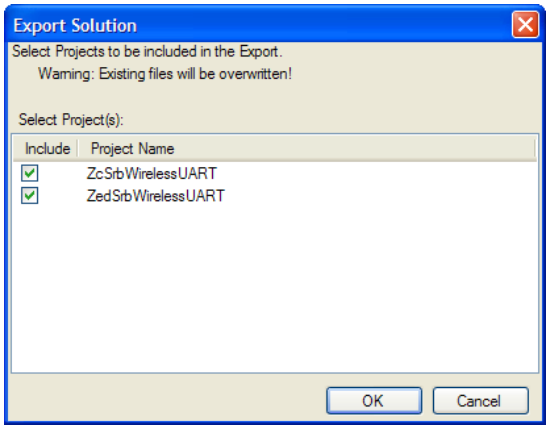

Figure 5-15. Export BeeKit Project Solution Window

2. Click on the OK button to start the export process. During the export process, BeeKit displays the Export Solution window as shown in Figure 5-16. BeeKit also displays the steps it takes during the export process in the Messages display area of the BeeKit main window.

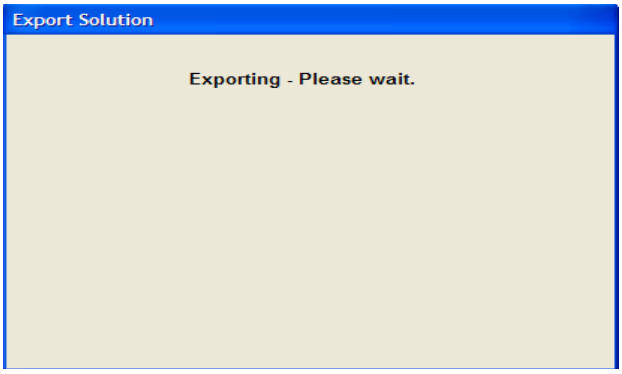

Figure 5-16. Please Wait – Exporting Project Window

3. When BeeKit completes the export, the Export Solution window goes away, but the Messages display area of the BeeKit main window still contains the steps taken during the export which users can scroll through and review. The export process is now complete.
4. Exit BeeKit by choosing File -> Exit from the tool bar.

5.2 Importing the Project into CodeWarrior

The project files created in BeeKit and saved as Solutions (Coordinator and End Node) must now be imported into CodeWarrior to build the binary output file suitable for programming into each MCU's flash.

NOTE

This section describes the steps necessary to import the Coordinator project `.xml` file. However, the steps for importing the End Node are identical with the exception of selecting the End Node `.xml` file.

To import the code for a device, perform the following tasks.

1. Start the CodeWarrior IDE, which opens to a blank window with the tool bar at the top.

2. Select File -> Import Project… as shown in Figure 5-17.

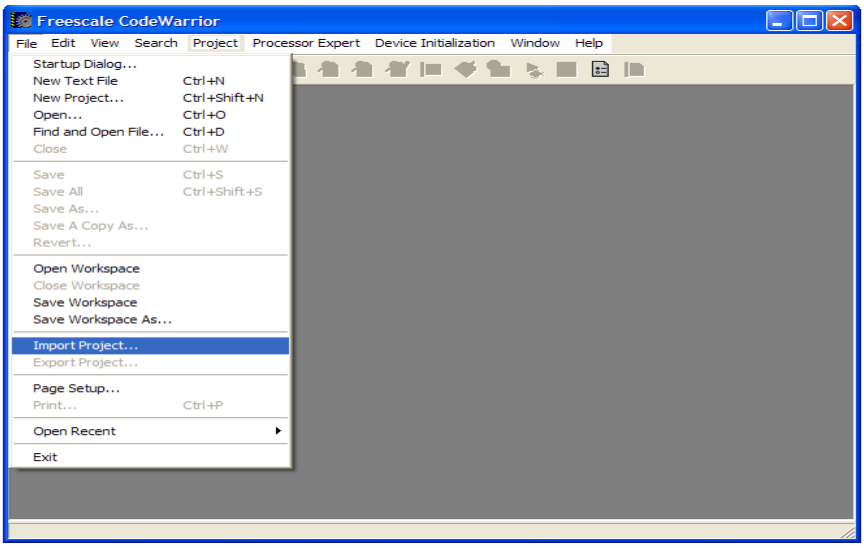

Figure 5-17. CodeWarrior Import Project…

3. Navigate to the directory created in the BeeKit export procedure.
   For the Coordinator, the directory is:

   ```
   C:\WirelessUART\WirelessUART Solution\ZcSrbWirelessUART
   ```

   For the End Node, the directory is:

   ```
   C:\WirelessUART\WirelessUART Solution\ZedSrbWirelessUART
   ```

   These directories contains the files and directories built and needed for the Coordinator and End Node devices created with BeeKit using the steps previously described in this chapter.
   The `.xml` file for the Coordinator project is `ZcSrbWirelessUART.xml`
   The `.xml` file for the End Node it is `ZedSrbWirelessUART.xml`.

   NOTE

   For the remainder of this example, the Coordinator is used, but the steps are the same for the End Node.

4. Navigate to the ZcSrbWirelessUART directory as shown in Figure 5-18, and click on the `.xml` file to be imported.

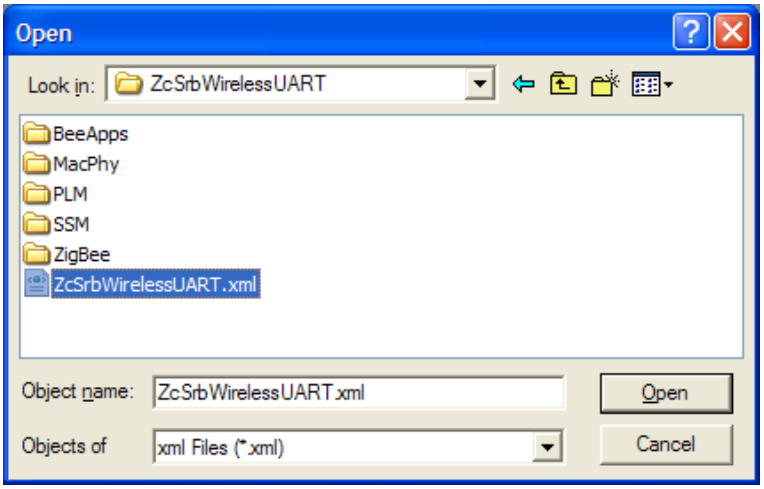

Figure 5-18. CodeWarrior Open XML Window

5. Click on the Open button. The Open window is replaced by the “Name new project as:” window in the same directory. Type in the name for the CodeWarrior project file without an extension. This example uses the same name as the `.xml` file as shown in Figure 5-18.
6. Click the Save button and CodeWarrior creates and loads the project file as shown in Figure 5-19.

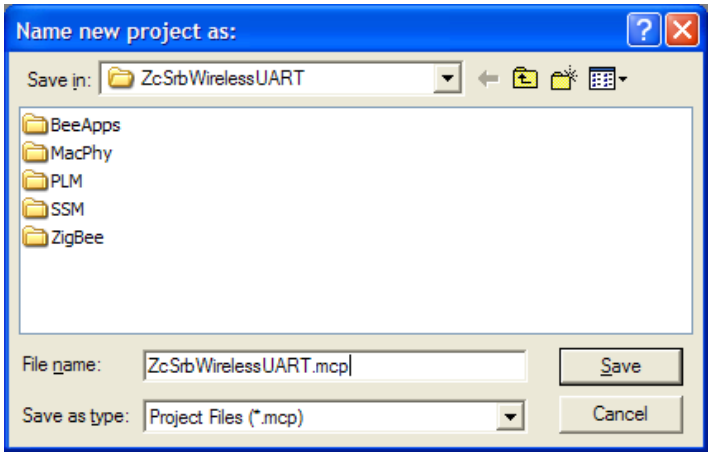

Figure 5-19. CodeWarrior Name New Project Window

7. Go back to Step 4 and repeat the import process for the End Node.

5.2.1 Building the Wireless UART Code Image Using CodeWarrior

Users can build the application binary in one of three ways:

1. Press the Make hot key F7**.**
2. Click the “make” icon (Looks like paper with a pen in hand.)**.**
3. From the menu select Project -> Make**.**

CodeWarrior reports the build progress in a window it opens and closes as shown in Figure 5-20.

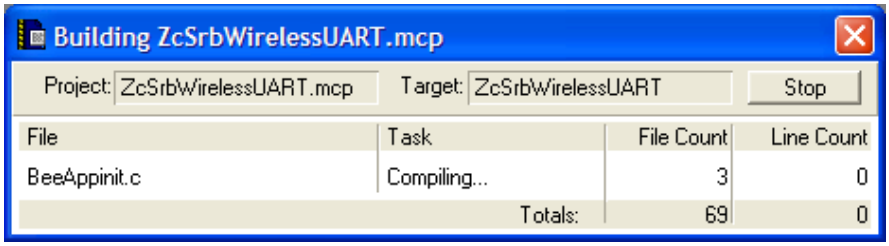

Figure 5-20. CodeWarrior Build Window

5.2.2 Loading the Wireless UART Code Image into a ZigBee Device

To use the Wireless UART application, the code images must be loaded into the Coordinator board and the End Node board. This section describes how to load the Coordinator code image to the board, but the steps to load the code image to the End Node board are the same.

Connect the P&E BDM pod to the host computer using the USB cable. A lighted blue LED indicates the BDM has power and a successful USB connection.

1. Connect the BDM pod to the board. Align pin 1 of the BDM port connector with the red wire of the flat cable connector.
   — For the SRB, the connector is J101, and pin 1 is marked with a ‘1’
2. Turn on the board.
3. The amber LED on the BDM lights up and the LED on the board also lights up. If not, switch the power off and on again on the board. Recheck the connection to the BDM port. Check the power adapter connection to the board or verify that the batteries are charged.
4. Download the compiled image to the board in one of three ways:
   a) Press the Debug hot key F5**.**
   b) Click the “Debug” icon (Looks like a bug with a green triangle.)**.**
   c) From the menu select Project -> Debug**.**

CodeWarrior opens the Debug window as shown in Figure 5-21.

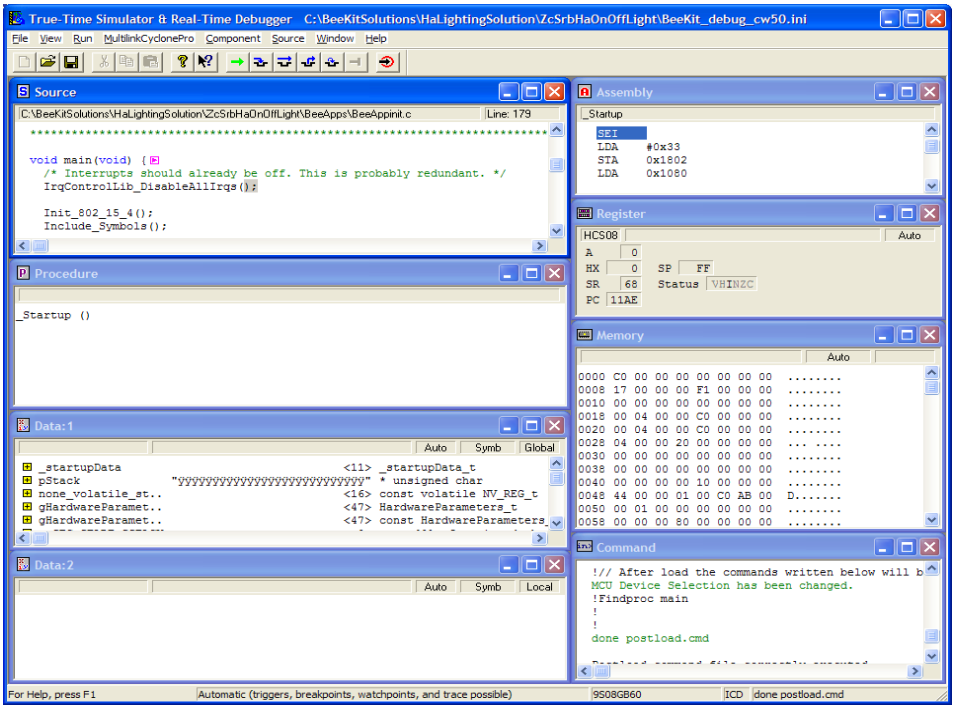

Figure 5-21. CodeWarrior Debugger Window

CodeWarrior then opens the connection manager window if this is the first programming/debug event since it was opened, as shown in Figure 5-22.

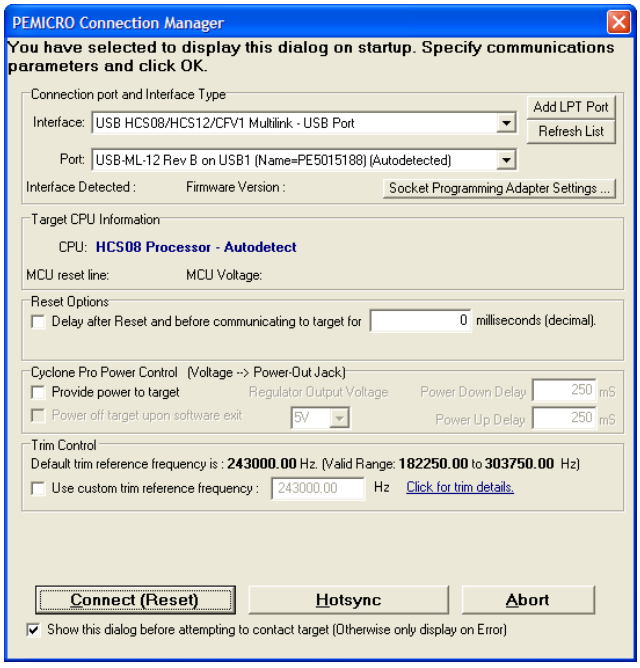

Figure 5-22. CodeWarrior Connection Manager Window

5. Click on the Connect button. The programmer window displays the running status in a window as shown in Figure 5-23 and then automatically closes this window.

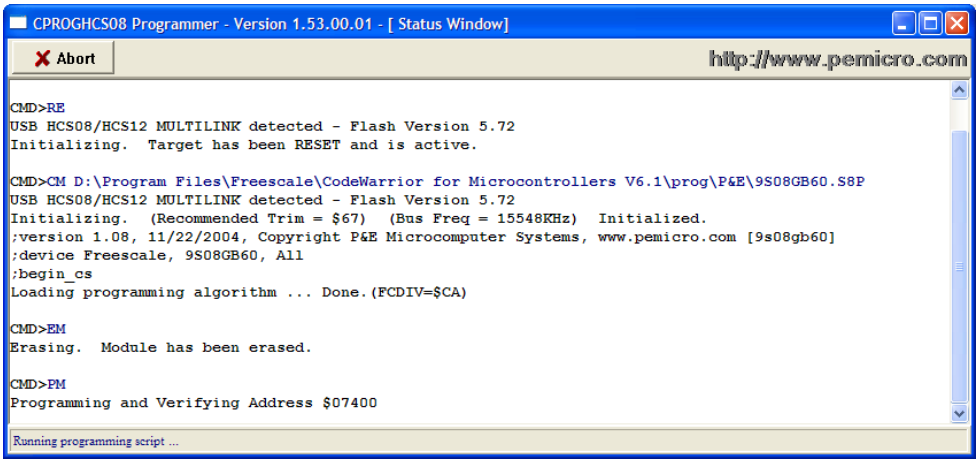

Figure 5-23. Status Window for Connection Manager

NOTE

If the Status window does not display the line “programming and verifying Address,” either the BDM or USB cables are not properly connected. Correct the problem before repeating the step.

6. Close the debugger by selecting File -> Exit.

WARNING

Exit the debug mode to avoid multiple debug appearances; having multiple appearances while setting up boards can produce unexpected results.

7. Disconnect the board and exit the CodeWarrior project, leaving CodeWarrior running.
8. Power cycle the board or press the reset button to get the board ready for use.
9. This concludes the steps needed to load the code image for the Coordinator.
10. Go to Step 1 and repeat the steps to load the code image to the End Node board but only after the End Node is imported into CodeWarrior using the steps shown in Section 5.2, “Importing the Project into CodeWarrior”.

5.3 Wireless UART Setup and Operation

The following sections show how to identify and setup the UART/USB virtual Com ports, set up Hyperterminal, start the Wireless UART application, form the network, and use the Wireless UART application.

5.3.1 Setting up the UART/USB Virtual Com Ports

1. To determine which com ports is being used by both nodes, plug a USB cable attached to a host PC into each device and powering on the boards.
2. In the Windows Device Manager, under the Ports (COM & LPT) option, two devices labeled “Freescale ZigBee/802.15.4 MAC COM Device” appear as shown in Figure 5-24. (The COM ports shown in Figure 5-24 will be different on every PC).

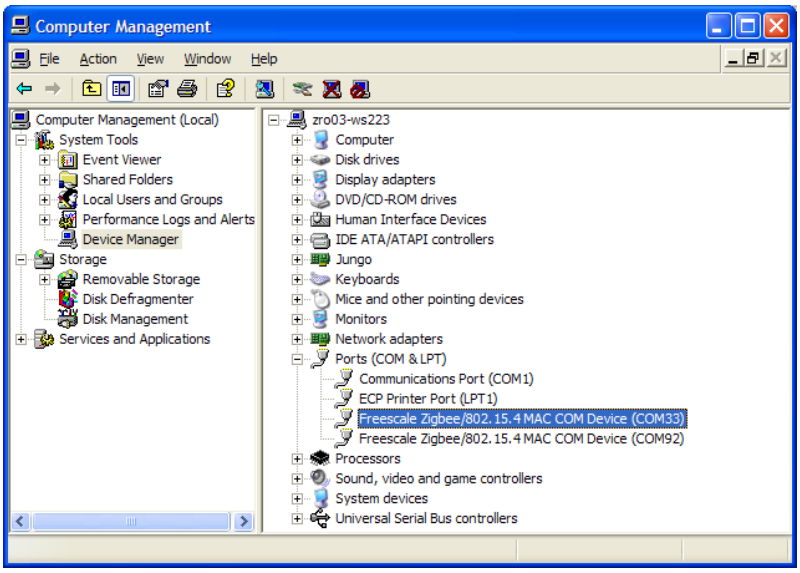

Figure 5-24. COM Ports in Device Manager

3. Using Hyperterminal, set up a Virtual Com Port for each of the two nodes. Figure 5-25 and Figure 5-26 show correct RS-232 settings used to run the Wireless UART application.

NOTE

There is no flow control used on either node.

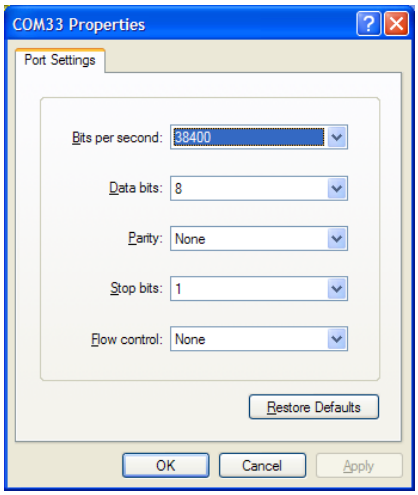

Figure 5-25. Default BeeKit RS-232 Settings

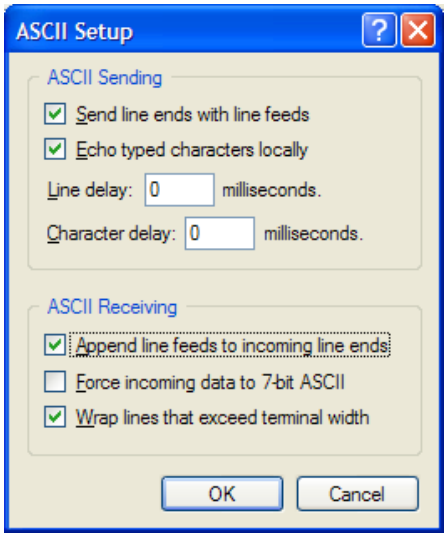

Figure 5-26. Additional Terminal Program Settings

5.3.2 Starting the Wireless UART Application

1. Connect both the Coordinator and End Node to a host PC using USB cables.
2. Open up each of the Hyperterminal programs for each virtual COM port using the Hyperterminal settings set up in Section 5.3, “Wireless UART Setup and Operation”.
3. Turn on the power for the Wireless UART Coordinator and End Node.

   LED1 flashes to indicate that the board is not on a network or that a network has not been formed ZigBee Coordinator (ZC).
4. As an option, on each node, press SW4 to select a channel other than the one selected in BeeKit. This is only needed if there are interference issues.

   With each key press, the four LEDs light up briefly to indicate the channels from 11 to 26. The LEDs display the offset from channel 11 in binary: 0000 is channel 11, and 1111 is channel 26.

NOTE

The network will not form if the two nodes are on different channels after Step 2, because the example applications are configured to look on only one channel for a network.

5.3.3 Forming and Starting the Network

1. Press SW1 on the Wireless UART node that is configured as the Coordinator.

   LEDs 1-4 light up in sequence and then go out until only LED1 is on. This is the first step in forming this small network.
2. Press SW1 on the Wireless UART node that is configured as the End Node.

   All of the LEDs on this board will flash in a series, and finally only LED1 stays lit to indicate that the End Node has joined the network that the Coordinator has formed.
3. To bind the devices, press SW3 first on one board, then within ten seconds, press SW3 on the other board.

   LED3 starts flashing, then goes solid when binding is complete. If LED3 flashes, but then goes out, the bind process failed. Turn off the boards and try again by repeating Steps 1-3 of this section.
4. On both boards, perform a long press (more than one second) of SW1 to bring both boards into run mode.

5.3.4 Using the Wireless UART Application

Once the Wireless UART application is loaded to boards, the application is started, and the network is formed and running, the Wireless UART application is ready to use.

1. Type some text in one of the Hyperterminal windows as shown in Figure 5-27.

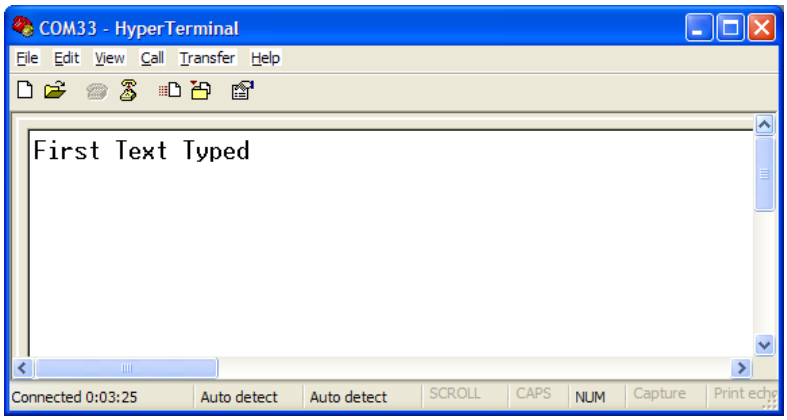

Figure 5-27. Text Typed in First Hyperterminal Window

2. The typed text is displayed as output in the second Hyperterminal as shown in Figure 5-28

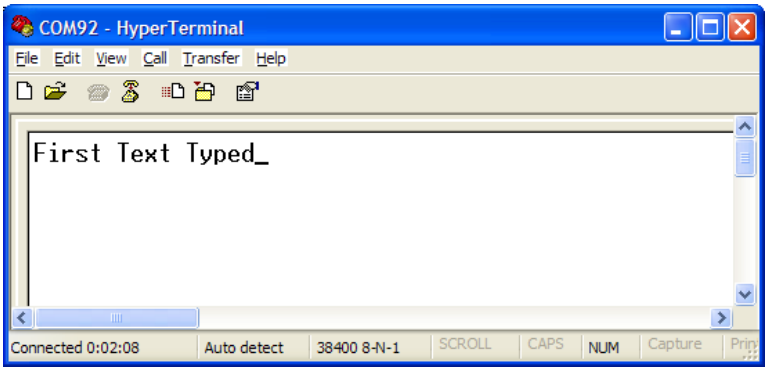

Figure 5-28. Text Echoed in Second Hyperterminal Window

Chapter 6
Example Applications

BeeStack includes example applications to:

- Enable building a simple network before writing any code
- Provide example-quality code for ZigBee applications

6.1 ZigBee Application Glossary

The following concepts explain a few of the significant elements in a ZigBee network.

Cluster	A container for one or more attributes in a command structure. If a command structure does not use attributes (such as the ZigBee Device Profile), each command and response within it is a cluster. Each cluster is identified by an enumeration that is unique within an application profile.
Cluster Identifier	An enumeration that uniquely identifies a cluster within an application profile. Cluster identifiers are designated as inputs or outputs in the simple descriptor for use in creating a binding table.
Application Profile	An agreement for messages, message formats and processing actions that enable developers to create an interoperable, distributed activity among applications that reside on separate devices. These profiles enable devices to send commands, request data and process commands and requests. For instance, a thermostat on one node can communicate with a furnace on another node. Together, they co-operatively form a heating application profile. A profile can be public, such as the Home Automation profile, so that many vendors can independently build ZigBee nodes that will interoperate. Or a profile can be private, so that a vendor can add proprietary features instead of or in addition to one or more public profiles.
Profile ID	The 16-bit number that identifies an application profile. Each unique profile must have a unique profile ID. The ZigBee Alliance issues profile IDs. Every data message and acknowledgement contains the ID of the profile that generated it. ZigBee nodes use this ID as a filter so they can discard any messages for a profile that the receiving node does not implement.
Group	An associated set of ZigBee nodes. Group addressing allows any node to send a message to all members of a group and to no other node. This allows a set of nodes to act together.
Scene	A stored set of configuration values that an incoming message can invoke, such as a light level on a dimmable lamp. Scenes work naturally in groups, so that a single

command can configure any number of related nodes for a particular purpose, such as configuring a media room for watching video.

6.2 Example Application User Interfaces

All of the Freescale ZigBee development boards have four LEDs (LED1-LED4) and four push-buttons (SW1-SW4). The example applications run on any of the boards. The NCB and 1320x-QE128EVB also have a two-line LCD screen in addition to the LEDs. All the example applications display status and command information on the LCD in addition to the four LEDs when they are built for the NCB. On every board, the push-buttons are in a row and the LEDs are in a row. On the EVB, 1320x-QE128EVB, NCB, and SRB, LED1 is above SW1, LED2 is above SW2, and so on.

NOTE

When referring to switches SW1-SW4 in this guide, it refers to the four push-buttons on the development boards. On the 1320x-QE128EVB the buttons are labeled SW2-SW5 and on the SARD they are labeled S101-S104. See Section 2.1, “HCS08 Board Details”.

6.3 Application Support Library (ASL) Keys

Most of the application use a library of common routines referred to as the Application Support Library (ASL). These routines include all of the common user interface elements of the applications, including the concept of two keyboard modes (configuration mode and run-mode), and a common set of keys that map to specific ZigBee functions.

All the example applications use the same keys to cause any activity that they all can do, such as join or leave a network. This section describes the key presses that are common to all the example applications.

The included key driver can read a long (approximately one second) keypress and a short (less than one second) keypress from each of the four keys. The example applications take advantage of this to support up to eight user-initiated events in each mode. The keypress interpretation is entirely in the application code; the applications can be rewritten to do anything.

The applications have two modes: configuration and run. The applications always start in configuration mode, where key presses cause the network formation and setup. When that is done, another key press takes the application to run mode, where it can do whatever the application is designed to do: turn a light on or off remotely, for instance.

6.3.1 Home Automation Configuration Mode

Short push button presses do the following:

- SW1 – Form/join network (form if ZC, join if ZR or ZED) with previous configuration
- SW2 – Toggle permit join (ZC/ZR) or toggle low power mode (ZED)
- SW3 – End device bind/match
- SW4 – Change radio channel: each press steps to the next higher channel and briefly displays a channel indicator on all four LEDs: 0000 is channel 11, 1111 is channel 26

Long push button presses do the following:

- Long SW1 – Go to run mode
- Long SW2 – Leave the network
- Long SW3 – Remove all bindings
- Long SW4 – Form/join network (form if ZC, join if ZR or ZED) with new configuration

6.3.2 Run Mode

Short push button presses do the following:

- SW1 – This does the main action of the application: toggle a light, etc.
- SW2 –
- SW3 – Toggle identify mode
- SW4 – Recall scene

Long push button presses do the following:

- Long SW1 – Go to configuration mode
- Long SW2 –
- Long SW3 – Send “Add group if in Identify mode” command
- Long SW4 – Send “Store scene if in Identify mode” command

6.4 Application Support Library (ASL) LEDs and Display

Most of the application use a library of common routines referred to as the Application Support Library (ASL). These routines include all of the common user interface elements of the applications, including keeping track of two displays, one for configuration mode and one for run mode. These two displays affect both the LEDs and the LCD display.

The LEDs on each board serve a variety of roles defined by a given application. Each LED can be used to indicate a variety of states, depending on whether it is on or off, flashes once or continuously, or serves as an indicator of what other LEDs are doing at the same time. This allows four LEDs to provide a wide array of information and offers the developer many choices for application testing.

6.4.1 Configuration Mode

Table 6-1. Configuration Mode LED Status

Internal State	LED1	LED2	LED3	LED4
Idle (not in a network, not joining)	Flashing	Off	Off	Off
Forming (ZC) or joining a network	All four LEDs in a flash sequence			
Network formed (ZC)	On	On	Off	Off
Network joined (ZR or ZED)	On	Off	Off	Off
Permit join toggle (ZC or ZR)		On/Off		

Table 6-1. Configuration Mode LED Status

Internal State	LED1	LED2	LED3	LED4
End device bind request: find a bind partner			Flash	
End device bind success			On	
End device bind failure			Off	
Low power ZED in light sleep	Off	On	Off	Off
Low power ZED in deep sleep	Off	Off	Off	On

When SW4 is pressed, the example applications change the channel to the next higher one and display a pattern on all four LEDs (Table 6-2) for one second before returning to the previous display pattern.

Table 6-2. Channel to LED State

Channel Number	LED1	LED2	LED3	LED4
11	Off	Off	Off	Off
12	Off	Off	Off	On
13	Off	Off	On	Off
14	Off	Off	On	On
15	Off	On	Off	Off
16	Off	On	Off	On
17	Off	On	On	Off
18	Off	On	On	On
19	On	Off	Off	Off
20	On	Off	Off	On
21	On	Off	On	Off
22	On	Off	On	On
23	On	On	Off	Off
24	On	On	Off	On
25	On	On	On	Off
26	On	On	On	On

6.4.2 Run Mode

In run mode, if the node is acting as a light, one or more of the LEDs acts as the light, but that behavior is specific to the individual application, so it is described later in this section. The LED behavior across all example applications is shown in Table 6-3.

Table 6-3. Run Mode LED Behavior (All Example Applications)

Internal State	LED1	LED2	LED3	LED4
Normal application display	Off	Off	Off	Off
Add group			Blink	
Store scene				On
Identify mode			Flash	

6.4.3 OnOffLight

This application acts as a lamp that can be turned on or off, but it cannot be dimmed.

6.4.4 Keyboard

This application allows the user to press a switch that toggles an LED on the same board. This feature is intended to allow the user to verify that the board is in the run mode and operating properly regardless of the state of any other node in the network.

- SW1 – Toggle local light (if on, turn off; if off, turn on)

6.4.5 Display

- The local light in this application is LED2.
- LED2 represents the state of the light

6.5 OnOffSwitch

A common use of a home automation lighting control is turning on and off a remote entity. It can run in any ZigBee node type, including a ZED with low power operation enabled (a “sleepy” end device). This application includes the ability to request that the receiving node acknowledge the command.

6.5.1 Keyboard

In run mode, key presses do the following:

- SW1 – Toggle remote light
- SW2 – Turn on remote light with acknowledgement
- Long SW2 – Turn off remote light with acknowledgement

6.5.2 Display

The LEDs in run mode do this.

- LED2 – Toggles with each toggle command (this can get out of synchronization with the remote light because the remote light's state can also be changed by other methods)
- LED2 on – Sent On Light Command with acknowledge
- LED2 off – Sent Off Light Command with acknowledge

6.6 Dimmable Light

The dimmable light application uses LED2-LED4 to represent a light with four states.

6.6.1 Keyboard

- SW1 – Decrease local light's level
- SW2 – Increase local light's level
- Long SW2 – Toggle local light on/off

6.6.2 Display

Once in run mode, the dimmable light control configuration makes the following LED assignments to indicate low through high lighting intensity.

- All LEDs off – Light off
- LED2 on – Low light intensity
- LED2 and LED3 on – Medium light intensity
- LED2, LED3 and LED4 on – Maximum light intensity

6.7 Dimmer Control Switch

This controls the dimmable light.

6.7.1 Keyboard

- SW1 – Decrease level of light
- SW2 – Increase level of light
- Long SW2 – Toggle light on/off

6.7.2 Display

LED2 toggle – When sending an increase or decrease message to a dimmable light

6.8 Thermostat

This application represents a device to control the temperature on a heating, ventilation, or air conditioning (HVAC) system. The thermostat uses two values of the temperature: a local value: Local Temperature (LT) and a desirable value: Desirable Temperature (DT). The difference between the two values establishes the state of the thermostat. That is, whether it is cooling, heating, or idle. When the LT is below the DT by a few degrees, the thermostat will be in heating mode (HeatOn). When DT is greater than the LT, the thermostat will be in cooling mode (CoolOn). When the two values are close, the thermostat is idle. The exact limits that set the mode are stored internally using the OccupiedCoolingSetpoint and OccupiedHeatingSetpoint attributes that are computed based on the value set externally for the DT. When the Thermostat is bound with a temperature sensor that is configured to report the sensed temperature, the thermostat will receive notifications from the other device and will update the LT to the value received from the sensor.

6.8.1 Keyboard

- SW1 – Decrease Desirable Temperature
- SW2 – Increase Desirable Temperature
- Long SW2 – Toggle (^{o}F / ^{o}C)

6.8.2 Display

- LED2 flashing – LT is below -5^{o}C
- All LEDs Off – LT is between -5^{o}C and 10^{o}C
- LED2 On, LED3 and LED4 Off – LT is between 10^{o}C and 20^{o}C
- LED2 and LED3 On, LED4 Off – LT is between 20^{o}C and 30^{o}C
- LED2, LED3 and LED4 On – LT is between 30^{o}C and 40^{o}C
- LED2, LED3 On and LED4 Flashing – LT is above 40^{o}C
- While changing the DT using the switches, the state of the LEDs as specified above will reflect the value of DT until LT is updated again
- LCD “LT=temp ^{o}F / ^{o}C” - When LT acquired from the temperature sensor is between the OccupiedCoolingSetpoint and OccupiedHeatingSetpoint.
- LCD “LT=temp ^{o}F / ^{o}C HeatOn” - When LT acquired from the temperature sensor is lower than the OccupiedHeatingSetpoint.
- LCD - “LT=temp ^{o}F / ^{o}C CoolOn” - When LT acquired from the temperature sensor is higher than the OccupiedCoolingSetpoint.
- LCD - “DT=temp ^{o}F / ^{o}C” – When changing the Desirable Temperature.

6.9 Temperature Sensor

The temperature sensor application uses the hardware temperature sensor found on the MC1321x-SRB boards or a simulated temperature value when the application runs on the other boards or the hardware sensor is disabled. Users should note that the temperature reported by the hardware sensor can be slightly different than the actual room temperature.

The Temperature Sensor can send its temperature value using indirect mode, that is, using the local binding table. It typically sends this data when another node has requested that the Temperature Sensor enter reporting mode using the ZigBee Cluster Library (ZCL) report attribute command or when the sensor is indicated to report using a Long SW2 press.

Use SW3 End Device Bind or the Configuration Tool application to bind the Temperature Sensor to a Thermostat, so that the Thermostat can receive temperature reports from the Sensor.

6.9.1 Keyboard

- SW1 – Decrease simulated temperature when hardware sensor is not used
- SW2 – Increase simulated temperature when hardware sensor is not used
- Long SW2 – Turn the Report Temperature On, and begins sending the temperature using indirect transmission (reports are sent to the devices to which the sensor is bound).

6.9.2 Display

- LED1 flashing – sensor is sending OTA temperature reports to bound devices
- LED2 flashing – sensed temperature is below -5^{o}C
- LED2, LED3, LED4 Off – sensed temperature is between -5^{o}C and 10^{o}C
- LED2 On, LED3, LED4 Off – sensed temperature is between 10^{o}C and 20^{o}C
- LED2 and LED3 On, LED4 Off– sensed temperature is between 20^{o}C and 30^{o}C
- LED2, LED3 and LED4 On – sensed temperature is between 30^{o}C and 40^{o}C
- LED2, LED3 On and LED4 Flashing – sensed temperature is above 40^{o}C

6.10 HA Generic Application

The generic application is a device with no unique features. It uses only the common user interface.

6.11 HA Range Extender

The range extender is a router with no unique features. It uses only the common user interface.

6.12 HA Combined Interface

The Combined Interface application acts as a gateway between a personal computer and the ZigBee network by using the ZTC library for UART communication. The run mode of the application does not include any functionality except the “Go to configuration mode” which is enabled by a long press of SW1 of the evaluation board. The behavior of the switches and LEDs in Configuration mode is the same as all other Home Automation applications as shown in Section 6.3.1, “Home Automation Configuration Mode” and Section 6.4.1, “Configuration Mode”.

6.13 Manufacturer Specific Applications

The Wireless UART and Accelerometer applications are examples that are not part of the Home Automation Application Profile. They are not part of any other ZigBee public profile and therefore are considered to be custom application profiles, which the ZigBee Alliance calls “Manufacturer Specific”. These applications are only meant to interoperate among devices from a single manufacturer or another manufacturer if permitted by the primary manufacturer.

6.13.1 Manufacturer Specific Configuration Mode User Interface

The following Configuration Mode User Interface is slightly different from that for the Home Automation applications.

Short push button presses do the following:

- SW1 – Form/join network (form if ZC, join if ZR or ZED)
- SW2 – Leave network
- SW3 – End-device-bind (bind/unbind toggle with another node) / Match
- SW4 – Choose channel, walks through all channels (uses 4 LEDs for a binary number 0- 15 which equates to channels 11-26)

Long (greater than 1 second) push button presses do the following:

- Long SW1 – Go to Run Mode
- Long SW2 – Toggle Permit Join
- Long SW3 – Unbind all
- Long SW4 – Reset node to factory defaults

6.14 Wireless UART

The Wireless UART can replace a serial cable with two or more ZigBee nodes. It can multi-hop for extended range using mesh routing. ZigBee is half-duplex and generally low data rate, so it is not applicable for all serial applications.

6.14.1 Keyboard

This application uses the standard ASL user interface, with a few exceptions. Since the Wireless UART does not have scenes or identify mode, SW3, SW4, and Long SW4 do not perform the same functions that they perform in most other applications.

This wireless UART application can replace a wire, and it supports HW flow control to make sure it does not miss messages. Flow control is only supported in one direction; the ZigBee node will assert flow control to tell the PC to stop transmitting. The ZigBee node does not recognize flow control asserted by the PC.

The Wireless UART application can be char or block mode, and/or in loop back mode.

In char mode, every character that is received from the UART is immediately sent to the remote node. This is intended for keyboards and other, similar applications.

In block mode, bytes from the UART are gathered together and sent as a block to the far end. This mode makes more efficient use of the available radio bandwidth (more bytes of payload transferred per ZigBee packet), and is intended for file transfers.

In loopback mode, every byte received from the UART is sent back out the UART. No data from the UART is sent over the air.

- SW1 – Toggle char vs. block mode
- SW2 – Toggle loopback mode on/off
- SW3 – Not used
- SW4 – Toggle through baud rates 1200,2400,4800,9600,19200,38400

- Long SW1 – Go to Configuration Mode
- Long SW2 – Not used
- Long SW3 – Add Group
- Long SW4 – Not Used

6.14.2 Display

The LEDs in application mode will display as follows:

- LED1 – on when in char mode, off when in block mode
- LED2 – on when in loopback mode, off otherwise
- LED3 – toggles on each packet sent to the remote node
- LED4 – toggles on each packet sent to the PC via the UART

- LCD (on NCB) indicates # of packets transferred
- line LCD 1: “Rx 1234 Tx 1234” (in hex)
- Line 2 on LCD will not be changed as this is used for the standard ASL interface.

6.14.3 Baud Rate Display User Interface

6.14.3.1 Application Mode

Each time SW4 is pressed to change baud rates, the LEDs will change for a moment to display the new baud rate as shown in the following table:

Table 6-4. Baud Rate LED Display

Baud Rate	LED1	LED2	LED3
1200	On	Off	Off
2400	Off	On	Off
2400	On	On	Off
9600	Off	Off	On
19200	On	Off	On
38400	Off	On	On

The default baud rate is 38400. The serial port is always set to 8N1:

- Eight data bits
- No parity
- One stop bit

6.15 Generic App

The Generic App application measures and displays movement of a board in three axis using the built-in accelerometer in the SRB and SARD boards, and shows this movement by lighting LEDs on another display board. Use SW3 to find the other board so that the display of movement takes place on the remote board instead of the local board.

The purpose of the Generic App (also called Accelerometer) application is to:

- Show how to interact with BeeStack 2006 without using the ASL interface
- Demonstrate a very simple application as a starting point for custom applications
- Show how to interact with hardware in the multi-tasking BeeStack environment

The Generic App application does not use the ASL interface. Instead, it blinks LED1 to indicate that it is not on the network when it first starts. When button 1 is pressed, it forms the network as a ZigBee Coordinator (ZC), or it joins a network as a ZigBee Router (ZR) or a ZigBee End Device (ZED). It does NOT have the application vs. configure mode like the ASL does, but instead uses the LED and keyboard routines directly from the driver. It also does not use the NVM interface. When not on hold, the accelerometer transmits once every 500 milliseconds to the remote display.

NOTE

Program the Generic App application into two boards. One board will be the accelerometer and the other board will be the remote display. Only press SW3 on the board that will be the accelerometer. The other board, without any key presses, will then be the display for the accelerometer.

6.15.1 Keyboard

In run mode, key presses do the following:

- SW1 – join/form the network
- SW2 – Walks through 5 tilt states (all LEDS off, 1 -4 on). SARD and SRB boards ignore this button and use actual accelerometer hardware.
- LSW2 – Toggles “hold” (starts/stops detecting and transmitting accelerometer data). “hold” is the default state when the board boots.
- SW3 – Find accelerometer display using match descriptor (LED3 will light on both boards)
- SW4 – Toggles through display axis (X, Y or Z). On NCB, all axes are displayed on LCD, but only 1 axis on LEDs. Will briefly display LED1=X, LED2=Y, LED3=Z to indicate which axis will be displayed on the LEDs.

6.15.2 Display

The LEDs in run mode do the following:

- LED1 blinks when not on network.
- LED1 is solid when joined/formed the network.
- LED3 is solid on when display is found.
- LEDs 1-4 indicate 5 states of tilt for the currently displayed access (0 to 4 LEDs on). All LEDs off means upside down on Z-axis. All LEDs on means right side up on Z-axis.